HP 3000 Computer Systems

HP 3000 SERIES 64/68/70 COMPUTER SYSTEMS MICROCODE MANUAL

FOR HP REPAIR USE ONLY

HEWLETT PACKARD
19447 PRUNERIDGE AVENUE, CUPERTINO, CA 95014

Part No. 30140-90045
E1086

Printed in U.S.A. 10/86

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the most recent version of each page in the manual. To verify that your manual contains the most current information, check the dates printed at the bottom of each page with those listed below. The date on the bottom of each page reflects the edition or subsequent update in which that page was printed.

Second Edition . . . . . October 1986

Effective Pages	Date
all	Oct 1986

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

First Edition . . . . . Jan 1984
Second Edition . . . . . Oct 1986

CONTENTS

PREFACE

This manual contains a complete listing of MPE V UB Delta-1 (Version G.02.01) microcode for the HP 3000 Series 64/68/70 computers. It is intended to be used by Hewlett-Packard TSEs, SEs, CEs, and other support personnel for troubleshooting purposes. The manual listing is divided into fourteen major parts which are listed in the table of contents. Miscellaneous commands and smaller code groups will be found between sections as a result of the page-segmented microcode architecture.

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

    2           $LLUT
    3           ********************************************************************
    4           *                                                                  *
    5           *                                                                  *
    6           **                                                                 *
    7           ***                                                                *
    8           ****                    ICF/55 Microprogram                        *
    9           *****                                                              *
   10           ****                Version with CST extension 8k                  *
   11           ***                                                                *
   12           **                                                                 *
   13           *                                                                  *
   14           *     See location !11 for embedded version number                 *
   15           *                                                                  *
   16           * REV 0709A    7/9/82   CHANGED DIMS (DEFERRED INTERRUPT SUBROUTINE) *
   17           *                       THAT THE CPU WILL GO BUSY WHEN ENTERING; AND *
   18           *                       CHECK FOR MSG INTERRUPT BEFORE EXITING.      *
   19           *                                                                    *
   20           * REV 0831A    8/31/82  CHANGED IN CLD1 TO READ 2ND BYTE OF COLDLOAD *
   21           *                       DEVICE ID AND DO UNLISTEN/UNTALK TO HPIB.    *
   22           *                       FIX FOR IOWA CITY PROBLEM WITH BFD COLDLOAD   *
   23           *                                                                    *
   24           * REV 0903A    9/3/82   CHANGED CMPS/CMPT COBOL INSTRUCTION TO FIX   *
   25           *                       COMPARE OF LONG TO SHORT (SR 33445, BOEING). *
   26           *                                                                    *
   27           *  *** CHANGES FOR CST EXTENSION MADE TO MICROCODE ***               *
   28           *                       AFFECTED INSTRUCTIONS: PCAL,EXIT,IXIT,SCAL   *
   29           *                       LLBL,XBR,PARC,LST,SST,LAP,LAIP.              *
   30           *                       ALSO AFFECTS: INT8,INT9,CLFN,STTV,CSTV.      *
   31           *                                                                    *
   32           * REV PR0118A 1/18/83  CHANGED IN STOREBYTE SBS&SBD (SR 4700-15545) *
   33           *                       FIX IN MPYD(SRD1) SR 9999-201353,4700-021048 *
   34           *                                                                    *
   35           * REV PR2305A 2/4/83   CHANGED IN MWOL TO FIX MVBW BOUNDS CHECKING   *
   36           *                       BUG. SR 4700-15461 LONGS DRUGS. ALSO CHANGED *
   37           *                       BNDV AND NRM TO REPORT PARM IN SPOA CORRECTLY*
   38           *                       SR 21287, FOUND BY RON JONAS. ALSO CHANGED   *
   39           *                       MICROCODE REV CODE TO BE STANDARD HP DATECODE*
   40           *                                                                    *
   41           * REV PR2306A 2/8/83   CHANGED IN DUMP TO WRITE SP1B (HALT CODE) TO *
   42           *                       MEM LOC %1514 FOR DPAN. SR 32447             *
   43           *                                                                    *
   44           * REV 2307A    3/9/83   CHANGED IN ALGN AND VMLD TO HANDLE CASE IN   *
   45           *                       WHICH THE LAST BYTE OF THE SOURCE FIELD      *
   46           *                       SHARES THE WORD WITH THE FIRST BYTE OF THE   *
   47           *                       FIELD.  ALSO FIXED BUG IN SETTING O'PUNCH/   *
   48           *                       NOT O'PUNCH XR2 FLAG. SR4700-034223(EBI CORP)*
   49           *                       ALSO FIXED MDWO SO THAT THE SIGNIFICANCE     *
   50           *                       TRIGGER IS SET CORRECTLY.  SR 4700-24075     *
   51           *                                                                    *
   52           * REV PR2310A 3/29/83  CHANGED IN ALGN TO BACK OUT 2307 CHANGE TO   *
   53           *                       OVERPUNCH-XRB2 LOGIC.  FIXED IN 'FILL' TO    *
   54           *                       CORRECTLY HANDLE OVERPUNCH CONDITIONS        *
   55           *                                                                    *
   56           * REV PR2311A 4/5/83   CHANGED IN MWOL TO LEAVE THE TOS REGS WITH   *
   57           *                       CORRECT VALUES IF TRAP TAKEN AND NO BNDV     *
```

```
RECORD  C.S.                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

   58           *                                                                    *
   59           * REV PR2312A 4/19/83  CHANGED CRTW TO CLEAR XRB101 ON COLDLOAD TO   *
   60           *                      DISABLE DCU LOGGING INTS UNTIL ENABLE BY MPE  *
   61           *                                                                    *
   62           * REV CX2314A 5/23/83  INITIAL RELEASE OF CST EXTENSION MICROCODE.   *
   63           *                      INCLUDES FIX TO ENDP IN WHICH THE WRONG ALU   *
   64           *                      WAS CALLED OUT--RESULTING IN ALWAYS LOGICAL   *
   65           *                      MAP MODE FOR THIS INSTRUCTION.                *
   66           *                                                                    *
   67           * REV CX2317A 7/11/83  FIXED LAP & LAIP FOR STACK HANDLING PROBLEM.  *
   68           *                      SR 9999-201395 WAS THE REPORTED PROBLEM IN    *
   69           *                      IN NON CSTX CODE.  THIS VERSION REWRITES      *
   70           *                      LRA-PCALO LOGIC FOR CSTX TO INCLUDE THIS FIX. *
   71           *                      WHEN PUSHING TO Z+1.  FORMERLY ALLOWED THIS.  *
   72           *                                                                    *
   73           * REV CX2318A 7/12/83  FIXED IN PSHM AND PSM2 TO STACK OVERFLOW      *
   74           *                      ALSO FIXED IN PCL7 TO FIX MAPFLAG PROBLEM.    *
   75           *                                                                    *
   76           * REV CX2319A 8/9/83   FIXED IN MMAP TO REWRITE XRA12 WITH 4000H     *
   77           *                      ON POWERFAIL RESTART.  ALSO CHANGED IN SIO1   *
   78           *                      AND CCPP TO REMOVE ALL USE OF RWX6 SPECIAL-A  *
   79           *                      (WHICH DOES NOT WORK DUE TO HARDWARE PROBLEM) *
   80           *                      REPLACED W/ ROX3. KANSAS CITY/3M/DMD PROBLEM  *
   81           *                      WITH IOB CACHING. ALSO REARRANGED PARC TO DO  *
   82           *                      AUDIT TRAIL FOR DEBUG OF POSSIBLE HDWE PROB.  *
   83           *                                                                    *
   84           * REV CX2320A 8/10/83  REMOVED AUDIT TRAIL IN PARC INSTRUCTION.      *
   85           *                      LEFT PARC REARRANGED AS IN PR2319A.  THIS IS  *
   86           *                      TO GET AROUND A HDWE PROBLEM WHICH SEEMS TO   *
   87           *                      CAUSE FAILURE BASED ON WHERE IN WCS THE CODE  *
   88           *                      RESIDES. EXACT CAUSE OF FAILURE IS UNKNOWN.   *
   89           *                                                                    *
   90           * REV CX2324A 9/29/83  MODIFIED MMAP TO DELETE CHANGE OF CX2319A     *
   91           *                      WHICH CAUSED PFAIL TO DIE.  CHANGED PON TO    *
   92           *                      WRITE 4000 INTO XR12.  CHANGED ENDP TO GET    *
   93           *                      AROUND POSSIBLE HDWE PROBLEM WITH LITERALS.   *
   94           *                      ALSO FIXED XEQ TO BACK UP P ON STACK O'FLOW.  *
   95           *                      ALSO FIXED BUG IN ABSD FOUND IN DESK CHECK.   *
   96           *                                                                    *
   97           * REV CX2326A 10/17/83 FIXED IN LAIP TO ADD A NOP LINE SO THAT SF1   *
   98           *                      WILL NOT BE KILLED BY FAKE NEXT.              *
   99           *                                                                    *
  100           * REV CX2328A 11/16/83 FIXED PCL3 CHECK FOR SEG > #ENTS              *
  101           *                                                                    *
  102           * REV CX2330A 11/29/83 FIXED READ STATUS COMMAND IN CS80 DISC CHANL  *
  103           *                      PROGRAM USED FOR LOADS, STARTS, AND DUMPS.    *
  104           *                      WAS 00, SHOULD HAVE BEEN 0D.  ALSO FIXED MVBW *
  105           *                      TEST FOR SPLIT STACK.  SHOULD HAVE USED "SUB",*
  106           *                      NOT "CAD".  ALSO FIXED BUG IN ENDP WHERE WE   *
  107           *                      GOT A BOUNDS VIOLATION WHEN WE SHOULDN'T      *
  108           *                      HAVE AFTER A NON-MATCHING PARAGRAPH NUMBER.   *
  109           *                                                                    *
  110           * REV CX2332A 12/10/83 Added two NOP lines after ABSD to stop        *
  111           *                      WCS PARITY ERRORS from ABSD during micro-     *
  112           *                      instruction pre-fetch cycles                  *
  113           *                                                                    *
```

```
RECORD  C.S.                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  114           * REV CX2334A 1/14/84  Fixed DUMP failure after "Invalid Address -- *
  115           *                      Module 5" errors.  (Note: NIR will be invalid *
  116           *                      in DUMPs made after this kind of sytem error)*
  117           *                      Added DSJ to channel program for mag tape     *
  118           *                      after read of transfer count to keep Antelope*
  119           *                      and Buckhorn happy during Loads and Starts.   *
  120           *                      Also, fixed CSEG problem with NRPGMSEGS.      *
  121           *                                                                    *
  122           * REV CX2336A 2/24/84  Fixed CSEG problem with LSTT address.         *
  123           *                                                                    *
  124           * REV CX2420A 6/1/84   Fixed CVDB in response to SR#'s 4700-099259,  *
  125           *                      4700-069906, and 9999-201617.  Also fixed     *
  126           *                      the SLD instruction (SR# 4700-105114).        *
  127           *                                                                    *
  128           * REV CX2428A 7/30/84  Fixed Bounds Violation problem in response    *
  129           *                      to SR# 4700-13326 (Anderson College).         *
  130           *                                                                    *
  131           * REV CX2511A 3/19/85  Changed CS-80 channel program (used by Start,*
  132           *                      Load, & Dump) to begin with Device Identify   *
  133           *                      instead of Selected Device Clear.  This is in *
  134           *                      response to the 7933/5 HPIB <Clear> <Clear>    *
  135           *                      disc spindown problem due to the 793X MR 5.0 *
  136           *                      microcode changes that make a <Clear> do a    *
  137           *                      recalibrate.                                  *
  138           *                                                                    *
  139           * REV CX2518A 5/16/85  FIXED 'BNDE RSB' INTERACTION PROBLEM IN LDWC *
  140           *             CJS/EBD  (BNDE WAS NOT BEING PROCESSED PROPERLY)       *
  141           *                                                                    *
  142           * REV TX2526A 7/29/85  Fixed generalized BNDE/RSB problem with TICB *
  143           *             EBD/BEL  construct to slow the RSB down, some places  *
  144           *                      drop the RSB down one line if not slower.     *
  145           *                                                                    *
  146           * REV CX2527A 8/6/85   Re-wrote FLSH (the FLUSH instruction) to      *
  147           *             CJS      flush the new 128Kb cache or the old cache.  *
  148           *                                                                    *
  149           * REV CX2545A 11/4/85 Second release to U-MIT.  Fixes the MM-3000    *
  150           *             JLT/EBD CSD COBOLII hang, and SUPRTEST ABORT problems *
  151           *                     SR#S 4700189167, 4700190629, AND 4700191379.  *
  152           *                     Suspected cause: pipeline interaction with    *
  153           *                     TICB (skip) construct on RSB/BNDE (STATE FARM)*
  154           *                                                                    *
  155           * REV TX2553A 12/30/85 Added XP performance enhancement microcode.   *
  156           *             MRG/BEL  This includes: new LST/SST,ISTR,ERON,ERXT,    *
  157           *             JLT/EBD  XGDB,TIMR,TMRQ,MSTA,and ENPF.  These changes *
  158           *                      don't appear until the "Enable Performance"   *
  159           *                      (ENPF) instruction has been executed.  Also   *
  160           *                      added a feature to DUMP which puts the ucode  *
  161           *                      datecode into the word at memory location     *
  162           *                      SYSGLOB Extension %20 so it can be printed.   *
  163           *                      Commentary is at header of each instruction.  *
  164           *                                                                    *
  165           * REV CX2603A  1/16/86 Documentation update.  Also added NOPs to     *
  166           *              BEL     each code break so never causes freezes.      *
  167           *                                                                    *
  168           * REV CX2604A 1/23/86 Fixed a bug in ERXT in the section of code     *
  169           *              JLT    that deals with aborting processes.           *
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  170           *                                                                    *
  171           * REV CX2605A 1/30/86 Fixed the version save code in dump, turning   *
  172           *             BEL     off WCS parity checking when we grab the       *
  173           *                     version number from location 11 as it enters   *
  174           *                     the pipeline.  Parity checking turned back on  *
  175           *                     immediately following  this by the JSB/RSB.    *
  176           *                                                                    *
  177           * REV CX2606A 2/10/86 Fixed ERXT so that it passes the correct       *
  178           *             JLT     parameters to Abort.                           *
  179           *                                                                    *
  180           * REV CXA606A 8/18/86 Fixed CVBD to call LEN with correct            *
  181           *             CYL     parameters.                                    *
  182           *                                                                    *
  183           * REV CX2625A 6/17/86 Fixed the routine TCLK in order to be able to  *
  184           *             JLT     keep track of time correctly even when the     *
  185           *                     systems are flushing constantly.               *
  186           *                                                                    *
  187           * REV CX2635A 8/26/86 Second release to UB-MIT.  Contains fixes to   *
  188           *          CYL/JLT    SR's concerning the following routines:        *
  189           *                     PCAL, PCL3, DCMD, IOMS, EBND, EFOV, XBR,       *
  190           *                     ASEG, ALGN, SETR, AND CVBD.  Also contains     *
  191           *                     the Prarie Project changes that allow the      *
  192           *                     HP9144 to be used as a coldload device, the    *
  193           *                     fix to TCLK that fixes the problem of time     *
  194           *                     loss in S70 A systems, and the fix to the      *
  195           *                     7MB hang problem.                              *
  196           *                                                                    *
  196 1         * REV CX2638A 9/26/86 Documentation update for 2635.                *
  196 2         *             JLT                                                    *
  196 3         *                                                                    *
  197           **********************************************************************
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  199           ********************************************************************
  200           *                                                                  *
  201           *    Unimplemented Instruction Entry Point                         *
  202           *                                                                  *
  203           ********************************************************************
  204           *
  205   0000    TRAP          JSZ  TRP       UNC            ADD                    Unimplemented instruction entry point
  207           *
  208           ********************************************************************
  209           *                                                                  *
  210           *    Power On Entry Point                                          *
  211           *                                                                  *
  212           ********************************************************************
  213           *
  214   0001    PONE C606     ADDL     RA                   JSL  PON          UNC   Power on entry point; RA := @ZI
  216           *
  217           ********************************************************************
  218           *                                                                  *
  219           *    Cold load entry point                                         *
  220           *                                                                  *
  221           ********************************************************************
  222           *
  223   0002    CLDE          INC      XR15                 JSL  CKSM BNKD    NF1   Set LOAD bit in XRA15; BNKD := 0 (for #CKSM)
  225                                                                                   checksum WCS & LUT and initialize module
  226                                                                                   map if not coming from the STRT instr'n
  227   0003                  ADD      X                    JSL  LOAD BKX3    UNC   X := 0; BKX3 := 0, go to cold load routine
  229           *
  230           ********************************************************************
  231           *                                                                  *
  232           *    Dump entry point                                              *
  233           *                                                                  *
  234           ********************************************************************
  235           *
  236   0004    DMPE          INC  LSL XR15                 JSZ  PSHA BKX3    SRNZ  Empty TOS registers, BKX3 := 0
  238   0005                  JSZ  DMPX      UNC            ADD            SF4B       Set DUMP bit in XR15; BKX4 := 0, go dump it
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

241             ********************************************************************
242             *   NOP (No Operation) Instruction                                 *
243             ********************************************************************
244             *
245             $LUT INSTR=NOP:0 000 000 000 xxx xxx, ENTRY=NOP, DSPL=6
246             *
247     0006    NOP           ADD                         ADD                      NOP instruction
249     0007    NEXT          ADD                         ADD           STFF NEXT  STFF in case of BCCNM; NEXT
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  252           ;*********************************************************************
  253           ;    PULM pulls the top of the stack in memory and places it in      *
  254           ;    the bottom of the stack in registers.  PULM checks for stack    *
  255           ;    underflow and therefore should be called before changing        *
  256           ;    the stack or any software known registers.                      *
  257           ;*********************************************************************
  258           ;
  259    0008   PULM       SM   ADD             ROS   FFFF SM   ADDL      SM                 Read TOS in memory; SM := SM - 1
  261    0009         UBA  Q    JSZC STU1       NCRY  SR        ADD       CTR  INSR          STUN if not (SM > Q), JSB to entry which
  263                                                                                         increments SM and decrements SR;
  264                                                                                         CTR := SR for REGN store, inc SR
  265    000A              OPA  ADD       REGN        SP3B      ADD                 RSB    REGN := TOS in memory; UBB := SP3B, RSB
  267           ;
  268           ;*********************************************************************
  269           ;    PUL2 pulls two locations from the top of stack in memory        *
  270           ;    into the TOS registers.  PUL2 checks for stack underflow and    *
  271           ;    therefore should be called before changing the stack or any     *
  272           ;    software known registers.                                       *
  273           ;*********************************************************************
  274           ;
  275    000B   PUL2       SM   ADD             ROS   FFFF SM   ADDL      SM
  277
  278
  279                                                                                         Read TOS in memory; SM := SM - 1
  280    000C         UBA       CAD       RH   ROS   SR        ADD       CTR  INSR          Read (TOS - 1) in memory;
  282                                                                                         CTR := SR for REGN store, inc SR
  283    000D              OPA  ADD       REGN        UBA  Q    CAD             ICTR NEG     REGN := TOS in memory; Inc CTR for next REGN
  285                                                                                         Skip if not ((SM - 1) > Q)
  286    000E              OPA  ADD       REGN        RH        CAD       SM    INSR RSB     REGN := (TOS - 1) in memory; SM := SM - 2,
  288                                                                                         inc SR, return
  289    000F              UBA  JSZ  STUN       UNC         SM  INC       SM    DCSR         UBA := QDWN, stack underflow;
  291                                                                                         Restore SM and SR
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

293             *
294             %0010
295             *
296             ********************************************************************
297             *     The following line is for interrupts detected before         *
298             *     executing the left stackop of a paired stackop instruction.  *
299             ********************************************************************
300             *
301     0010    IR              JSZ  IRH       UNC             ADD                  Jump to interrupt handler
303             *
304             ********************************************************************
305             *     This location is used to communicate information to the DCU.  *
306             *     It should never be executed nor pulled into the pipeline.     *
307             *     Do not modify this location without knowing what the DCU is   *
308             *     expecting to find here.  The FUNC & SFNC fields contain a     *
309             *     date code in the form MMDDVb, i.e., 0929M                     *
310             ********************************************************************
311             *
312     0011        4D4A 4D20 2638 5820                  CON                      ASCII data for the DCU
314             *
315             ********************************************************************
316             *      The following line is for interrupts detected before any     *
317             *      instruction other than a paired stackop.  P is decremented   *
318             *      by one before jumping to the interrupt handler.              *
319             ********************************************************************
320             *
321     0012    IRD             JSZ  IRH       UNC   FFFF P    ADDL      P        JSB for interrupt; P := P - 1
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 324           ********************************************************************
 325           *                                                                  *
 326           *     There are two identical copies of the SR preadjust routine.  *
 327           *     If the instruction in NIR is not a paired stackop then the   *
 328           *     jump target of the overhead line is incremented by two.      *
 329           *     The second copy of the routine handles this case.  The       *
 330           *     preceeding overhead line is:                                 *
 331           *                                                                  *
 332           *          SM   JSZ  SRPA      ROBS  FFFF SM   ADDL      SM       *
 333           *                                                                  *
 334           ********************************************************************
 335           *
 336    0013   SRPU           JSZ  STNU      UNC        SM   INC       SM   DCSR      Jump; SM := SM + 1, decrement SR
 338           *
 339           %0014
 340           *
 341    0014   SRPA UBA  Q    JSZC SRPU      NEG   SR         ADD       CTR  INSR      JSB if (SM <= Q) for stack underflow
 343                                                                                    (P, SM and SR need to be adjusted);
 344                                                                                    CTR := SR for REGN store, inc SR
 345    0015                 ADD            NEXT       OPB  ADD       REGN STFF      NEXT; REGN := TOS in memory, STFF for BCCnm
 347    0016        UBA  Q   JSZC SRPU      NEG   SR         ADD       CTR  INSR      JSB if (SM <= Q) for stack underflow
 349                                                                                    (P, SM and SR need to be adjusted)
 350                                                                                    CTR := SR for REGN store, inc SR
 351    0017                 ADD            NEXT       OPB  ADD       REGN STFF      NEXT; REGN := TOS in memory, STFF for BCCnm
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 354            ***************************************************************
 355            *    PSHM pushes the bottom of the stack in the TOS registers   *
 356            *    onto the top of the stack in memory.  PSHM checks for stack*
 357            *    overflow and should therefore be called before changing    *
 358            *    the stack or any software known registers.                 *
 359            ***************************************************************
 360            *
 361     0018   PSHM       SM   INC            WRS         SM   INC       SM   DCSR F5B   Write at (SM + 1); SM := SM + 1, decrement
 362                                                                                        SR, skip STOV check if from #IMS
 363     0019              QDWN ADD            DATA  Z     SREG JSZC STO1           NEG   Write QDWN; STOV if not (Z >= new (SM))
 364     001A              RH   ADD                  SP3B      ADD                  RSB   UBA := RH; UBB := SP3B, RSB
 365            *
 366            ***************************************************************
 367            *    PSM2 pushes the bottom two TOS registers onto the top of the*
 368            *    stack in memory.  PSHM checks for stack overflow and should*
 369            *    therefore be called before changing the stack or any software*
 370            *    known registers.                                           *
 371            ***************************************************************
 372            *
 373     001B   PSM2       SM   INC            WRS               ADD           DCSR       Write at (SM + 1); Decrement SR
 374     001C              QDWN ADD            DATA        UBA   INC       SM   DCSR       Write QDWN; SM := SM + 2, decrement SR again
 375     001D              QDWN ADD            DATA  Z     SREG JSZC STO1           NEG   Write QDWN; STOV if not (Z >= new (SM))
 376     001E              RH   ADD                  SP3B      ADD                  RSB   UBA := RH; UBB := SP3B, RSB
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 385            ********************************************************************
 386            *                                                                  *
 387            *    Interrupt Handler                                             *
 388            *                                                                  *
 389            *    Entered if a hardware interrupt was pending at the end of the *
 390            *    preceding clock or an overflow interrupt is pending at the    *
 391            *    beginning of the current clock.  The following interrupts are *
 392            *    processed by their bit settings in CPX1 and CPX2:             *
 393            *                                                                  *
 394            *    CPX1 bit set              Interrupt pending                   *
 395            *                                                                  *
 396            *    R    1            -      Integer Overflow                    *
 397            *    R    2            -      Bounds Violation                    *
 398            *    R H  4            -      Run/Halt Switch                     *
 399            *    R    6            -      TCLK Interrupt                      *
 400            *    R H  8            -      DCU Command Interrupt               *
 401            *    R    9            -      Message Interrupt                   *
 402            *    R   11            -      Memory Breakpoint Interrupt         *
 403            *    R H 12            -      Power Fail Interrupt                *
 404            *                                                                  *
 405            *    CPX2 bit set              Condition existing                  *
 406            *                                                                  *
 407            *         6            -      RUN FF                              *
 408            *                                                                  *
 409            *    All interrupts are processed thru #INT8 (non-ICS) or #INT9    *
 410            *    (ICS), with external program label in SP1B and parameter in   *
 411            *    SPOA, with the exception of Run/Halt Switch, DCU Command      *
 412            *    Interrupt and Power Fail Warning.                             *
 413            *                                                                  *
 414            ********************************************************************
 415            *
 416  001F  IRDN 0500      ADDL                      FFFF P    ADDL      P              UBA := mask for FAKENEXT; P := P - 1
 418  0020                 JSB  IRH       MEDJ       UBA  ADD                CCPX       Wait one clock so that FAKENEXT does not
 420                                                                                     inhibit STFF; FAKENEXT initializes flags
 421  0021  IRH  0200 CPX2 ANDL                                 ADD       CTR  STFF       UBA := RUNFF; CTR := 0, STFF in case CIR is
 423                                                                                       a BCC
 424  0022           CPX1 CSL       SP4A            UBA  JSL  HMIT            ZERO     SP4A := UBA := CPX1 & csl[1];
 426                                                                                     Jump if a halt mode interrupt
 427  0023      0A00      ADDL      RH         1120 UBA  ANDL                          RH := mask to clear TCLK interrupt;
 429                                                                                       UBB := run mode interrupts
 430  0024           SP4A JSB  IRO       NEG        UBB  JSL  RMIT            NZRO     Jump if run mode interrupt else
 432                                                                                       jump if integer overflow
 433  0025           CPX1 JSB  IMS       MSGI  UBA  UBA  JSB  BNDV            NEG      Else jump if bounds violation else
 435                                                                                       jump if message interrupt
 436  0026  PFWM 0008 CPX1 ANDL      SP4A       0400 SP4A ANDL                ZERO     SP4A := PFWINT; Skip if not TCLK interrupt
 438  0027           UBA  ADD                        JSL  TCLK BKX3            UNC      UBA := PFWINT (for #IMS);
 440                                                                                       BKX3 := 0, jump if TCLK interrupt
 441  0028                JSB  UEXI      UNC        SP4A JSB  PFW             NZRO     Jump if powerfail interrupt else
 443                                                                                       SYSHALT if any other interrupt
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 445            ************************************************************************
 446            *                                                                      *
 447            *           Set SR to Recover Stack Module                             *
 448            *                                                                      *
 449            ************************************************************************
 450            *
 451    0029    STSR          ADD                 SR        REPC          CLSR ZERO  ; UBB := SR, REPC if <> 0, SR := ESR := 0
 453    002A                  ADD                 UBB       CAD           INSR ZERO  ; Increment SR & ESR
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  456           ********************************************************************"
  457           *                                                                  "
  458           *          Non-ICS Type Interrupt entry                            "
  459           *                                                                  "
  460           *          Entered if a non-ICS type hardware or firmware          "
  461           *          interrupt is detected.  This module will:               "
  462           *                                                                  "
  463           *      INT8 ->  1)  Empty TOS registers                            "
  464           *               2)  Write a four word stack marker                 "
  465           *      IR1  ->  3)  Place parameter on top of user's stack         "
  466           *               4)  Set STATUS to privileged mode and code         "
  467           *                   segment# 1 (ININ)                              "
  468           *               5)  If from #BNDV then clear interrupt bit in      "
  469           *                   CPX1                                           "
  470           *               6)  Place CIR into X register                      "
  471           *               7)  Go set up code segment #1 (#PCL3)              "
  472           *                                                                  "
  473           ********************************************************************"
  474           *
  475  002B  INT8       SM   INC            WRS   SR   SM   REPC       SM   DCSR SRZ   YRGA := SM + 1 for write; SM := SM + SR
  477  002C             QDWN ADD            DATA       P    ADD              DCSR SRL2  Empty TOS registers to memory;UBB=P REG
  479  002D       X          ADD            DATA  UBB  PB   SUB              CLSR       Write (X) to stack marker;UBB=P-PB ,SR=0
  481  002E                  ADD                  UBB  XR12 IOR                         ;UBB=P-PB IOR MAPFLAG      (CSTX)
  483  002F       UBB        ADD            DATA            ADD                        Write (delta P) WTH L/P BIT           (CSTX)
  485  0030            STA  ADD       RD   DATA  0004 SM   ADDL       SM              Write (STATUS), RD := STATUS; SM := SM + 4
  487  0031       UBB  Q    SUB            DATA  UBB       ADD        Q    INCN       Write (delta Q); Q := SM, set NAMER
  489  0032  IR1       SM   INC            WRS   8001      ADDL       SP2B            YRGA := SM + 1 for parameter; SP2B := priv
  491                                                                                   mode & segment# 1 for STATUS
  492  0033       SPOA       ADD            DATA       UBA  ADD        SM   CF2  NF1   Write (parameter); SM := SM + 1, clear F2,
  494                                                                                   skip if not from #BNDV thru STSR
  495  0034            SP2B ADD       STA                 XR14 ADD              CCPX    Set STATUS; Clear internal interrupt in CPX1
  497  0035            CIR  ADD       X                        JSL  PCL3               UNC  X := CIR; Go set up segment# 1
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 500            ************************************************************************
 501            *                                                                      *
 502            *          Interrupt Control Stack Set Up Module                       *
 503            *                                                                      *
 504            *          Called when switching from the user's stack to the          *
 505            *          Interrupt Control Stack (ICS).  The ICS set up              *
 506            *          module will:                                                *
 507            *                                                                      *
 508            *               1)  Read QI and set Q                                  *
 509            *               2)  Read ZI and set Z                                  *
 510            *               3)  Set DL to QI                                       *
 511            *               4)  Set the contents of QI-6 to S-relative             *
 512            *                   address (S - user DB)                              *
 513            *               5)  Set ICSFLAG bit in CPX2 to 1                       *
 514            *               6)  Set SM to Q+2 for parameter                        *
 515            *                                                                      *
 516            ************************************************************************
 517            *
 518  0036  ICS  0005      ADDL                          ADD       BKX3
 520  0037           UBA   ADD          ROB3  7000       ADDL      XR11
 522  0038           UBA   INC          ROB3       SM    ADD       RC
 524  0039      FFFC       ADDL                    OPB   ADD       Q
 526  003A      UBA  UBB   ADD          ROX3       OPB   ADD       Z
 528  003B      FFFE UBA   ADDL                    Q     ADD       DL
 530  003C           UBA   ADD          WRX3  0002 Q     ADDL      SM
 532  003D      RC   OPA   SUB          DATA       XR11  ADD             CCPX RSB
```

```
Address absolute 5 for QI; BKX3 := 0
Read @QI; XRB11 := mask to set ICSFLAG
Read @ ZI; RC := @ user's stack
UBA := -4; Q := QI
Read (QI-4); Z := ZI
UBA := QI-6; DL := QI
Set up write at QI-6; S := Q+2
(QI-6) := S - (QI-4); Set ICSFLAG, return
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  535           ********************************************************************
  536           *                                                                  *
  537           *          Firmware Interrupt Entry Points                         *
  538           *                                                                  *
  539           *          Called by the firmware when various conditions          *
  540           *          are detected.  These interrupts include:                *
  541           *                                                                  *
  542           *          Interrupt Entry Point      Description                  *
  543           *                                                                  *
  544           *            TRP                -    unimplemented instruction    *
  545           *            STTV               -    Segment Transfer Table       *
  546           *                                    Violation                     *
  547           *            CSTV               -    Code Segment Table Violation *
  548           *            DSTV               -    Data Segment Table Violation *
  549           *            STUN               -    STack UNderflow              *
  550           *            TRP6               -    privileged mode violation    *
  551           *            STOV               -    STack OVerflow               *
  552           *            UNCL               -    STT entry UNCaLlable         *
  553           *            BNDV               -    BouNDs Violation             *
  554           *            PON                -    Power ON (enters from @1)    *
  555           *            PFW                -    Power Fail Warning           *
  556           *            ABDS               -    ABsent Data Segment          *
  557           *            ABCS               -    ABsent Code Segment          *
  558           *            TRCE               -    TRaCE segment                *
  559           *            NRM                -    Non-Responding Module        *
  560           *                                                                  *
  561           ********************************************************************
  562           *
  563   003E  TRP              ADD                    9001      ADDL      SP1B              ; SP1B := external label (%110001))
  565   003F  PSHL             ADD             CF1               ADD               CRF       Clear F1; Clear RFLAG
  567   0040        SP1B ADD        SPOA               JSB  INT8          UNC   SPOA := external label as parameter; Jump
  569   0041  STTV             ADD                    9101      ADDL      SP1B                    ; SP1B := ext label (%110401) (CSTX)
  571   0042        00FE STA    ANDL                     XR12 ADD               ZERO  SEG 1 AND NOT PHYS MAPPED?              (CSTX)
  573   0043                    JSB  PSHL       UNC      UBA  JSB  STTH        ZERO  SYSHALT if segment# 1 else trap
  575   0044  CSTV             ADD                    9201      ADDL      SP1B                    ; SP1B := ext label (%111001) (CSTX)
  577   0045        00FE STA    ANDL                     XR12 ADD               ZERO  SEG 1 AND NOT PHYS MAPPED?              (CSTX)
  579   0046                    JSB  PSHL       UNC      UBA  JSB  CSTH        ZERO  SYSHALT if segment# 1 else trap
  581   0047  DSTV             JSB  PSHL       UNC    9301      ADDL      SP1B        Trap; SP1B := external label (%111401)
  583   0048  STU1             ADD                      SM   INC       SM    DCSR      STUN entry to inc sm and dec SR
  585   0049  STUN             JSB  PSHL       UNC    9401      ADDL      SP1B        Trap; SP1B := external label (%112001)
  587   004A  TRP6             JSB  PSHL       UNC    9501      ADDL      SP1B        Trap; SP1B := external label (%112401)
  589   004B  STO1 1000 STA    ANDL       SP4A        FFFF P    ADDL      SP1B        SP4A := right stackop bit; SP1B := P - 1
  591   004C        UBA  STA    XOR        STA              UBB  ADD               ROP  Clear right stackop bit; Reread instruction
  593   004D             UBB    ADD             ROP          SP4A JSB  STOV        NZRO  Reread instruction into OPA also;
  595                                                                                     Jump if right stackop bit was set
  596   004E        003F        ADDL                   F000 OPB  ANDL              ZERO  RREG := right stackop mask; Skip if stackop
  598   004F        RREG OPA    AND             ZERO         SP1B JSB  STOV P      UNC   Skip if no right stackop;
  600                                                                                     P := P - 1, jump if not a stackop
  601   0050        1000 STA    IORL       STA              ADD                         Set right stackop bit in STATUS
  603   0051  STO2             ADD                    FFFF P    ADDL      P           ; P := P - 1 to restart instruction
  605   0052  STOV 1000 CPX2   ANDL                   9801      ADDL      SP1B        On ICS?; SP1B := external label (%114001)
  607   0053             UBB    ADD        SPOA CF1         UBA  JSB  STOH        NZRO  SPOA := external label, clear F1;
  609                                                                                     SYSHALT if on ICS
  610   0054                    JSB  INT9       SF2              ADD               CF4B  Jump to ICS interrupt, set F2; Clear F4
  612   0055  UNCL       SP1B ADD        SPOA CF1   A101    ADDL      SP1B              SPOA := parameter (P - LABEL), clear F1;
  614                                                                                       SP1B := external label (%120401)
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
615	0056				ADD						JSB	IR1			UNC	; Jump to set up segment# 1
617	0057	BNDV			ADD			SF1	A000		ADDL		XR14			Set F1 (for #IR1);
619																XRB14 := mask to clear BNDVINT in CPX1
620	0058				ADD				8101		ADDL		SP1B			Jump to undo EPOP's and EPSH's & enter #INT8
622																; SP1B := external label (%100401) (2305)
623	0059			UBB	ADD		SP0A				JSB	STSR			UNC	SP0A=SP1B=PARM;JMP TO UNDO & INT8 (2305)
625	005A	PFW			ADD						ADD			SF4B		; Set F4 (for #L2PF & #IOMS)
627	005B		0C02		ADDL						JSL	PFW0	BKX3		UNC	UBA := Command to go BUSY; BKX3 := 0, jump
629	005C	ABDS			JSB	INT8		CF1	A201		ADDL		SP1B			Trap, clear F1; SP1B := ext label (%121001)
631	005D	ABCS		RD	ADD		STA	CF1	9F01		ADDL		SP1B			STATUS := TSTA, clear F1;
633																SP1B := external label (%117401)
634	005E				JSB	IR1		UNC			JSB	INT8			F2	Absent seg in EXIT; Absent seg in PCAL
636	005F	TRCE		RD	ADD		STA	CF1	A001		ADDL		SP1B			STATUS := TSTA, clear F1
638																SP1B := external label (%120001)
639	0060				JSB	IR1		UNC			JSB	INT8			F2	Trace in EXIT; Trace in PCAL
641	0061	NRM			ADD			CF1		P	INC		P	RONP		Non-responding module (actually non-existing
643																IOA module), clear F1; Increment P & read
644																instr after 2nd word (I/O opcode)
645	0062				ADD				8301		ADDL		SP1B			;SP1B= EXT LABEL=%101401 (2305)
647	0063			UBB	ADD		SP0A				JSB	INT8			UNC	Set up trap to seg# 1, STT# 3(non-ICS)(2305)

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  650           ************************************************************************
  651           *                                                                      *
  652           *          Arithmetic and User Trap Handler                            *
  653           *                                                                      *
  654           *          Called by arithmetic firmware modules when the              *
  655           *          following conditions are detected:                          *
  656           *                                                                      *
  657           *          Interrupt Entry Point          Description                  *
  658           *                                                                      *
  659           *            TRP1                 -      integer overflow             *
  660           *            TRP2                 -      floating point overflow      *
  661           *            TRP3                 -      floating point underflow     *
  662           *            TRP4                 -      integer divide by 0          *
  663           *            TRP5                 -      floating point div by 0      *
  664           *            TRP0                 -      decimal overflow             *
  665           *            TRPA                 -      invalid ascii digit          *
  666           *            TRPD                 -      invalid decimal digit        *
  667           *            TRPS                 -      invalid source count         *
  668           *            TRPR                 -      result count overflow        *
  669           *            TRPZ                 -      decimal divide by 0          *
  670           *                                                                      *
  671           ************************************************************************
  672           *
  673    0064   IRO  F7FF STA  ANDL      STA                 ADD                      Clear overflow bit in status register
  675    0065   TRP1          INC       SPOA                JSB  UTRP           UNC  Integer overflow, param = 1
  677    0066   TRP2 0002     ADDL      SPOA                JSB  UTRP           UNC  Floating point overflow, param = 2
  679    0067   TRP3 0003     ADDL      SPOA                JSB  UTRP           UNC  Floating point underflow, param = 3
  681    0068   TRP4 0004     ADDL      SPOA                JSB  UTRP           UNC  Integer divide by 0, param = 4
  683    0069   TRP5 0005     ADDL      SPOA                JSB  UTRP           UNC  Floating point divide by 0, param = 5
  685    006A   TRP0 000B     ADDL      SPOA                JSB  UTRP           UNC  Decimal overflow, param = %13
  687    006B   TRPA 000C     ADDL      SPOA                JSB  UTRP           UNC  Invalid ascii digit, param = %14
  689    006C   TRPD 000D     ADDL      SPOA                JSB  UTRP           UNC  Invalid decimal digit, param = %15
  691    006D   TRPS 000E     ADDL      SPOA                JSB  UTRP           UNC  Invalid source count, param = %16
  693    006E   TRPR 000F     ADDL      SPOA                JSB  UTRP           UNC  Result word count overflow, param = %17
  695    006F   TRPZ 0010     ADDL      SPOA                JSB  UTRP           UNC  Decimal divide by 0, param = %20
  697    0070   UTRP 2000 STA  ANDL                  9901   ADDL      SP1B           Traps enabled?  SP1B := ext label (%114401)
  699    0071          UBA  JSB  STSR      NZRO             ADD             CF1      JSB to set SR and enter #INT8 if process
  701                                                                                   trap enabled; Clear F1
  702    0072                ADD                           ADD             SOV NEXT If traps not enabled then set overflow bit
  704                                                                                   in STATUS and do NEXT instruction
```

```
RECORD  C S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  706           ************************************************************************
  707           *                                                                      *
  708           *          External Interrupt/Channel Service Request Module           *
  709           *                                                                      *
  710           *          Entered by interrupt handler if message interrupt flag      *
  711           *          in CPX1 is set.                                             *
  712           *                                                                      *
  713           *              1)  Read message in incoming buffer                     *
  714           *              2)  Clear message interrupt flag                        *
  715           *              3)  If CSRQ then set F2                                 *
  716           *              4)  If external interrupt then #INT9 to set up ICS      *
  717           *                  and process                                         *
  718           *              5)  Else if channel service request flag is set         *
  719           *                  process it                                          *
  720           *              6)  System halt (!10) if neither is set                 *
  721           *                                                                      *
  722           ************************************************************************
  723           *
  724   0073    IMS  0C02       ADDL                        UBA  JSZ  PFWM          NZRO  UBA := mask to read FROM code and go BUSY;
  726                                                                                       If PFWINT then ignore MSGI and process PF
  727   0074         0402       ADDL                        UBA  ADD       RH   BUSC      UBA := BUSC command to read upper word;
  729                                                                                       RH := !0C02, read FROM code, go busy
  730   0075         0090       ADDL      SP4A              UBA  ADD            BUSC      SP4A := mask to clear MSGINT;
  732                                                                                       Read upper word
  733   0076                    JSZ  PSHM       SR7         OPB  ADD            SP3B SF5B Empty RG if SR = 7; SP3B := FROM code,
  735                                                                                       set F5 to inhibit STOV check in #PSHM
  736   0077         0038 UBB   ANDL      XR9       00C0  OPB  ANDL           SP1B      XRA9 := module#;
  738                                                                                       SP1B := CSRQ and IRQ bits
  739   0078         UBB  UBB   ADD  SWAB      HBF2        SP4A ADD            CCPX      F2 := CSRQ bit; Clear MSGINT
  741   0079              SP1B  JSZ  NOSX      ZERO              JSL  TCSQ BKX5      F2    If CSRQ then handle it else if neither IRQ
  743                                                                                       nor CSRQ then SYSHALT; BKX5 := 0
  744   007A              STA   ADD  LSL       POS   8000        ADDL       SP3B           Skip if interrupts disabled;
  746                                                                                       SP3B := MASK and IPOLL command
  747   007B                    JSB  INT9      UNC   FFF8        ADDL       SP2B           Set up ICS if IRQ and interrupts enabled;
  749                                                                                       SP2B := UBB := -1 & lsl[3] for module# and
  750                                                                                       mask to set DINTFF
  751   007C         UBB  XR9   ADD            ZERO  SP3B        REPC                     Skip if module# = 1; UBB := !8000, repeat
  753   007D         RREG UBA   ADD            ZERO        UBB   ADD  LSR                 Shift mask in ALUB right until module# is
  755                                                                                       decremented to 0 in ALUA
  756   007E         UBB  XR15  IOR       XR15              SP2B CAD            CCPX      XRA15 := vector of interrupting modules;
  758                                                                                       Set DINTFF (UBB = 7)
  759   007F                    ADD                   1000        ADDL       SP3B           ; SP3B := write to IMBI REGO command
  761   0080         D000       ADDL      RG                  JSL  IOMS           UNC   Re-enable CSRQ2, leave IRQ disabled
  763   0081         1000 STA   ANDL                      RH   ADD  LLZ       BUSC      UBA := right stackop mask; Go unBUSY
  765   0082  RSRT        P     ADD            RONP        UBA  ADD                ZERO  Read next instruction;
  767                                                                                       Skip if not right stackop
  768   0083                    JSZ  NOP       UNC         P    ADD            RNSP      NEXT in two clocks; Read right stackop
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  771           ************************************************************************
  772           *                                                                      *
  773           *          Deferred Interrupt Handler                                  *
  774           *                                                                      *
  775           *          Entered whenever interrupts are turned ON and DINTFF,       *
  776           *          (deferred interrupts flipflop,) is SET.  SINCBIT = 1        *
  777           *          if it was a deferred external interrupt.                    *
  778           *                                                                      *
  779           *               1)  If SINC and TON = 1 then #CSNC                     *
  780           *               2)  XRA9 gets module number                            *
  781           *               3)  Send message to clear IRQ on module, thereby       *
  782           *                   enabling interrupt to be retransmitted             *
  783           *               4)  Repeat until vector after clearing                 *
  784           *                   equals zero                                        *
  785           *               5)  Clear DINTFF iff SINC = 0                          *
  786           *                                                                      *
  787           ************************************************************************
  788           *
  789     0084  DIMS 0C02       ADDL      RH          000F      ADDL      SP2B            RH:=command to read upper words;       (0709)
  791                                                                                       SP2B := mask to clear DINTFF
  792     0085             XR15 ADD  LLZ       POS               ADD                     Mask upper byte,skip if no sincbit     (0709)
  794     0086             XR15 JSB  CSNC      BIT8  UBA   UBA   ADD             CTF1    Service clock interrupt if SINC and TON = 1;
  796                                                                                       modules, set F1 if SINCBIT is set
  797     0087                  ADD                         RH    ADD             BUSC    GO BUSY                                (0709)
  799     0088             XR15 ADD  LLZ                          ADD                                                          (0709)
  801     0089       UBA   UBA  JSB  DIM2      ZERO        OPB   ADD                       Jump if no deferred interrupt          (0709)
  803                                                                                       dont jump if any bits set for interrupt
  804                                                                                       force BUSC to complete
  805     008A       2E5C       ADDL      SP4A                    ADD                     SP4A.(0:8) := REGN code for !4000,     (0709)
  807                                                                                       SP4A.(8:8) := mask to enable IRQ & CSRQ;
  808     008B                  JSZ  PSHM      SR7         UBA   ADD  LRZ  CTR  SF5B      Empty RG if SR = 7;
  810                                                                                       CTR := REGN code for !4000 mask, set F5 to
  811                                                                                       inhibit STOV check in #PSHM
  812     008C             XR15 ADD        SP0A CF2                 XFRR      BKX5 DCTR    Save XRA15 in SP0A, clear F2; BKX5= 0,(0709)
  814                                                                                       dec CTR;
  815     008D       0008       ADDL      XR9         UBA   REGN RPCA      SP2B DCTR NZRO  XRA9 := module# 1; SP2B := bit 1 of inter-
  817                                                                                       rupting mod vector, repeat if 0, dec CTR
  818     008E       0008 UBA   ADDL      XR9         RREG  REGN AND       SP2B DCTR NZRO  Increment module # until bit n of inter-
  820                                                                                       rupting module vector is set
  821     008F                  JSZ  NEXT      MSGI  1000        ADDL      SP3B            Jump to NEXT if MSGINT happened before going
  823                                                                                       BUSY; SP3B := write to IMBI REGO command
  824     0090             SP4A ADD  RLZ       RG                      JSL  IOMS      UNC  RG := mask to enable IRQ and CSRQ;
  826                                                                                       Send command
  827     0091       SP0A  SP2B XOR       XR15                    JSB  DIMS      ZERO  Turn off bit for module to be serviced;
  829                                                                                       Check for another deferred interrupt
  830     0092  DIM2           JSZ  NEXT      MSGI              ADD                                                          (0709)
  832     0093                 ADD                         RH    ADD  LLZ       BUSC F1  ; Go UNBUSY; skip if SINCBIT = 1        (0709)
  834     0094                 JSZ  NOP       ZERO        SP2B  ADD              CCPX    Kill some time for INT; clear DINTFF
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  837           ********************************************************************
  838           *                                                                  *
  839           *         Deferred Clock Interrupts                                *
  840           *                                                                  *
  841           *         Entered from #DIMS if SINC and TON = 1                   *
  842           *                                                                  *
  843           *         on entry:  DINTFF  = 1                                   *
  844           *                    interrupts ON                                 *
  845           *                    SINCBIT = 1                                   *
  846           *                    TON     = 1                                   *
  847           *                                                                  *
  848           *         on exit:   DINTFF  := 0  ( iff DINT vectors = 0 )        *
  849           *                    SINCBIT := 0                                  *
  850           *                    TON     := 0                                  *
  851           *                                                                  *
  852           *         begin                                                    *
  853           *         SINC := TON := 0;                                        *
  854           *         if DIV'S = 0 then DINTFF := 0;                           *
  855           *         if CLKSTAT <> 0 then     << INT caused by SINC instr >>  *
  856           *            PARM := 0             << PARM is zero for SINC int >> *
  857           *         else                     << INT caused by CLK rollover >>*
  858           *            begin                                                 *
  859           *            PARM := (TR);         << PARM is time since last int >>*
  860           *            (TR) := 0;            << time since last INT := 0 >>  *
  861           *            end;                                                  *
  862           *         PLABEL := %106001;       << seg 1, STT %14 >>            *
  863           *         INT9;                    << set up ICS and PCAL plabel >>*
  864           *         end.                                                     *
  865           *                                                                  *
  866           ********************************************************************
  867           *
  868    0095   CSNC      XR15 ADD  LSL             7F7F      ADDL      RH              UBA := XRA15 & lsl(1);
  870                                                                                      Mask to clear SINC & TON bits
  871    0096        000F      ADDL                F000 UBA  ANDL                NZRO  Mask to clear DINTFF; Skip if DIV's <> 0
  873    0097        RH   XR15 AND       XR15      UBA       ADD           CCPX      Clear SINC & TON; Clear DINTFF if DIV's = 0
  875    0098        0016      ADDL                          ADD       BKX3            Addr of STAT for SINC instr; BKX3 := 0
  877    0099        0013      ADDL                UBA       ADD            ROA3       Addr of TR; Read (STAT)
  879    009A                  ADD       SPOA CF1  UBA       ADD            ROA3       Clear parameter, clear F1; Read (TR)
  881    009B             OPA  JSB  CSN2      NZRO            ADD                      Jump if (STAT) <> 0
  883    009C             OPA  ADD            SPOA            ADD            DATA      SPOA(param) := (TR), (TR) := 0 if (STAT) = 0
  885    009D   CSN2           JSZ  INT9            SF2  8C01  ADDL      SP1B        Service interrupt, set F2;
  887                                                                                    SP1B := external label (%106001)
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

  889           ********************************************************************
  890           *                                                                  *
  891           *    ICS-type Interrupt Entry                                      *
  892           *                                                                  *
  893           *    Entered if an ICS-type hardware or firmware interrupt is      *
  894           *    detected.  #INT9 will:                                        *
  895           *                                                                  *
  896           *    1)  Empty TOS registers                                       *
  897           *    2)  Write a four word stack marker                            *
  898           *    3)  Write stack DB-bank and DB-address to user's stack        *
  899           *    4)  Set S-bank to zero                                        *
  900           *    5)  If not on ICS then set up ICS                             *
  901           *    6)  Process external interrupt if F2 is not set               *
  902           *                                                                  *
  903           *    Jump to #IR1 which will:                                      *
  904           *                                                                  *
  905           *    7)  Clear internal hardware interrupt in CPX1                 *
  906           *    8)  Place parameter on ICS                                    *
  907           *    9)  Place CIR in X register                                   *
  908           *   10)  Set up code segment# 1 (ININ)                             *
  909           *                                                                  *
  910           ********************************************************************
  911           *
  912   009E  INT9       SM   INC            WRS   SR   SM   REPC       SM   DCSR SRZ   Set up write @ SM+1; SM := SR + SM
  914   009F             QDWN ADD            DATA       P    ADD             DCSR SRL2  Empty TOS registers to memory;UBB=P REG
  916   00A0       X          ADD            DATA  UBB  PB   SUB             CLSR       Write (X); UBB := P - PB ; CLR SR
  918   00A1                  ADD                  UBB  XR12 IOR                        ;UBB=P-PB IOR MAPFLAG     (CSTX)
  920   00A2       UBB        ADD            DATA  0006 SM   ADDL       SM             Write (delta P); UBB := SM := SM + 6  (CSTX)
  922   00A3            STA   ADD       RD   DATA  0004 SM   ADDL       Q              Write (STATUS), RD := STATUS (for #IXT3),
  924                                                                                    Q := UBB := old SM + 4
  925   00A4       1000       ADDL       SP4A       UBB  Q    XFRR            WRS        SP4A := !1000; SREG := old Q, write @ SM+4
  927   00A5       RREG CPX2  AND             NZRO  RREG SREG SUB        SP2B DATA       Skip if ICSFLAG; SP2B := delta Q, write it
  929   00A6                  JSB   ICS        UNC        BNKD ADD        RH   DATA       Set up ICS if not ICSFLAG; RH := BNKD, write
  931   00A7            CPX2  ADD   SWAB       EVEN       DB   ADD             DATA NF4B  Skip if not in Dispatcher; Write (DB),
  933                                                                                      skip if not starting Dispatcher
  934   00A8                  JSB   IN9D       UNC                 JSL  IXT4 BNKS       UNC   Jump if starting Dispatcher else
  936                                                                                      jsb if in Dispatcher; BNKS := 0
  937   00A9       3FFF       ADDL                                   JSB  IRX  BNKS       NF2   UBA := mask for setting FSS; BNKS := 0,
  939                                                                                      process external interrupts if not F2
  940   00AA            RH    JSB   IR1        NZRO       UBA  INC             CCPX       Jump if BNKD <> 0; Set FSS (UBB = !4000)
  942   00AB                  JSB   IR1        UNC   UBB  SP4A IOR             CCPX       Jump to finish up; Clear FSS (UBB = !5000)
  944   00AC  IN9D 0F00       ADDL                 8000 SP2B IORL                        UBA := mask to clear DISPFLAG;
  946                                                                                      UBB := delta Q with bit 0 set
  947   00AD            UBB   ADD             DATA       UBA  ADD             CCPX RSB   Write delta Q; Clear Dispatcher flag, RSB
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 950            ************************************************************************
 951            *                                                                      *
 952            *          EXTERNAL INTERRUPT HANDLER                                  *
 953            *                                                                      *
 954            *          Entered after saving user's environment on his              *
 955            *          stack and interrupt handler has set up the ICS.             *
 956            *                                                                      *
 957            *          1)  Issue IPOLL command to determine the channel which      *
 958            *              had the interrupt.  Note the system halt if no          *
 959            *              channel responding                                      *
 960            *          2)  Issue OBII command to highest channel responding        *
 961            *              to IPOLL command to determine device which had the      *
 962            *              interrupt.                                              *
 963            *          3)  Set DB bank to zero and DB register from Device         *
 964            *              Reference Table entry 1.                                *
 965            *          4)  Set Status register to privileged mode and enable       *
 966            *              interrupts.                                             *
 967            *          5)  Push Module/Channel/Device number on top of ICS.        *
 968            *          6)  Clear message interrupt and save CIR in X register.     *
 969            *          7)  Get external program label from DRT entry 2.            *
 970            *          8)  Go set up code segment for external interrupt.          *
 971            *                                                                      *
 972            ************************************************************************
 973            *
 974    00AE    IRX  0050      ADDL      SP4A                 JSL  IOMS CTR       UNC   SP4A := code to clear FSS (to be swapped);
 976                                                                                       CTRS.(8:8):=0, send IPOLL command
 977    00AF         RG   RG   JSB  NCHL      ZERO             ADD       BKX3 CCTX      No channel responding; clear CTX, BKX3;
 979                                                                                       Repeat if not channel 0; CTR:=CTR + 1
 980    00B0              RG   ADD            CF2              REPC                      Find the highest channel #
 982    00B1              UBA  ADD  LSL      HBF2      CTRS   ADD            ICTR F2    Shift until highest channel bit in bit 0;
 984                                                                                       UBB := channel number; CTR:=CTR + 1
 985    00B2         UBB  UBB  ADD  LSL            2000       ADDL      SP3B           UBA:=channel# &lsl(2); SP3B:=OBII command
 987    00B3              RG   JSB  CHLO     NEG   UBA        ADD  LSL  BKX5 SF5B      JSB if channel# = 0; BKX5:=channel# &lsl(3)
 989                                                                                       F5:=1 to inhibit DRT0 read in #GDRT
 990    00B4                   ADD                           JSL  IOMS BNKD      UNC   BNKD := 0; send OBII command
 992    00B5                   ADD            SF3A      RG   JSL  GDRT BKX5      UNC   F3 := 1 for #GDRT rtn, BKX5 := CDEV#, goto
 994                                                                                       #GDRT to get DRT0 adr and MCDEV#
 995    00B6              SM   INC            WRD       UBB  ADD                       write adr := SM+1 (BNKD=0); UBB := DRT0 adr
 997    00B7              RH   ADD            DATA      UBB  INC                       UBA := MCDEV#; write MCDEV# as parameter;
 999                                                                                       UBB:=DRT1 adr (#GRDT rtrns w/UBB=DRT0 adr)
1000    00B8         UBB       INC            ROBS      SP4A ADD  SWAB      CCPX      Read external program label (DRT2);
1002                                                                                       UBB := !5000 to clr split bank flag (FSS)
1003    00B9         UBA       CAD            ROBS      1000      ADDL      SP3B      read DBI (DRT1); SP3B := write command
1005    00BA              CIR  ADD       X    SF1       OPB  ADD       SP1B           X := CIR; F1 := 1 to set priv;
1007                                                                                       SP1B := external program label
1008    00BB         0C00      ADDL      RH                     ADD       BNKS CF5B      RH := GO UNBUSY command; BNKS := 0 (for ICS)
1010    00BC         0007 RG   ANDL      RG             0C00 RG IORL      BKX5           RG.(12:1) := clear interrupt,
1012                                                                                       RG.(13:3) := device #
1013                                                                                       BKX5 := register #C for chan # in RG
1014    00BD         2E00      ADDL      SP4A                   OPB  JSL  IOMS DB  UNC   UBA := mask shifted right; DB := DBI,
1016                                                                                       clear interrupt on channel/device
1017    00BE              SP4A ADD  LSL  RG   CF3A                JSL  IOMS BKX5  F1    RG := !5C00 for mask to enable IRQ and CSRQ,
1019                                                                                       clear F3; BKX5  = 0, slow down jump
1020    00BF              SM   INC                              RH   ADD       BUSC      UBA := SM + 1; go unbusy (UBB = !0C00)
1022    00C0                   INC            STA  F1HB         UBA  JSL  PCL3 SM  UNC   STA := %100001 (privileged, segment #1);
1024                                                                                       SM := SM + 1, set up segment #1
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1026            ********************************************************************
1027            *                                                                  *
1028            *          SYSTEM HALT CONDITIONS                                  *
1029            *                                                                  *
1030            *          Error Code     SP1B        Description                  *
1031            *            (octal)     (hex)                                    *
1032            *                                                                  *
1033            *              000000       0000     Unexpected (unknown) interrupt *
1034            *              000001       0001     STT violation in segment #1    *
1035            *              000002       0002     Absent segment on ICS          *
1036            *              000003       0003     Absent or TRACE on segment #1  *
1037            *              000004       0004     Stack overflow on ICS          *
1038            *              000005       0005     CST length = 0                 *
1039            *              000006       0006     Channel program timeout        *
1040            *              000007       0007     Bootstrap checksum error       *
1041            *              000010       0008     Bootstrap channel program abort*
1042            *              000011       0009     PSEB instruction while QI-18 < 0*
1043            *              000012       000A     Module send message timeout    *
1044            *              000013       000B     Incorrect module responding    *
1045            *              000014       000C     Channel not system controller  *
1046            *              000015       000D     Code segment violation in seg #1*
1047            *              000016       000E     Non-responding channel         *
1048            *              000017       000F     Channel zero responding        *
1049            *              000020       0010     No service request or external *
1050            *                                       interrupt on message interrupt*
1051            *              000021       0011     Channel controller cannot be   *
1052            *                                       made controller-in-charge  *
1053            *              000022       0012     Module receive message timeout *
1054            *              000023       0013     I/O error, parity / timeout    *
1055            *              000024       0014     WCS checksum error             *
1056            *              000025       0015     LUT checksum error             *
1057            *              000026       0016     Bad DCU command code           *
1058            *                                                                  *
1059            ********************************************************************
1060            *
1061  00C1  UEXI          JSB  SHLT      UNC            ADD       SP1B      Unexpected (unknown) interrupt
1063  00C2  STTH          JSB  SHLT      UNC            INC       SP1B      STT violation in segment #1
1065  00C3  ICSH          JSB  SHLT      UNC            INC   LSL SP1B      Absent code segment while on ICS
1067  00C4  ABTH          JSB  SHLT      UNC      0003  ADDL      SP1B      Absent segment or trace in segment# 1
1069  00C5  STOH          JSB  SHLT      UNC      0004  ADDL      SP1B      Stack overflow on ICS
1071  00C6  CTLV          JSB  SHLT      UNC      0005  ADDL      SP1B      CST length violation
1073  00C7  CPTO          JSB  SHLT      UNC      0006  ADDL      SP1B      Channel program timeout
1075  00C8  CPCS          JSB  SHLT      UNC      0007  ADDL      SP1B      Bootstrap channel program checksum
1077  00C9  CPAB          JSB  SHLT      UNC      0008  ADDL      SP1B      Bootstrap channel program abort
1079  00CA  PSEH          JSB  SHLT      UNC      0009  ADDL      SP1B      Pseudo-Enable violation (QI-18) < 0
1081  00CB  MSTO          JSB  SHLT      UNC      000A  ADDL      SP1B      Module send message timeout
1083  00CC  INVM          JSB  SHLT      UNC      000B  ADDL      SP1B      Incorrect module responding
1085  00CD  NSYC          JSB  SHLT      UNC      000C  ADDL      SP1B      Channel not system controller
1087  00CE  CSTH          JSB  SHLT      UNC      000D  ADDL      SP1B      Code segment violation in segment #1
1089  00CF  NCHL          JSB  SHLT      UNC      000E  ADDL      SP1B      No channel responding
1091  00D0  CHL0          JSB  SHLT      UNC      000F  ADDL      SP1B      Channel 0 responding
1093  00D1  NOSX          JSB  SHLT      UNC      0010  ADDL      SP1B      Message interrupt w/o IRQ or CSRQ
1095  00D2  NCON          JSB  SHLT      UNC      0011  ADDL      SP1B      Not able to put it to controller-in-charge
1097  00D3  MRTO          JSB  SHLT      UNC      0012  ADDL      SP1B      Module receive message timeout
1099  00D4  IOEE          JSB  SHLT      UNC      0013  ADDL      SP1B      I/O error, parity / timeout
1101  00D5  WCCE          JSB  SHLT      UNC      0014  ADDL      SP1B      WCS checksum error
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
1103	00D6	LTCE			JSB	SHLT		UNC	0015		ADDL		SP18			LUT checksum error
1105	00D7	BDCC			JSB	SHLT		UNC	0016		ADDL		SP18			Bad DCU command code
1107	00D8	SHLT			ADD				0060		ADDL					; UBB := mask to set SYSHALT bit in CPX2
1109	00D9				ADD				UBB		ADD			CCPX		; Set system halt bit in CPX2
1111	00DA			XR9	DIAG						ADD					DIAG for DCU to take control. XRA9 for I/O error analysis
1113																

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1115            ************************************************************************
1116            *                                                                      *
1117            *    Utility Subroutines                                               *
1118            *                                                                      *
1119            ************************************************************************
1120            *
1121            ************************************************************************
1122            *    ETO2 - Empties TOS registers until only 2 are filled              *
1123            ************************************************************************
1124            *
1125    00DB  ETO2 FFFD      ADDL                 Z    SM   SUB       SP2B SF2         UBA := -3; SP2B := Z - SM, set F2 to
1127                                                                                     indicate not split banks (for #SCU, #SCW)
1128    00DC           SM    INC            WRS   SR   UBA  REPC      RH   DCSR NCRY  Set up write @ SM + 1;
1130                                                                                     RH := SR - 3, repeat if SR >= 3, dec SR
1131    00DD           QDWN  ADD            DATA  UBB       CAD            DCSR NEG   Write QDWN;
1133                                                                                     End when SR counts down to 2, dec SR
1134    00DE      RH   SP2B  JSZS STO2      POS   RH   SM   INC       SM   INSR RSB   STOV if ((SR - 3) - (Z - SM)) >= 0;
1136                                                                                     SM := (SR - 3) + SM + 1, inc SR, return
1137            ************************************************************************
1138            *    ETO3 - Empties TOS registers until only 3 are filled              *
1139            ************************************************************************
1140            *
1141    00DF  ETO3 FFFC      ADDL                 Z    SM   SUB       SP2B SF2         UBA := -4;
1143                                                                                     SP2B := Z - SM, set F2 (for #MVBL)
1144    00E0           SM    INC            WRS   SR   UBA  REPC      RH   DCSR NCRY  Set up write @ SM + 1;
1146                                                                                     RH := SR - 4, repeat if SR >= 4, dec SR
1147    00E1           QDWN  ADD            DATA  UBB       CAD            DCSR NEG   Write QDWN;
1149                                                                                     End when SR counts down to 3, dec SR
1150    00E2      RH   SP2B  JSZS STO2      POS   RH   SM   INC       SM   INSR RSB   STOV if ((SR - 4) - (Z - SM)) >= 0;
1152                                                                                     SM := (SR - 4) + SM + 1, inc SR, return
1153            ************************************************************************
1154            *    PSHA - Empties all TOS registers                                  *
1155            ************************************************************************
1156            *
1157    00E3  PSHA     SM    INC            WRS   SR   SM   REPC      SM   DCSR SRZ   Set up write to SM + 1; SM := SR + SM
1159    00E4           QDWN  ADD            DATA       ADD            DCSR SRL2  Empty TOS registers; Dec SR
1161    00E5                 ADD            RSB        ADD            CLSR       Return; Clear SR
1163            *
1164            ************************************************************************
1165            *    MWOL - Bounds checking for 'move while' instructions              *
1166            ************************************************************************
1167            *
1168    00E6  MWOL     RA    INC       RA                 JSL  MBWX          NWIL  RA= DESTINATION POINTER;              (2311)
1170                                                                                     JMP IF NOT WHILE CONDITION (DONE)
1171    00E7                 ADD                          JSZ  BNDV          UNC   ; JUMP IF REALLY A TRAP CONDITION... (2311)
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1174            ********************************************************************
1175            *          TIME - Measurement microcode routine                    *
1176            *                                                                  *
1177            *                1.  delta t := countsave - tclk                   *
1178            *                2.  countsave := tclk                             *
1179            *                3.  desired accumulator := desired accumulator    *
1180            *                                         + delta t                *
1181            ********************************************************************
1182            *
1183    00E8    TIME      UBA  ADD       SP4A       0030       ADDL       CTR            Save right stackop bit in SP4A for RSRT;
1185                                                                                       CTR := TCLK REGN option
1186    00E9                   ADD                   00A1       ADDL       CTR            ; CTR := XR33 REGN code (countsave)
1188    00EA                   ADD                         REGN ADD        SP3B           ; SP3B := current value of TCLK
1190    00EB                   ADD                   UBB   REGN RSUB       SP3B           ; SP3B := countsave - TCLK (delta t)
1192    00EC                   ADD                   SP3B       ADD        REGN           ; Countsave := current timer value
1194    00ED                   ADD                         SP2B ADD        CTR            ; CTR := REGN code of desired accumulator
1196    00EE                   ADD                              ADD                       Not much to do except wait around for CTR
1198    00EF              REGN ADD       REGN       SP3B REGN LINK       REGN ICTR      Add delta t to proper accumulator, inc CTR
1200                                                                                         to next REGN code
1201    00F0                   ADD            RSB   E000       ADDL                      Return; UBB := mask to clear PAUSEFF
1203            *
1204            ********************************************************************
1205            *          IOER - Error routine used by I/O reads & writes         *
1206            ********************************************************************
1207            *
1208    00F1    IOER           ADD            SF2   FF00 BKX4 ANDL                NZRO  Set F2; Check if reading FIFO
1210    00F2                   JSZ  IOEE      UNC        RG   JSL  SUSB SP3B      ODD   I/O error; suspend if DNV when reading FIFO
1212            *
1213            ********************************************************************
1214            *          TSRT - Measurement microcode entry to #RSRT from #CPEX  *
1215            ********************************************************************
1216            *
1217    00F3    TSRT      UBA  JSZ  TIME       UNC   00A2       ADDL       SP2B           Add channel program time, UBA := right
1219                                                                                         stackop bit (from #CPEX)
1220    00F4              SP4A JSZ  RSRT       UNC              ADD                       UBA := right stackop bit, restart instr
1222            *
1223            ********************************************************************
1224            *     DTS1 - Sets condition code and status carry bit for #DTST    *
1225            *            and #MPYL instructions                                *
1226            ********************************************************************
1227            *
1228    00F5    DTS1      UBA  ADD            NZRO       RB   ADD            CCA            Skip if ms 17 bits are not the same;
1230                                                                                         CCA on msw
1231    00F6                   ADD            NEXT       RA   ADD            DCC            NEXT if ms 17 bits are the same; DCC on lsw
1233    00F7                   ADD            NEXT            ADD            SCRY           NEXT; Set carry bit in status
1235            *
1236            ********************************************************************
1237            *    DMPX - Path to DUMP routines                                  *
1238            ********************************************************************
1239            *
1240    00F8    DMPX           ADD                                JSL  WMST RF       UNC   RF := 0 to start with IOA1, jump
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1243          ************************************************************************
1244          *                                                                      *
1245          *    STNU - SR Preadjust Stack Underflow Pointer                       *
1246          *                                                                      *
1247          ************************************************************************
1248          *
1249    00F9  STNU 2000 CPX1 ANDL                       ADD                          CHECK TO SEE IF BNDV HAS OCCURRED  (2428A)
1251    00FA                 ADD                  UBA   JSL  STNX           ZERO    IF NOT THEN CONTINUE STOV PROCESS  (2428A)
1253    00FB                 JSZ  BNDV      UNC   Q     ADD       SM   CLSR       ELSE PROCESS BNDV;RESTORE SM; SR=0 (2428A)
1255    00FC                 ADD                        ADD                          NOP to keep system happy             (2555)
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1260            %0100
1261            *
1262            ************************************************************************
1263            *                                                                      *
1264            *          PCAL INSTRUCTION                                            *
1265            *                                                                      *
1266            *          This module will perform the internal label processing      *
1267            *          of the PCAL instruction:                                    *
1268            *                                                                      *
1269            *              DSPL = 0       -    program label on TOS                *
1270            *                   > 0       -    program label in STT                *
1271            *                                                                      *
1272            *              1)  if DSPL = 0 then label on TOS else label is         *
1273            *                  in Segment Transfer Table (DSPL = STT #).           *
1274            *              2)  empty TOS registers.                                *
1275            *              3)  if SM + 4 > Z then STACK OVERFLOW.                  *
1276            *              4)  write out four word stack marker.                   *
1277            *              5)  if DSPL > total # of labels in STT then STT         *
1278            *                  VIOLATION.                                          *
1279            *              6)  if PCAL 0 and segment number on TOS = 0 then        *
1280            *                  CST VIOLATION.                                      *
1281            *              7)  if program label is external then set up            *
1282            *                  code segment else set up address.                   *
1283            *              8)  if address outside PB - PL then BOUNDS              *
1284            *                  VIOLATION.                                          *
1285            *                                                                      *
1286            ************************************************************************
1287            *
1288            $LUT INSTR=PCAL:0 011 001 0xx xxx xxx, DSPL=8, ENTRY=PCAL
1289            *
1290    0100    PCAL       DSPL ADD        SP4A CF1   PL        ADD          ROP        F1=EXTERNAL; PL=STTL ENTRY FOR INT/EXT(CSTX)
1292    0101    PCL1       UBA  JSB  PCL0       ZERO  PL   PB  SUB      SP2B TICB      TEST N=0;SP2=PL-PB;CTR=OF(OPB);N=0 IS EXTERN
1294                                                                                   (CSTX)
1295    0102               SM   INC            WRS         OPB  ADD  LRZ  RH             POINT TO TOS;UBB=RH=#INTERNALS          (CSTX)
1297    0103         UBB   SP4A SUB            CTF1        OPB  ADD  RRZ  SP3B           F1=INTRNL IF N>#INTERNALS;SP3B=TOTAL #LABELS
1299                                                                                   (2635)
1300    0104               ADD                       PL   SP4A SUB          ROP        NOP;READ PL-N INTO OPB                  (2635)
1302    0105    PTOS       P    ADD        SP0A      SR   SM   REPC     SM   DCSR SRZ  SP0A=OLD P;START EMPTY TOS REGS;SKP IF DONE
1304
1305    0106               QDWN ADD             DATA        UBB  ADD            DCSR SRL2 EMPTY TOS REGS;UBB=S;SR=SR-1;SKIP DONE
1307    0107    PSTK X             ADD             DATA  0004 UBB  ADDL     RF         PUT X IN STACK MARKER; RF=SM+4
1309            SP0A PB    SUB                       XR12 ADD                          UBA:=P-PB ;UBB:=MAPFLAG                 (CSTX)
1311    0109         UBA   UBB  IOR             DATA        REGN ADD      SP1B       PUT MAP BIT IN P-PB  ;SP1B=LABEL        (CSTX)
1313    010A               STA  ADD        RD   DATA  Z   RF   JSBS PCSO          NEG  STK MRKR=RD=STATUS;O'FLO IF NEW SM>Z
1315    010B         RF    Q    SUB             DATA  3FFF SP1B ANDL     RG         WRT DELTA Q TO STK;RG=ADDR PART OF LBL
1317    010C         UBB   PB   ADD        SP4A RONP        RF   ADD      SM   CLSR     SP4A=UBA=PB+DELTA P (NEW P);SM=SM+4
1319    010D               SP1B ADD  RRZ       F1    SP3B SP4A SUB                CRRY UBA=LABEL.(8:8) (SEG #),SKIP IF LOCAL LABEL;
1321                                                                                   SKIP IF TOTAL #LABELS >= DISP           (2635)
1322    010E         UBA   DSPL JSZ  CSTV      ZERO              JSZ  STTV          UNC  CSTV IF PCAL 0 AND LABEL.(8:8)=0;STTV (2635)
1324    010F         RG    SP2B JSZC BNDV      CRRY        RF   JSB  PCL3 Q        NF1  BND VIO IF ADR>PL-PB;Q=SM+4;JSB IF EXT(CSTX)
1326    0110               JSZ  NEXT           UNC         SP4A ADD      P            INTERNAL. NEXT INSTRUCTION;P=NEW P      (CSTX)
1328    0111    PCSO       SP4A JSZ  STO2      NZRO              ADD          CLSR       JSB IF NOT PCAL 0; CLSR                 (CSTX)
1330    0112               JSZ  STO2           UNC         REGN ADD      RA   INSR     JSB FOR STACK OVERFLOW; RESTORE LABEL, SR =1
1332            *
1333            ************************************************************************
1334            *          PCAL INSTRUCTION - LABEL ON TOP OF STACK                    *
```

```
RECORD  C S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL  RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1335            ********************************************************************
1336            *
1337    0113    PCLO        P    ADD        SPOA SRZ         SM   ADD        RH   POP  SRZ   SPOA:=P, SKIP IF SR=0;
1339                                                                                          RH:=SM (RG:=SM AFTER POP), POP, SKIP IF SR=0
1340    0114                SM   JSBI PTOS       WRS   0007       ADDL   .   CTR              JSB, WRITE TOS REGISTERS AT SM + 1;
1342                                                                                          CTR:=REGN CODE FOR RH (LABEL NOW IN RH)
1343    0115        RG      Q    JSZC STUN       NEG         SM   ADD                 CLSR   STUN IF SM <= Q; UBB:=SM, SR:=0
1345    0116                SM   JSB  PSTK       ROBS  UBB         CAD        SM            JUMP BACK, READ LABEL INTO OPB; SM:=SM - 1
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1348            $NOWARN
1349            ********************************************************************
1350            *                                                                  *
1351            *          CODE SEGMENT SET UP MODULE                              *
1352            *                                                                  *
1353            *          This module is called by PCAL and the interrupt         *
1354            *          handlers to set up execution on a particular code       *
1355            *          segment with:                                           *
1356            *                                                                  *
1357            *               SP1B = external program label                      *
1358            *               RH= number of internals                            *
1359            *               DSPL and STA set                                   *
1360            *                                                                  *
1361            *               1)  Read LSTT bank, addr pointer from 1221,1222 (8)  *
1362            *               2)  Read #LSTT ents (first word of LSTT)           *
1363            *               3)  If logically mapped, then                      *
1364            *                   a) Read LSTT number of entries (first word)    *
1365            *                   b) STT violation if LSTT bank=LSTT addr =0     *
1366            *                   c) STT violation if seg # > # LSTT entries     *
1367            *                   d) If segment # (from label) = 0 then          *
1368            *                      1) Index into LSTT with N-#internals        *
1369            *                      2) Read new label                           *
1370            *               5)  If LABEL(8:8)=0 then CST violation             *
1371            *               6)  Phys Seg (RH)=LABEL(8:8)                       *
1372            *               7)  If logically mapped, then                      *
1373            *                   a) If LABEL(8:8)>#LSTT entries then STT violation*
1374            *                   b) Index into LSTT with LABEL(8:8)*2 get phys add*
1375            *               8)  If logically map and LABLE(8:8)<=loc1223(8) then *
1376            *                      use CSTX(1) for CST base                    *
1377            *                   Else use CSTB(0) for CST base                  *
1378            *               9)  set PB BANK, PB and PL from CST entry           *
1379            *              10)  set reference bit in CST entry #1              *
1380            *              11)  STATUS register is set to new segment #        *
1381            *              12)  if TRACE/ABSENT flag is set then               *
1382            *                   TRACE/ABSENT trap                              *
1383            *              13)  if STT# < STTL then STT VIOLATION trap         *
1384            *              14)  if external label in STT then STT VIOLATION TRAP *
1385            *              15)  if STT# = 0 then                               *
1386            *                      if called from priviledged then             *
1387            *                         P:= PB                                   *
1388            *                      else                                        *
1389            *                         STT UNCALLABLE                           *
1390            *                   else                                           *
1391            *                      if called from priviledged then             *
1392            *                         P:= PB + label address                   *
1393            *                      else                                        *
1394            *                         if called segment is priviledged then    *
1395            *                            STT UNCALLABLE;                       *
1396            *              16)  if PB <= P <= PL is not true then BOUNDS       *
1397            *                   VIOLATION                                      *
1398            *              17)  next instruction, start execution of code      *
1399            *                   segment                                        *
1400            *                                                                  *
1401            ********************************************************************
1402            *
1403    0117    PCL3 0292      ADDL      RE                    ADD       BKX3 SF2       RE=LSTT ADDR;BKX3=0,SET F2 FOR MED JMP(CSTX)
```

RECORD NO	C S ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
1405	0118		RH	DSPL	SUB		SPOA			SP1B	ADD	RRZ	RH			SPOA=-(N-#INTS);RH=LABEL(8:8) (SEG #) (CSTX)
1407	0119			RE	INC			ROX3	RE		CAD			ROX3		READ #ENTRIES (1223),READ LSTT BANK (CSTX)
1409	011A			STA	ADD	RRZ	RG			SP1B	JSB	PCLP			NEG	RG=STA(8:8) ;JMP IF PHYSICAL MAPPED (CSTX)
1411	011B		RH		ADD		RE	ZERO		RE	ADD			ROX3		RE=RH,SKP IF SEG=0 ;READ LSTT ADDR (CSTX)
1413	011C		RH	OPA	JSBC	PCLX		NCRY		OPB	ADD		BKX4			JMP IF SEG<=#SEGS(1223); SAVE LSTT BNK(CSTX)
1415	011D		RG	RG	INC		SP4A			OPB	ADD		RD		NZRO	UBA=STA(8:8)*2+1 ;UBB=LSTT BASE,SKP IF NZRO
1417																(CSTX)
1418	011E			UBB	ADD			ROX4		BKX4	JSZ	CSTV			ZERO	READ #LSTT ENTS; CSTV IF LSTT BANK=ADDR=0
1420																(CSTX)
1421	011F		RD	SP4A	ADD			SF1		RH	JSB	PCL6			NZRO	UBA=LSTTB+STA(8:8)*2+1;JMP IF NZRO LOG MP
1423																(CSTX)
1424	0120			OPA	ADD	RRZ				UBA	ADD			ROX4		UBA= #LSTT ENTS ;READ PTR TO LABEL (CSTX)
1426	0121		SPOA	RD	ADD				RH	UBA	JSZC	CSTV			CRRY	UBA=LSTTB-(N-#INTS);CSTV IF SEG>#LSTT ENTS
1428																(CSTX)
1429	0122				ADD				UBA	OPB	ADD			ROX4		;READ LABEL (CSTX)
1431	0123				ADD						ADD			CF1		;INDIC SEG # =0 (CSTX)
1433	0124				ADD					OPB	ADD	RRZ	RH			;RH=NEW LABEL(8:8) (SEG #) (CSTX)
1435	0125			UBB	JSZ	CSTV		ZERO		OPB	JSB	PCLP	SP1B		NEG	CSTV IF NEW SEG=0 ;SP1B=NEW SEG,JP IF PHY LB
1437																(CSTX)
1438	0126	PCL6	RH	OPA	JSZC	CSTV		CRRY	RH	RH	ADD					CSTV IF SEG>#ENTS;UBB=LABEL(8:8)*2 (CSTX)
1440																(CX2328A)
1441	0127				ADD			F1	RD	UBB	ADD			ROX4		SKIP IF F1 SET;READ LSTT ENT (LSTTB+LABEL*2)
1443																(CSTX)
1444	0128			RH	ADD		RE				ADD					IF SEG# WAS 0, PUT IN SEG # FROM LSTT (CSTX)
1446	0129				JSB	PCLP		UNC		OPB	ADD		RH		UNC	GO READ CSTB(0);RH=NEW 16 BIT PHY SEG (CSTX)
1448	012A	PCLX			INC			ROX3			JSB	PCL7			UNC	READ CSTX(1); JMP AROUND (CSTX)
1450	012B	PCLP			ADD			ROX3		RH	ADD		RE			READ CSTB(0) FOR PHY MAPPED;RE=RH FOR PHYS
1452																(CSTX)
1453	012C	PCL7		RH	JSZ	CSTV		ZERO	8000	SP1B	ANDL					CSTV IF SEG=0; UBB=MAPPING (LOG/PHY) (CSTX)
1455	012D			UBB	ADD	LSR	RC			UBB	ADD	LSR	XR1			RC=NEWMAPFLG ;XR812:=NEWMAPFLAG (2318)
1457	012E			STA	ADD	LLZ	SP4A		7F00	SP1B	ANDL					SP4A.(0:8):=STA.(0:8); UBB:=STT# (CSTX)
1459	012F			OPA	ADD		SPOA	ROX3		UBB	ADD	LRZ	SP2B	CF4B	ZERO	READ CSTL; SP2B:=STT#, SKP IF= 0, SF4B(CSTX)
1461	0130		RE	SP4A	IOR		RD		RH	RH	INC	LSL		SF4B		RD:=TSTA; (CSTX)
1463																UBB:=SEG# * 4 + 2, SF4B IF STT# <> 0
1464	0131		SPOA	UBB	ADD		SP4A	ROB3	RH	RH	ADD	LSL			NZRO	SP4A:=(CSTB/CSTX) + SEG# * 4 + 2, READ BANK;
1466																UBB:=SEG# * 4, SKIP IF SEG# <> 0
1467	0132		SPOA	UBB	ADD			ROX3			JSB	PCL8			UNC	READ AMRT-LENGTH/4 ((CSTB/CSTX) + SEG# * 4);
1469																CST VIOLATION IF SEG# = 0
1470	0133		RH	OPA	JSBC	PCL8		CRRY	8004		ADDL		SP3B			CST OR CTL VIOLATION IF SEG# > CSTL;
1472																SP3B:=MASK FOR ABSCENCE AND TRACE BITS
1473	0134			OPA	CSL				00FF	OPB	ANDL		BNKP			UBA:=AMRT-LENGTH/4 & CSL(1); BNKP:=NEW BNKP
1475	0135		UBA	UBA	ADD	LSL		CTF1		SP4A	INC			ROX3		SP4A:=UBA:=LENGTH & LSL(1), (M AND R BITS
1477																SHIFTED OUT, A AND T BITS WILL BE ZERO),
1478																F1:=MODE BIT;
1479																READ PB ((CSTB/CSTX) + SEG# * 4 + 3)
1480	0136		UBA		CAD	LSR	RG			SP3B	UBA	AND			NZRO	RG:=UBA:=LENGTH - 1;
1482																UBB:=ABSCENCE AND TRACE BITS, SKIP IF <> 0
1483	0137			UBB	JSB	TABP		NZRO	UBA	OPB	ADD		PL	ROP		JSB IF TRACE OR ABSENT;
1485																PL:=LENGTH - 1 + PB, READ STTL
1486	0138		2000	OPA	IORL				UBB	SP2B	SUB			ROP	NPRV	UBA:=AMRT-LENGTH/4 WITH REFERNECE BIT SET;
1488																READ LABEL AT PL - STT#, SKIP IF NPRV
1489	0139			UBA	ADD			DATA		OPB	ADD	LRZ		SF1		WRITE AMRT-LENGTH/4;
1491																UBB:=# LOCAL LABELS, SF1 IF PRIVILEGED
1492																(F1 WILL BE SET IF PRIVILEGED SEGMENT OR
1493																CALL FROM PRIVILEGED) (2635)
1494	013A			UBB	SP2B	JSZS	STTV		NEG	3FFF	OPB	ANDL		SP2B		STT VIOLATION IF STT# > STTL;

```
RECORD  C.S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1496                                                                                  SP2B:= DELTA P                                  (2635)
1497    013B        SP2B ADD                 ZERO      OPB  ADD                       SKIP IF STT#:=0;UBB:=LABEL AT PL-STT# (2635)
1499    013C        UBB  ADD  LSL       POS   PL   RG   SUB       SP3B       NF4B  SKIP IF NOT UNCALLABLE.
1501                                                                                  SP3B:=NEW PB (PL-LENGTH-1), SKIP IF STT# = 0
1502                                                                                                                  (2635)
1503    013D        STA  JSZ  UNCL      POS   RG   SP2B UBNE                  UNC   UNCALLABLE TRAP IF (STT# <> 0,LABEL.[1:1]=1,
1505                                                                                    AND NOT PRIVILEGED) OR (STT#:=0 AND NOT
1506                                                                                    PRIVILEGED);
1507                                                                                  BOUNDS CHECK SEGMENT LENGTH - 1 >= DELTA P
1508                                                                                    IF STT# <> 0, SKIP                            (2635)
1509    013E   0003 XR15 ANDL      SP4A        SP3B       ADD       P     RONP UNC   SP4A:= DUMP OR LOAD BIT;
1511                                                                                  P:=PB, READ INSTRUCTION IF STT# = 0   (2635)
1512    013F        RD   ADD       STA  F1HB   SP3B SP2B ADD       P     RONP       STA:=NEW STA, SET MODE BIT;
1514                                                                                  P:=NEW P (DELTA P + PB)                       (2635)
1515    0140        SP4A ADD                 NZRO       XR1  ADD       XR12       SKIP IF DUMP OR LOAD;UPDATE MAPFLG(2318)
1517                                                                                    READ INSTRUCTION IF STT# <> 0
1518    0141             JSZ  NEXT      UNC   SP3B       ADD       PB    CCTX       NEXT IF NOT LOAD OR DUMP;
1520                                                                                  PB:=NEW PB, CLEAR CTX FOR #CLDH
1521    0142             ADD                      A690       ADDL                     ; UBB := mask
1523    0143             ADD                           UBB  ADD             CCPX     ; Clear BNDVINT and MSGINT, set RUNFF
1525    0144   0004 XR15 ANDL      XR15        00B2       ADDL      CTR              Clear SINCBIT for clock interrupt;
1527                                                                                    CTR := SWREG REGN#
1528    0145             ADD                           XR16 ADD       REGN           SWREG:=XR16 (MCDEV#)
1530    0146             ADD                                ADD       BNKD       NEXT  ; BNKD:=0, NEXT
1532    0147   PCL8 SPOA ADD                 ROB3            JSB  *+1             F2    REREAD CSTL; MEDIUM JUMP TO WAIT ONE CLOCK
1534    0148             JSZ  CSTV      UNC        OPB  JSZ  CTLV                ZERO  CST VIOLATION UNLESS...;
1536                                                                                  CST LENGTH VIOLATION IF CSTL = 0
1537    0149   TABP      UBA  ADD                  DATA       UBA  ADD       RE         NPRV  WRITE AMRT-LENGTH/4 WITH REFERENCED BIT SET;
1539                                                                                  RE.(0:1):=ABSCENCE BIT, SKIP IF NPRV
1540    014A             SP1B ADD       SPOA                       ADD             SF1   SPOA:=PARAMETER:=LABEL; SF1 IF PRIVILEGED
1542    014B             RD   ADD       RD   F1HB        XR1  JSB  TABS XR12       UNC   SET MODE BIT IN TSTA; JSB                 (2318)
1544    014C   TABE      DSPL ADD       SPOA                       ADD             CF2   SPOA:=PARAMETER:=N;
1546                                                                                  CF2 TO DISTINGUISH EXIT FORM PCAL
1547    014D   TABS      RC   JSB  TABT      ZERO                  ADD                   BYPAS SEG 1 TEST FOR LOG MAPPED ABSENC(CSTX)
1549    014E        1000 CPX2 ANDL                 00FE RD   ANDL                 NZRO  UBA:=ICSFLAGFF; SKIP IF NEW SEG# > 1  (CSTX)
1551    014F             UBA  JSZ  ICSH      NZRO                  JSZ  ABTH      UNC   HALT IF ABSCENCE OR TRACE ON ICS;
1553                                                                                  HALT IF ABSCENCE OR TRACE GOING TO SEG# 1
1554    0150   TABT           JSZ  TRCE      UNC        RE   JSZ  ABCS            NEG   TRACE trap unless ... ;
1556                                                                                    ABSENCE trap if absence bit           (CSTX)
1557           $WARN
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1559           ***************************************************************
1560           *                                                               *
1561           *    Load Label (LLBL) Instruction                              *
1562           *                                                               *
1563           *    N = disp                                                   *
1564           *                                                               *
1565           ***************************************************************
1566           *
1567           $LUT INSTR=LLBL:0 011 011 1xx xxx xxx, DSPL=8, ENTRY=LLBL
1568           *
1569    0151   LLBL       STA  ADD  RLZ  RH          PL        ADD             ROP        RH:0:8=SEG# ;READ STT LENGTH & #INTS
1571    0152         UBB  DSPL SUB             ROP              JSZ  PSHM            SR7   READ LABEL (PL-N); SAVE RG IF SR=7
1573    0153              DSPL CAD       SP4A          OPB  ADD  RRZ        EPSH        SP4A=UBA=-(N+1) ;UBB=STT LENG,PUSH RH TO TOS
1575    0154         UBB  UBA  JSZI STTV       NCRY             ADD       BKX3        STT VIOL IF N>STT LENG; BKX3=0          (CSTX)
1577    0155              STA  ADD  RRZ  SP0A          OPB  ADD  LRZ  SP3B            SP0A=STA(8:8)    ; SP3B=UBB=#INTERNALS  (CSTX)
1579    0156         UBB  DSPL JSBS LLB3       NCRY        XR12 ADD  LSL        HBF2   JMP IF N>#INTS; F2=MAPFLAG              (CSTX)
1581    0157              DSPL JSZ  STTV       BIT8             ADD                    STT VIOL IF N>127;                      (CSTX)
1583    0158                   ADD             NEXT  RH   SP4A CAD  SWAB RH   F2HB     NEXT>RH ON TOS;RH=INT LABL CHGD TO EXT(CSTX)
1585    0159   LLB3       OPA  ADD  RRZ  RH          0291      ADDL                   RH=LABEL(8:8);UBB=ADDR OF LSTT BANK     (CSTX)
1587    015A         UBB  INC                ROX3        UBB  ADD             ROX3   READ LSTT BASE; READ LSTT BANK (291)   (CSTX)
1589    015B         OPA  ADD       RH                  RH   JSZ  NEXT             NZRO  RH=LABEL ;NEXT IF SEG# <>0               (CSTX)
1591    015C         OPA  ADD       SP0A                OPB  ADD       BKX4        SP0A=LSTT BASE ;BKX4=LSTT BANK          (CSTX)
1593    015D   SP0A       ADD  LSL                    RH   JSZ  STTV             NEG   UBA=STA(8:8*2 ;STTV IF SEG=0 & PHY MAP(CSTX)
1595    015E   SP0A  UBA  INC                ROX4  SP3B SP4A INC                    READ PTR LSTTB+STA*2+1;UBB=-(N-#INTS)   (CSTX)
1597    015F   SP0A  UBB  ADD                                 ADD                    UBA=LSTTB-(N-#INTS);                    (CSTX)
1599    0160   UBA  OPA  ADD                ROX4        BKX4 ADD             ZERO  READ LABEL FROM LSTTB+PTR-(N-#INTS)     (CSTX)
1601    0161   SP0A       JSZ  CSTV       ZERO             JSB  *+1              UNC   JSZ IF BOTH LSTT BANK=ADDRESS=0          (CSTX)
1603    0162              OPA  ADD       RH                  ADD                    NEXT  RH=LABEL FROM LSTT; NEXT                (CSTX)
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1606            ************************************************************************
1607            *                                                                      *
1608            *          ICF/55 EXIT instruction                                     *
1609            *                                                                      *
1610            *          This module will perform the execution of the EXIT          *
1611            *          instruction:                                                *
1612            *                                                                      *
1613            *               1) SM := Q; RQ := Q - (Q); RS := Q - 4 - N              *
1614            *               2) If RS or RQ > Z then stack overflow                 *
1615            *               3) If RS or RQ < DB and EXIT to nonpriv then           *
1616            *                  stack underflow                                     *
1617            *               4) If nonpriv EXITs to priv or external                *
1618            *                  interrupt bit changes then mode violation trap      *
1619            *               5) Disable external interrupts, STA(1) := 0            *
1620            *               6) Restore X register from stack marker, X := (Q-3)    *
1621            *               7) If EXIT to different segment or mapping mode,       *
1622            *                  then set up new segment                             *
1623            *               8) If PB <= PB + (Q-2) <= PL is not within limit       *
1624            *                  then bounds violation                               *
1625            *               9) P := PB + (Q-2); STA := (Q-1); S := RS; Q := RQ      *
1626            *              10) Next instruction                                    *
1627            *                                                                      *
1628            ************************************************************************
1629            *
1630            $LUT INSTR=EXIT:0 011 001 1xx xx, DSPL=8, ENTRY=EXIT
1631            *
1632  0163  EXIT      Q    ADD       SPOA ROS   FFFC Q    ADDL      SP1B           SPOA := Q, read delta Q; SP1B := UBB := Q-4
1633  0164      UBB DSPL SUB       RH         UBA      CAD       SP3B ROS       RH := RS (Q-4-N); SP3B := Q-1, read STA
1634  0165      SPOA OPA SUB       RG   FSS        Q    ADD       SM   CLSR      RG := RQ (Q-delta Q), skip if split banks;
1635                                                                              SM := Q, clear SR
1636  0166      UBA  DB  JSBS EXSU       NCRY SP3B      CAD            ROAS      JSB for further checks if DB < RQ and not
1637                                                                              split banks; Read delta P at (Q-2)
1638  0167           STA ADD       SP4A       Z    RH   JSZC STO2           NCRY  SP4A := UBA := oldSTA;  Stack overflow if
1639                                                                              not (Z > RS) logically. (S will be at
1640                                                                              least 4 less than Z if less than Z at all)
1641  0168      RG   RH  JSZC STUN       CRRY UBA  OPB  XOR       SP3B           Stack underflow if RQ > RS;                (CSTX)
1642                                                                              SP3B.(1:1) := new ext interrupt bit <> old
1643  0169           OPA ADD       RC         BFFF      ADDL      SP2B           RC=DELTA P W/ L/P BIT;UBB=MSK TO CLR L/P BIT
1644                                                                                                                     (CSTX)
1645  016A           SP1B INC            ROS        OPB  ADD       RD        NEG  Read newX at (Q-3);                        (CSTX)
1646                                                                            D=NEW STA,SKIP IF GOING TO PRIVILEGED
1647  016B      RC   SP2B AND       RB         SP3B      ADD  LSL            POS  RB := UBA := new delta P; Skip if the new &
1648                                                                              old ext interrupt bit are the same     (CSTX)
1649  016C           OPA ADD       X                         JSB  TR6E       NPRV X := NEW X                             (CSTX)
1650                                                                              trap if not privileged and EXIT is to priv
1651                                                                            or attempt to change ext interrupt bit(CSTX)
1652  016D      RB   PB  ADD       SP4A       RC   XR12 ADD  LSL            POS  SP4A=NEW P;NOP IF NEWMAPFG=OLDMAPFLAG (CSTX)
1653                                                                            TRICK: ADD=XOR W/O CARRIES--XR12=0.4000
1654  016E      RD   SP4A SUB  RRZ              PL   PB   JSBS EXI6       UNC  UBA=NEW PGM SEG-OLD PGM SEG;               (CSTX)
1655                                                                            UBB=PL-PB, SET UP NEW PGM SEG IF MAPFLAGS
1656                                                                            DIFFERENT                                (CSTX)
1657  016F      UBB  RB  JSZS BNDV       NCRY UBA       JSB  EXI6       NZRO BNDV IF DLTA P>PL-PB;EXI6 IF PRGSEG NOT SAME
1658                                                                                                                     (CSTX)
1659  0170  EXI9 1000      ADDL                       RH   ADD       SM         UBA := left/right stackop mask; SM := RS
1660  0171      UBA  RD  AND             NZRO       SP4A ADD       P    RNSP  Skip if right stackop, P := newP, read right
1661                                                                              stackop
```

RECORD NO	C S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
1677	0172			SP4A	ADD			RONP		RG	ADD		Q			Read next instruction; Q := RQ
1679	0173			CPX2	ADD					RD	ADD	LSL			POS	UBA := CPX2; Skip if interrupts will be off
1681	0174			RD	ADD		STA	UNC		UBA	JSZ	DIMS			ODD	STA := newSTA, skip JSB to mode trap; JSB if deferred interrupt and interrupts on
1683																
1684	0175	TR6E			JSZ	TRP6		UNC			ADD				NEXT	Mode violation if entered here; Else NEXT
1686	0176	EXSU		DL	ADD				Z	DB	SUB				NCRY	UBA := DL; Skip if DB > Z
1688	0177				ADD			RSB	UBA	DB	JSZC	STUN			NCRY	RSB; Stack underflow if not on split stack (not split stack if DL <= DB <= Z)
1690																

```
RECORD  C.S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1692            $NOWARN
1693            ********************************************************************
1694            *                                                                  *
1695            *          CODE SEGMENT SET UP MODULE FOR EXIT INSTRUCTION         *
1696            *                                                                  *
1697            *          This module is called by EXIT to set up execution       *
1698            *          of a particular code segment:                           *
1699            *                                                                  *
1700            *               1)  if physical or logical mapped with seg<=nr prog*
1701            *                   segs, then read CSTB(0), else read CSTX(1)     *
1702            *               2)  if log map, then index into LSTT & get seg #   *
1703            *                   CST violation or system halt if CSTL = 0.      *
1704            *               3)  set PB BANK, PB and PL from CST entry.         *
1705            *               4)  set reference bit in CST entry #1.             *
1706            *               5)  STATUS register is set to new segment #.       *
1707            *               6)  if TRACE/ABSENT flag in CST is set then        *
1708            *                   TRACE/ABSENT trap.                             *
1709            *               7)  set mode in STATUS (PRIV or NON-PRIV).         *
1710            *               8)  P := PB + delta P.                             *
1711            *               9)  finish up like internal EXIT.                  *
1712            *                                                                  *
1713            ********************************************************************
1714            *
1715  0178 EXI6 RC   XR12 AND        RC                      ADD        BKX3 SF2        RC=MASKED MAPPING FLAG;CLR BK3          (CSTX)
1717  0179      0292      ADDL                                RD   ADD  RRZ  RF         UBA=ADDR FOR LSTT;RF=NEW SEGMENT NUMBR(CSTX)
1719  017A                ADD                 ROX3 UBA        INC             ROX3      READ CSTB(0) ; READ # PROG SEGS (0293)(CSTX)
1721  017B          RC    JSB  EXI7           NZRO RREG       CAD        RE   ROX3      JMP IF PHYS MAPPD;READ LSTT BANK(0291)(CSTX)
1723  017C                INC                 ROX3 RF   OPB  CAD                 CRRY  READ CSTX(1) ;SKIP IF SEG# > #PROG SEG(CSTX)
1725  017D          OPA   ADD         SPOA                    JSB  EXI7           UNC   SAVE CSTB(0) IN SPOA,JMP USING CSTX(1)(CSTX)
1727  017E          RE    INC                 ROX3      OPB  ADD        BKX4            READ LSTT ADDR (292);SET LSTT BANK      (CSTX)
1729  017F    RF    RF    ADD                           UBB  ADD        BKX3            UBA=SEG# *2; BKX3=BKX4=LSTT BANK        (CSTX)
1731  0180    UBA   OPA   XFRS                ROX4      BKX4 ADD                    NZRO  READ #LSTT ENTS; UBB=LSTT BNK,SKP IF 0(CSTX)
1733  0181    RREG  UBA   ADD                 ROB3      UBA  JSZ  CSTV             ZERO  READ LSTTB+SEG*2,CST VIO IF LSTT BANK=ADDR=0
                                                                                          (CSTX)
1736  0182          OPA   ADD  RRZ                                ADD        BKX3            UBA=#LSTT ENTRIES ;CLR BKX3 FOR FOLLWG(CSTX)
1738  0183    RF    UBA   JSZC CSTV          CRRY      OPB  JSZ  CSTV RF             ZERO  JP IF SEG>#LSTT EN;OPB=RF=PHYS SEG#,JMP IF 0
1740                                                                                      (CSTX)
1741  0184                ADD                                   JSB  EXI8           UNC                 ;BYPASS--SPOA ALREADY SET(CSTX)
1743  0185 EXI7       OPA   ADD         SPOA                    JSB  *+1            F2    SPOA=BASE;MED JMP TO LET SPOA TO STORE(CSTX)
1745  0186 EXI8 SPOA        ADD                 ROX3 RF   RF    INC  LSL                  READ CSTL;UBB=SEG# *4+2                 (CSTX)
1747  0187      UBA   UBB   ADD         SP4A ROB3 RF   RF    ADD  LSL            NZRO  SP4A:=(CSTX/CSTB) * SEG# * 4 + 2, READ BNKP;
1749                                                                                      UBB:=SEG# * 4, SKIP IF <> 0
1750  0188      SPOA  UBB   ADD                 ROX3      -       JSB  PCL8           UNC   READ AMRT-LENGTH/4 AT (CSTX/CSTB) + SEG#*4;
1752                                                                                      CST VIOLATION IF SEG# = 0
1753  0189      RF    OPA   JSBC PCL8           CRRY      00FF OPB  ANDL       BNKP         CST OR CTL VIOLATION IF SEG# > CSTL;
1755                                                                                      BNKP:=NEW BNKP
1756  018A      OFFF  OPA   ANDL                                  SP4A INC            ROX3  UBA:=LENGTH/4;
1758                                                                                      READ PB AT (CSTX/CSTB) + SEG# * 4 + 3
1759  018B            OPA   ADD  LSL            HBF2 UBA  UBA  ADD  LSL                   F2:=MODE BIT; UBB:=LENGTH
1761  018C      2000  OPA   IORL                      UBB       CAD        RF             UBA:=AMRT-LENGTH/4 WITH R BIT SET,
1763                                                                                      RF:=UBB:=LENGTH - 1
1764  018D            UBA   ADD         RE   DATA UBB  OPB  ADD        PL          NF2   RE.(0:1):=ABSCENCE BIT FOR #TABE,
1766                                                                                        WRITE AMRT-LENGTH/4;
1767                                                                                      PL:=LENGTH - 1 + PB
1768                                                                                        SKIP IF SEGMENT NOT PRIVILEGED
1769  018E            SREG JSB  TABE            NEG       OPB  JSZ  TRP6           NPRV  JSB IF ABSCENT;
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
1771																UBB:=PB, TRAP IF EXIT TO PRIVILEGED AND NOT
1772																PRIVILEGED
1773	018F		RB	UBB	ADD		SP4A		RB	JSB	TABE				NEG	SP4A:=DELTA P + PB;
1775																TRACE IF DELTA P.(0:1)
1776	0190				ADD				OPB	ADD			PB			;PB=NEW PB (CSTX)
1778	0191		RF	RB	UBNE				RC	JSB	EXI9	XR12			UNC	BNDV IF LNGTH-1>=DEL P;SET NEWMAPFLG (CSTX)
1780		$WARN														

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1782            ************************************************************************
1783            *                                                                      *
1784            *          ICF/55 IXIT Instruction                                     *
1785            *                                                                      *
1786            *          This module will perform the execution of the IXIT          *
1787            *          instruction.  IXIT is executed only on the ICS and          *
1788            *          can be entered by the interrupt procedures or the           *
1789            *          Dispatcher.  For interrupt procedures:  If redispatch       *
1790            *          requested QI[0] = 1.  If Dispatcher interrupted             *
1791            *          Q[0] = 1 and Q <> QI.  If pseudo-disabled QI-18 <> 0.       *
1792            *                                                                      *
1793            *          1).  Dispatcher launch of a process.                        *
1794            *          2).  Interrupted process, return to process.                *
1795            *          2a). Interrupted process, redispatch requested              *
1796            *               but pseudo-disabled so return to process.              *
1797            *          3).  Interrupted interrupt, return to interrupt.            *
1798            *          4).  Interrupted Dispatcher, return to Dispatcher.          *
1799            *          4a). Interrupted Dispatcher, redispatch requested           *
1800            *               but pseudo-disabled so return to Dispatcher.           *
1801            *          5).  Interrupted process, redispatch requested so           *
1802            *               start Dispatcher.                                      *
1803            *          6).  Interrupted Dispatcher, redispatch requested           *
1804            *               so restart Dispatcher.                                 *
1805            *                                                                      *
1806            *          IXIT paths:  (1)                IXT1  IXT7  IXT8       *
1807            *                       (2), (2a)    IXT2  IXT1  IXT7  IXT8       *
1808            *                       (3)          IXT2  IXT3  IXT7  IXT8       *
1809            *                       (4), (4a)    IXT2  IXT4  IXT3  IXT7  IXT8 *
1810            *                       (5), (6)     IXT2  IXT4  IXT9  IXT8       *
1811            *                                                                      *
1812            *          The displacement for IXIT is set to four so that in         *
1813            *          the event of a TRACE or absence trap on IXIT, zero          *
1814            *          is placed on the stack as the parameter.                    *
1815            *                                                                      *
1816            ************************************************************************
1817            *
1818            $LUT INSTR=IXIT/PCN:0 010 000 011 11, DSPL=4, ENTRY=IXIT
1819            *
1820    0192    IXIT      DSPL ADD            ZERO       Q    ADD            ROAS      Skip if IXIT instr; Read delta Q
1822    0193              CPX2 JSB  PCN       UNC             JSZ  TRP6 BKX3       NPRV  It's a PCN instr; BKX3 := 0, mode violation?
1824    0194         FFFB Q    ADDL     RH          0100 UBA  ANDL                 ZERO  RH := Q-5; Are we in Dispatcher?
1826    0195              OPA  ADD      RD                    JSB  IXT1           UNC   RD := delta Q; Yes, so launch process
1828    0196    IXT2      UBA  JSB  IXT3      NZRO       UBA  JSB  IXT4           NEG   If Q[0] = 1 then path (4,5,6) else if
1830                                                                                      Q[0] = 0 and Q <> QI then path (3)
1831                                                                                      (Note:  Q = QI iff delta Q = 0)
```

```
RECORD  C S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1833            ************************************************************************
1834            *                                                                      *
1835            *         ICF/55 IXIT path (1,2,2A)                                    *
1836            *                                                                      *
1837            *         This module will launch the interrupted process,             *
1838            *         dispatcher launch of process or launch of process            *
1839            *         if Dispatcher is pseudo-disabled.                            *
1840            *                                                                      *
1841            *              1).  S-BANK := (QI-5).                                  *
1842            *              2).  Q := (QI-6) - 2 + (QI-4).                          *
1843            *              3).  DL := (QI-7) + (QI-4).                             *
1844            *              4).  Z := (QI-8) + (QI-4).                              *
1845            *              5).  Clear Dispatcher and ICS flags.                    *
1846            *              6).  Read delta Q.                                      *
1847            *                                                                      *
1848            ************************************************************************
1849            *
1850    0197    IXT1 FF00       ADDL                RH        CAD       SP3B ROX3      UBA:=MASK TO CLEAR DISP AND ICSFLAG;
1852                                                                                     SP3B:=(QI-6), READ DS
1853    0198            RH   INC            ROX3        UBA  ADD             CCPX      READ STKDB (QI-4); CLEAR DISP AND ICS FLAG
1855    0199            RH   ADD            ROX3  SP3B       CAD       RH   ROX3      READ S-BANK (QI-5); READ DDL (QI-7)
1857    019A            OPA  ADD       RG   CF1   FFFE OPB  ADDL                     RG:=STACK DB, (Q); UBB:=DS-2
1859    019B       RH        CAD            ROX3  UBA  UBB  ADD       Q             READ DZ(QI-8); Q:=DS-2+(QI-4)
1861    019C            OPA  ADD            RG    OPB  ADD       DL            UBA:=S-BANK; DL:=DDL+(QI-4)
1863    019D       RG   OPA  ADD       SPOA        UBA  ADD       BNKS          UBA:=DZ(QI-8)+(QI-4); BANKS:=S-BANK(QI-5)
1865    019E            Q    ADD       RH    ROS   UBA  ADD       Z             READ(DELTA Q); Z:=DZ (QI-8)+(QI-4)
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1868            ********************************************************************
1869            *                                                                  *
1870            *          ICF/55 IXIT PATH (1,2,2A,3,4,4A)                        *
1871            *                                                                  *
1872            *          THIS MODULE JOINS PATHS (1,2,2A) AND (3,4,4A).          *
1873            *                                                                  *
1874            *              1).  RS:=Q-4.                                       *
1875            *              2).  RQ:=Q-DQ.(1:15) OR RA:=Q-DQ IF PATH (1,2,2A).  *
1876            *                                                                  *
1877            ********************************************************************
1878            *
1879    019F    IXT7 UBB  SP4A SUB        RG   F1          Q    INC            ROS        Q:=Q-DQ(1:15); READ(DB-BANK)
1881    01A0         RH   OPA  SUB        RG               UBB  INC            ROS        RQ:=Q-DQ, IF PATH (1,2,2A); READ(DB)
1883    01A1         FFFC Q    ADDL       RH               OPB  ADD       BNKD CLSR       RS:=Q-4; DB-BANK:=(Q+1),SR:=0
1885    01A2    IXT8 UBA  INC            SP4A ROS         Q    ADD       SM              SP4A:=Q-3, READ X; S:=Q
1887    01A3         SPOA RH   JSZS STOV       NEG         OPB  ADD       DB              STOV IF RS > Z AND WITHIN 32K;
1889                                                                                     DB:=(Q+2)
1890    01A4         SPOA RG   JSZS STOV       NEG         SP4A INC            ROAS       STOV IF RQ > Z AND WITHIN 32K; READ DELTA P
1892    01A5              OPA  ADD        X                UBB  INC            ROS        SET X REGISTER; READ STATUS
1894    01A6              OPA  ADD        RC          BFFF      ADDL                      RC=DELTA P IOR L/P BIT; UBB=L/P MASK (CSTX)
1896    01A7         UBB  OPA  AND        RB          PL   PB   SUB       SP3B           RB:=DELTA P W/O L/P BIT; SP3B:=PL-PB (CSTX)
1898    01A8         UBA  PB   ADD        SP4A             OPB  ADD       RD              SP4A:=DELTA P + PB; RD:=TSTA
1900    01A9         UBB  STA  SUB  RRZ   RE               BNKS ADD                     RE:=0 IF SEGMENT NUMBERS SAME; UBB:=BNKS
1902    01AA         3FFF      ADDL                   UBB  BNKD XOR                 ZERO  UBA:=MASK; SKIP IF NOT SPLIT BANKS
1904    01AB         1000      ADDL                        UBA  INC            CCPX UNC  UBA:=MASK; SET FSS IF SPLIT BANKS, SKIP
1906    01AC                   ADD                    UBA  UBB  IOR            CCPX                    ;CLR FSS          (CSTX)
1908    01AD              RE   JSB  EXI6       NZRO  4000 RC   ANDL      RC              JSB IF SEG CHG;MASK TO NEWMAPFLG          (CSTX)
1910    01AE                   ADD                    UBB  XR12 JSBS EXI6          NZRO  JMP IF BOTH L/P BITS NOT EQUAL            (CSTX)
1912    01AF                   JSB  EXI9       UNC   SP3B RB   UBNE                      JSB TO FINISH;
1914                                                                                       BOUNDS CHECK (PL - PB) >= DELTA P
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1916           ********************************************************************
1917           *                                                                  *
1918           *          ICF/55 IXIT PATH (3,4,4A)                               *
1919           *                                                                  *
1920           *          THIS MODULE WILL CREATE THE DELTA Q ADJUSTMENT          *
1921           *          FROM DQ.(1:15).                                         *
1922           *                                                                  *
1923           *               1).  DQ:=DQ.(1:15).                                *
1924           *               2).  JOIN PATHS (1,2,2A).                          *
1925           *                                                                  *
1926           ********************************************************************
1927           *
1928    01B0   IXT3 RD   RD   ADD  LSR  SP4A SF1   Z         ADD                    SP4A:=DQ(1:15); UBB:=Z
1930    01B1          UBB  ADD       SPOA            Q     JSB  IXT7         UNC   SPOA:=Z; UBB:=Q, JSB
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1933            ************************************************************************
1934            *                                                                      *
1935            *         ICF/55 IXIT Paths (2A,4,5,6)                                 *
1936            *                                                                      *
1937            *         This module checks to see if the DISPATCHER was              *
1938            *         interrupted or a re-DISPATCH was requested.                  *
1939            *                                                                      *
1940            *              1). Set DISP flag.                                      *
1941            *              2). If QI(0) = 0 return to DISPATCHER (path 4).         *
1942            *              3). If QI <> Q return to DISPATCHER (path 4A).          *
1943            *              4). If (QI-18) = 0 start DISPATCHER (path 5,6).         *
1944            *              5). Return to launch process (path 2A).                 *
1945            *                                                                      *
1946            ************************************************************************
1947            *
1948    01B2  IXT4 0005      ADDL                    0700      ADDL                    SET ADR 5; SET DISP FLAG (PATH 2A,4,5,6)
1950    01B3          UBA    ADD             ROX3      UBB    ADD           CCPX      READ(QI ADDR); SET DISP FLAG
1952    01B4                 ADD             CF1     0012     ADDL    SP1B            CLEAR DQ FLAG; SP1B:=18
1954    01B5          OPA    ADD       SP4A  ROX3    Z        ADD                     READ(QI); UBB:=Z
1956    01B6          UBB    ADD       SP0A        UBA   SP1B SUB           ROX3      SP0A:=Z; READ(QI-18)
1958    01B7          OPA    JSB  IXT3        POS         Q    ADD                     RETURN TO DISP (PATH 4)) IF QI.(0:1)=0;
1960                                                                                   UBB:=Q
1961    01B8      UBB SP4A JSBS IXT3        NZRO        OPB   JSB  IXT9          ZERO  RETURN TO DISP (PATH 4A) IF Q <> QI;
1963                                                                                   DISPATCH IF (QI-18)=0, (NOT PSEUDO DISABLED)
1964    01B9                 ADD                                 JSB  IXT1          UNC   ; RETURN TO INT. PROC (PATH 2A)
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1967            *****************************************************************
1968            *                                                               *
1969            *          ICF/55 IXIT PATH (5,6)                               *
1970            *                                                               *
1971            *          THIS MODULE WILL START THE DISPATCHER AFTER A         *
1972            *          INTERRUPTTED PROCESS HAS COMPLETED.                  *
1973            *                                                               *
1974            *               1).  (QI):=0.                                   *
1975            *               2).  Q:=QI.                                     *
1976            *               3).  RS:=QI+2                                   *
1977            *               4).  RQ:=QI                                     *
1978            *               5).  JOIN ALL OTHER PATHS (IXT8).               *
1979            *                                                               *
1980            *****************************************************************
1981            *
1982    01BA    IXT9       SP4A INC       SP4A ROS        SP4A ADD       Q     WRX3        READ(DB-BANK);Q:=QI, WRITE TO QI
1984    01BB         0002  UBB  ADDL      RH                   ADD            DATA        RS:=QI+2; QI:=0
1986    01BC               OPA  ADD                       SP4A INC            ROS         UBA:=DB-BANK; READ(DB)
1988    01BD               Q    ADD       RG              UBA  ADD       BNKD            RQ:=QI; DB BANK:=UBA
1990    01BE         FFFC  Q    ADDL                           JSB  IXT8             UNC   UBA:=Q-4; FINISH TO PATH (1,2,2A,3,4,4A,5,6)
```

```
RECORD  C S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

1993            ************************************************************************
1994            *                                                                      *
1995            *           ICF/55 PCN instruction                                     *
1996            *                                                                      *
1997            *           This module will push a four (4) onto the top of           *
1998            *           stack to signify an ICF/55 CPU.                            *
1999            *                                                                      *
2000            *           Entered from #IXIT with privileged mode already            *
2001            *           checked.                                                   *
2002            *                                                                      *
2003            ************************************************************************
2004            *
2005    01BF    PCN  FFFC DSPL ANDL                          ADD                       Check for non-PCN instructions
2007    01C0              DSPL JSZ  NEXT      ODD       UBA  JSZ  TRP           NZRO  NEXT if old LOCK/UNLK instruction;
2009                                                                                    Unimplemented trap if not PCN or LOCK/UNLK
2010    01C1                   JSZ  PSHM      SR7            ADD                       JSZ to empty one TOS if SR = 7
2012    01C2                   JSZ  NEXT      EPSH  0004     ADDL      RH              JSZ for NEXT, EPSH; New (S) := 4 for ICF/55
```

```
RECORD  C S                ********** ALU A **********  ************** ALU B **************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2015            ********************************************************************
2016            *          SCAL - SUBROUTINE CALL INSTRUCTION                      *
2017            ********************************************************************
2018            *
2019            $LUT INSTR=SCAL:0 011 000 1xx xxx xxx, DSPL=8, ENTRY=SCAL
2020            *
2021    01C3    SCAL       DSPL ADD             ZERO  PL   PB   SUB        SP2B TICB       SKIP IF N=0
2023                                                                                      SP2B:=PL - PB, TICB CAUSES CTR:=REGN CODE
2024                                                                                        FOR OPB
2025    01C4                    JSB  SCNZ        UNC   PL   UBA  SUB             ROP  SRZ  JSB IF N <> 0; READ (PL - N), SKIP IF SR = 0
2027    01C5               SM   JSB  SMEM        ROBS                JSB  STOS CTR       UNC  READ (SM) AND JSB IF SR = 0;
2029                                                                                      CTR:=REGN CODE FOR RA AND JSB IF SR<>0(CSTX)
2030    01C6    SCNZ            JSZ  PSHM        SR7   PL   REGN XFRR            ROP       RG>MEM IF SR=7;READ #INT,OPB/RA BEFORE POP
2032                                                                                                                        (CSTX)
2033    01C7                    ADD                   3FFF SREG ANDL       SP3B            ;SP3B:=DELTA P                  (CSTX)
2035
2036
2037    01C8            P   ADD       SPOA                  OPB  ADD  LRZ                 SPOA=P;UBB=#INTERNALS HI ORDR STT LENG(CSTX)
2039    01C9       UBB  DSPL JSZS STTV        NCRY  SP3B        ADD                       STTV IF N>#INTS; UBB=DELTA P            (CSTX)
2041    01CA       UBB  SP2B JSZC BNDV        CRRY  SP3B  PB    ADD             RONP      BNDV IF ADDR > (PL - PB)                (CSTX)
2043                                                                                      UBB:=ADDR + PB, READ
2044    01CB       SPOA PB   SUB         RH    EPSH        UBB  JSZ  NEXT P        UNC  NEW (S):=P - PB, EPSH; P:=NEW P, NEXT
2046    01CC    STOS           ADD             UNC                 ADD             POP  RSB  INH JSB TO PSHM;POP TOS IAW SCALO INST(CSTX)
2048    01CD    SMEM           ADD                         UBA   Q   JSZC STUN         NEG  ; STUN IF S WOULD BE LESS THAN Q
2050    01CE                   ADD             RSB   RREG        CAD        SM             RSB; SM:=SM -1
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2053            ************************************************************************
2054            *          SXIT - SUBROUTINE EXIT INSTRUCTION                          *
2055            ************************************************************************
2056            *
2057            $LUT INSTR=SXIT:0 011 010 0xx xxx xxx, DSPL=8, SR=1, ENTRY=SXIT
2058            *
2059    01CF    SXIT       SM   INC             WRS   SR   SM   REPC      SM   DCSR SRZ   WRITE TOS REGISTERS AT SM + 1;
2061                                                                                    SM:=SR + SM, SR:=SR - 1, REPEAT IF SR <> 0
2062    01D0               QDWN ADD             DATA  Z         ADD             DCSR SRL2  WRITE QDWN;
2064                                                                                    UBB:=Z, SR:=SR - 1, END WHEN SR IN RANK1 < 2
2065    01D1       UBB  SM   JSZS STO2          NEG         SM   ADD             CLSR       STACK OVERFLOW IF Z < SM AND WITHIN 32K;
2067                                                                                    UBB:=SM, CLSR
2068    01D2       UBB  DSPL CAD       SP4A     PL    PB   SUB                             SP4A:=UBA:=S-DSPL-1; UBB:=PL - PB
2070    01D3       UBA  Q    JSZS STUN          NEG   UBB  RA   UBNE                        STACK UNDERFLOW IF SM < Q AND WITHIN 32K;
2072                                                                                    BOUNDS CHECK (PL - PB) >= (S)
2073    01D4                 ADD                      RA   PB   ADD       P    RONP       ; P:=(S) + PB, READ NEXT INSTRUCTION
2075    01D5                 JSZ  NOP         UNC         SP4A ADD       SM              NEXT IN TWO CLOCKS; SM:=S-1-DSPL
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2078            ************************************************************************
2079            *                                                                      *
2080            *    STNU - SR Preadjust Stack Underflow Routine                       *
2081            *                                                                      *
2082            *           The purpose of this routine is to move NIR into CIR        *
2083            *           when a stack underflow is detected during the SR           *
2084            *           preadjust.  Note that if a right stackop is the cause      *
2085            *           of the STUN it will appear as an unpaired left stackop.    *
2086            *           SM, SR and P are adjusted accordingly.  I cannot           *
2087            *           emphasize this enough:  If this routine is not broken,     *
2088            *           don't try to fix it!                                       *
2089            *                                                                      *
2090            ************************************************************************
2091            *
2092    01D6    STNX 0500      ADDL      SP4A                    ADD                  SP4A := mask to set FAKENEXT
2094    01D7           SM     INC             WRS   SR    SM     REPC       SM   DCSR SRZ   Force the CPU to drop the SRREQD line by
2096    01D8           QDWN   ADD             DATA              ADD             DCSR SRL2    accessing the stack registers
2098    01D9                  ADD                         SP4A  ADD             CCPX       Produce a FAKENEXT to move NIR to CIR before
2100                                                                                         clearing the SR register
2101    01DA                  ADD                               ADD             CLSR UNC   ; SR := ESR := 0, skip next line 1st time
2103    01DB                  ADD                         P     JSZI STUN P           UNC  Increment P if no right stackop pending
2105                                                                                         before jumping to process stack underflow
2106    01DC           1000 STA  ANDL                           ADD                        UBA := right stackop bit
2108    01DD                  JSZ  STUN       UNC         UBA   JSB  *-2           ZERO  Jump to increment P if no right stackop
2110                                                                                         pending else jump to process stack
2111                                                                                         underflow w/o incrementing P
```

```
RECORD  C S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2113            ************************************************************************
2114            *                                                                      *
2115            *          MAGNETIC TAPE I/O PROGRAM BUILD MODULE                      *
2116            *                                                                      *
2117            *          This module will build the Mag Tape I/O program to          *
2118            *          read the bootstrap I/O program.                             *
2119            *                                                                      *
2120            ************************************************************************
2121            *
2122    01DE    CLDT 0401 0001 0000 4400                      CON                      SELECT UNIT #0 FOR MAGTAPE
2124    01DF         0314 0200 0000 0500                      CON                      ISSUE WAIT
2126    01E0         0000 0000 0401 0001                      CON                      ISSUE DSJ
2128    01E1         0000 4400 0341 0200                      CON                      ISSUE READ COMMAND                    (2334)
2130    01E2         0000 0500 0000 0000                      CON                      ISSUE DSJ
2132    01E3         0300 0100 0440 8000                      CON                      READ BOOTSTRAP CHANNEL PROGRAM
2134    01E4         0E40 0000 0002 0000                      CON                      STARTING ADDR
2136    01E5         FFF2 0407 0001 0000                      CON                      DO ANOTHER BURST READ IF NOT DONE
2138    01E6         4400 0342 0302 0002                      CON                      ISSUE END COMMAND                     (2334)
2140    01E7         0000 0400 0343 0200                      CON                      READ XFER COUNT                       (2334)
2142    01E8         0000 0500 0000 0000                      CON                      ISSUE DSJ; ISSUE INTERRUPT HALT       (2334)
2144    01E9         0180 0000 0008 0013                      CON                      READ RECORD COMMAND, END COMMAND      (2334)
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2147            ********************************************************************"
2148            *    LRA; P relative                                                  "
2149            ********************************************************************"
2150            *
2151            $LUT INSTR=LRA(P+):1 111 000 0XX XX,DSPL=8,P,ENTRY=LAP
2152            $LUT INSTR=LRA(P+):1 111 100 0XX XX,DSPL=8,X,P,ENTRY=LAP
2153            $LUT INSTR=LRA(P-):1 111 000 1XX XX,DSPL=8,P,F2,ENTRY=LAP
2154            $LUT INSTR=LRA(P-):1 111 100 1XX XX,DSPL=8,X,F2,P,ENTRY=LAP
2155            *
2156    01EA    LAP  UBA  DSPL ADSB      RH        3200      ADDL                      RH:=UBA:=PB +/- DSPL;UBB=PCAL 0 OPCODE(2317)
2158    01EB         UBA  JSZ  PSHM      SR7   UBB OPB   JSBS LAP5           ZERO   JSB IF SR=7;JSB IF LRA FOLL BY PCAL0  (CSTX)
2160    01EC    LAP1 UBA  PB   SUB       RH    EPSH      JSZ  NEXT           UNC    RH:=EFFECTIVE ADDRESS - PB,EPSH; NEXT (2317)
2162            *
2163            ********************************************************************"
2164            *    LRA; Indirect P relative                                         "
2165            ********************************************************************"
2166            *
2167            $LUT INSTR=LRA(P+):1 111 010 0XX XX,DSPL=8,INDR,P,ENTRY=LAIP
2168            $LUT INSTR=LRA(P+):1 111 110 0XX XX,DSPL=8,X,INDR,P,ENTRY=LAIP
2169            $LUT INSTR=LRA(P-):1 111 010 1XX XX,DSPL=8,INDR,F2,P,ENTRY=LAIP
2170            $LUT INSTR=LRA(P-):1 111 110 1XX XX,DSPL=8,X,INDR,F2,P,ENTRY=LAIP
2171            *
2172    01ED    LAIP UBA  DSPL ADSB      RH    ROP   3200      ADDL                      RH:=UBA:=PB +/- DSPL, READ;UBB=PCAL0 OPCODE
2174    01EE         UBA  JSZ  PSHM      SR7   UBB OPB   JSBS LAI5           ZERO   JSB IF SR=7;JSB IF LRA FOLL BY PCAL0  (CSTX)
2176    01EF         XC   RH   ADD                 RH    PB   BNDE                    UBA:=XC + INDIRECT CELL ADDR;
2178                                                                                  BOUNDS CHECK INDIRECT CELL ADDR >= PB
2179    01F0         UBA  OPA  JSB  LAP1         UNC   PL    RH   BNDE                UBA=XC+ADDR+CELL CONT;JSB BNDS CK PL>=(2317)
2181    01F1    LAI5 XC   RH   ADD                 RH    PB   BNDE                    UBA=XC+INDR CELL ADR;BND CK INDIR>=PB (CSTX)
2183    01F2         UBA  OPA  ADD       RH        PL    RH   BNDE                    RH=XC+ADR+CELL CNT;BND CK PL>=INR   (2317)
2185    01F3    LAP5 SR   SM   ADD                 0500      ADDL      SP3B           UBA=SM+SR ;SP3B=FAKENXT CODE              (CSTX)
2187    01F4         0004      ADDL                Z     UBA SUB                      UBA=4 ;UBB=TOTAL AVAIL SPACE ON STACK (CSTX)
2189    01F5                   JSZ  PSHM      SR7   UBA   UBB JSZC STO2           CRRY   JMP IF STACK REGS FULL (MOVE OUT 1 TO MEM)
2191                                                                                  STACK OVERFLW IF NOT ENOUGH FOR PCAL 0(CSTX)
2192    01F6         RH   PB   SUB       RH    EPSH      ADD                          RH=EFF ADDR-PB,EPSH ;                     (2317)
2194                                                                                  GO DO INTERNAL PCAL0 DIRECTLY AND FAKE P
2195                                                                                  UBA=0 FOR FAKE DSPL FOR PCAL0
2196    01F7                   ADD         SP4A      SP3B      ADD            CCPX                                              (2317)
2198    01F8                   ADD                             ADD                      NOP TO ALLOW FOLLOWING SF1 TO WORK    (2326)
2200    01F9                   JSB  PCL1      SF1         P    INC        P     RONP                                              (CSTX)
```

```
RECORD  C S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2203            *
2204            ********************************************************************
2205            *                                                                  *
2206            *     CKPB:  Check to see if PB is valid. Zero BNKP if it is       *
2207            *            too large so that DUMP does not die of "Invalid Address  *
2208            *             -- CBI5" if BNKP is too large. Read NIR with RONP.   *
2209            *                                                                  *
2210            ********************************************************************
2211            *
2212    01FA  CKPB           ADD                  RB   BNKP SUB                CRRY  Is BNKP within installed memory ?   (2334)
2214    01FB                 ADD                       ADD       BNKP               No -- Zero BNKP if too large !    (2334)
2216    01FC                 ADD                       ADD                          Wait for store to occur.          (2334)
2218    01FD                 ADD                  RF   ADD            RONP         Read NIR.                         (2334)
2220    01FE                 ADD                       ADD                          Wait for read to occur.           (2334)
2222    01FF                 ADD                  OPB  ADD       RF         RSB   RF := NIR, return to DUMP         (2334)
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2225            ********************************************************************
2226            *                                                                  *
2227            *          COLD LOAD AND DUMP MODULE                               *
2228            *                                                                  *
2229            *          This module will perform the cold load or memory dump   *
2230            *          operations from magnetic tape or disc.                  *
2231            *                                                                  *
2232            *          1)  Entered with cold load/dump device number in RH,    *
2233            *              (from DCU).                                         *
2234            *          2)  Find the size of main memory.                       *
2235            *          3)  Initialize all of main memory to %30370 (HALT %10). *
2236            *          4)  Copy the contents of CPU registers to locations     *
2237            *              %1401-1422 in main memory.                          *
2238            *          5)  Copy the cold load device DRT to locations          *
2239            *              %1516-1521, the DRT bank and offset to %1522-1523,  *
2240            *              and the interrupt masks to %1524-1527.              *
2241            *          6)  I/O CLEAR all channels, initialize channel device,  *
2242            *              identify the device (returning the IDENT byte for   *
2243            *              the device,) initialize channel device and clear the*
2244            *              channel.                                            *
2245            *          7)  Copy the appropriate channel program to %1423-1505. *
2246            *              IDENT byte = 0 for disc, 1 for mag tape.            *
2247            *          8)  Modify the disc I/O program seek command to sector 2.*
2248            *          9)  Set up a DRT entry for cold load device.            *
2249            *         10)  Issue SIOP command to load 256 bytes from mag tape  *
2250            *              or disc to location %7100.                          *
2251            *         11)  When the channel program halts, compute and check   *
2252            *              the bootstrap checksum.                             *
2253            *         12)  Issue SIOP command to bootstrap the system.         *
2254            *         13)  When the channel program halts,check (CPVA)<>%100000.*
2255            *              If not, set up ICS and trap to segment #1, STT #44. *
2256            *                                                                  *
2257            ********************************************************************
2258            %0200
2259    0200    LOAD 0090      ADDL     SP4A      1000      ADDL     XR6              clear MSGINT; XR6 := channel talk command
2261    0201         0078 RH   ANDL                RH   RH   ADD  LSL  SP2B SF1      isolate chan #; SP2B := DRT offset; set F1
2263    0202         UBA  ADD  LSR                 B000      ADDL     XR7              shift down channel #; XR7 := IOCL message
2265    0203         0002      ADDL     SP4A              UBA  ADD  LSR  XR4  CLSR     disable ECM; XR4:= shifted down chan#; SR:=0
2267    0204                   JSB  ECM       UNC         SP4A ADD                CCPX  determine memory size; clear MSGINT
2269            ***             initialize memory to HALT %10's                        ***
2270    0205                   ADD      RG   WRX3        XR4  ADD  LSR  SP1B         start address := 0; SP1B := channel#
2272    0206                   JSB  *+1       TEST              ADD       BKX5         interrupt pending?; clear BKX5 for messages
2274    0207           XR3     ADD                       DATA   RG   JSBI *-1  RG   NZRO  write 'HALT 10'; loop back 'til bank is full
2276    0208                   ADD                            RB   BKX3 SUB       BKX4 ZERO  end of bank memory?
2278    0209                   JSB  CLDO      SF1               BKX3 JSBI *-4  BKX3 UNC   continue cold load; no loopback
```

```
RECORD  C S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2281            ************************************************************************
2282            *                                                                      *
2283            *     This module is used during memory initialization to:             *
2284            *                                                                      *
2285            *          1.  Enable/Disable error correction                         *
2286            *          2.  Determine memory size                                   *
2287            *                                                                      *
2288            ************************************************************************
2289            *
2290    020A    ECM  F804      ADDL      RB        0120      ADDL      BNKS           RB := UBA := mem size mask & shift count;
2292                                                                                    BNKS := read & write memory status msg
2293    020B         0400      ADDL      RG              UBA  ADD       CTR  PSHR      RG := read upper word; CTR := UBB.(8:8) + 4
2295    020C         02B8      ADDL                      SP4A ADD            WRS       UBA := command word; UBB := disable err corr
2297    020D         30F8      ADDL      XR3             UBA  ADD            BUSC      XR3 := 'HALT %10''; send message to memory
2299    020E                   ADD             MSGI  RG        ADD            BUSC      Skip if MSGINT; Read upper wrd of mem status
2301    020F                   JSB  *-1        UNC   RG        ADD  LSL       BUSC      Wait for MSGINT; Read lower wrd of mem stat
2303    0210         0090      ADDL      RB              OPB  ADD            POPR      RB := clear MSGINT; UBB := msw mem status wd
2305    0211               UBB ADD                       RB   OPB  RPCA                UBA := msw mem status;
2307                                                                                    UBB := lsw mem status.(0:5)
2308                                                                                    Repeat next line 6 times
2309    0212               UBA ADD   LSL                       UBB  LINK           DCTR CTR0  Shift memsize into UBA
2311    0213         UBA       CAD       RB        RB        ADD                CCPX RSB   RB := # memory banks - 1; Clear MSGINT, rtn
```

```
RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2314           ***************************************************************
2315           *                                                             *
2316           *          CPU REGISTER SAVE MODULE                           *
2317           *                                                             *
2318           *          This module will copy the CPU registers            *
2319           *          to main memory locations %1401-%1422.              *
2320           *                                                             *
2321           *               %1401  -  CHANNEL/DEVICE NUMBER               *
2322           *               %1402  -  X REGISTER                          *
2323           *               %1403  -  DL REGISTER                         *
2324           *               %1404  -  DB-BANK REGISTER                    *
2325           *               %1405  -  DB REGISTER                         *
2326           *               %1406  -  Q REGISTER                          *
2327           *               %1407  -  SM REGISTER                         *
2328           *               %1410  -  S-BANK REGISTER                     *
2329           *               %1411  -  Z REGISTER                          *
2330           *               %1412  -  STATUS REGISTER                     *
2331           *               %1413  -  PB-BANK REGISTER                    *
2332           *               %1414  -  PB REGISTER                         *
2333           *               %1415  -  P REGISTER                          *
2334           *               %1416  -  PL REGISTER                         *
2335           *               %1417  -  CIR REGISTER                        *
2336           *               %1420  -  # OF BANKS                          *
2337           *               %1421  -  CPX1 REGISTER                       *
2338           *               %1422  -  CPX2 REGISTER                       *
2339           *                                                             *
2340           *  It will also write the ucode version number at SYSGLOB EXT %20 *
2341           *                                                             *
2342           ***************************************************************
2343           *
2344   0214    DUMP 034C       ADDL                        ADD       BKX4 CF4B      UBA := %1514; BANK 0 ACCESSES          (2306)
2346   0215            UBA  ADD          WRX4        SP1B ADD                        WRITE TO %1514;UBB=SYS HALT NUMBER     (2306)
2348   0216            UBB  ADD          DATA  0301       ADDL                       WRITE SYS HALT CODE TO %1514;UBB=%1401(2306)
2350   0217            UBB  ADD          WRX4        RH   ADD       XR16           WRITE TO %1401; XRB16=MCDEV#
2352   0218            RH   ADD          DATA  1000       ADDL      XR6            WRITE(DEVICE#); XR6:=CHANNEL TALK COMMAND
2354   0219        X        ADD          DATA  UBA   UBA  ADD  LSL  SP2B           Write (X); SP2B := DRT entry for dump device
2356   021A            DL   ADD          DATA        BNKD ADD                     WRITE(DL); UBB:=DB-BANK
2358   021B            UBB  ADD          DATA             INC       BNKD           WRITE(DB-BANK); BNKD := 1 (for #SAVO)
2360   021C            DB   ADD          DATA  0078  RH   ANDL                     WRITE(DB); CLEAR MOD,DEVICE# ONLY
2362   021D            Q    ADD          DATA        UBB  ADD  LSR  SP1B           WRITE(Q); SHIFT DOWN CHANNEL#
2364   021E            SM   ADD          DATA        BNKS ADD                     WRITE(SM); UBB:=S-BANK
2366   021F            UBB  ADD          DATA  Z          ADD                     WRITE(S-BANK); UBB:=Z REGISTER
2368   0220            UBB  ADD          DATA             ADD       BKX5           WRITE(Z); CLEAR BKX5 FOR MESSAGES
2370   0221            STA  ADD          DATA        BNKP ADD                     WRITE(STATUS); UBB:=PB-BANK
2372   0222            UBB  ADD          DATA        SP1B ADD  LSR  XR4            WRITE(PB-BANK); SHIFT DOWN CHANNEL #
2374   0223            PB   ADD          DATA        UBB  ADD  LSR  SP1B           WRITE(PB); SP1B:=CHANNEL #
2376   0224            P    ADD     RF   DATA  PL         ADD                     WRITE(P); UBB:=PL                      (2334)
2378   0225            UBB  ADD          DATA             ADD                     WRITE(PL)                              (2334)
2380   0226            CIR  ADD          DATA  B000       ADDL      XR7            Write (CIR); XRB7 := IOCLEAR COMMAND
2382   0227                 ADD     SP4A                  JSB  ECM            UNC  SP4A := 0, determine memory size        (2334)
2384   0228            UBA  INC     RB   DATA  7777       ADDL      XR8            Write # of banks to %1420; XRB8 := CCPX code
2386                                                                                  RB := # of banks for CKPB.             (2334)
2387   0229            CPX1 ADD          DATA  27         ADDL      CTR            Write (CPX1); Set for WCS parity off   (2605)
2389   022A            CPX2 ADD          DATA  0011       ADDL                     Write (CPX2); Point to ucode version   (2553)
2391   022B        UBB      ADD          WRX3             ADD       RG             Ucode vers addr -> YREGA,RG=0(sav ctr)(2605)
2393   022C                 ADD     REGN RAR   003A       ADDL      CTR             Get WCS addrs. into RAR               (2553)
```

RECORD NO	C S ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT	
2395																Point CTR at WCS 2nd. quarter	(2553)
2396	022D				ADD						ADD					Let things settle down	(2553)
2398	022E				JSB	*+2		MEDJ		REGN	ADD				RSB	Magic incantation to read WCS	(2553)
2400	022F				ADD						ADD					The read line	(2553)
2402	0230		02FF		ADDL					SREG	ADD					UBA <- PTR to SYSGLOB EXT area	(2553)
2404																Hold data for a little bit	(2553)
2405	0231		UBA		ADD			ROX3		UBB	ADD					Read SYSGLOB EXT base into OPA	(2553)
2407																Hold data some more	(2553)
2408	0232		0210		ADDL					UBB	ADD					UBA <- offset to ucode version word	(2553)
2410	0233		UBA	OPA	ADD			WRX3		UBB	ADD					Set up to write into ucode vers. wd	(2553)
2412	0234			UBB	ADD			DATA	034D		ADDL					Write version out	(2553)

```
RECORD  C S                ********** ALU A **********  ************** ALU B **************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2415            ********************************************************************
2416            *                                                                  *
2417            *       Save new DRT locations, DRT pointers and I/O masks         *
2418            *                                                                  *
2419            *       %1515  -  NIR                                              *
2420            *       %1516  -  DRT0       << This is the data (in bank 1)       *
2421            *       %1517  -  DRT1          that was present at the            *
2422            *       %1520  -  DRT2          locations which will be            *
2423            *       %1521  -  DRT3          overlayed by DRT info >>           *
2424            *       %1522  -  DRT bank         abs(8)                          *
2425            *       %1523  -  DRT offset       abs(9)                          *
2426            *       %1524  -  mask for IMB0    abs(1A)                         *
2427            *       %1525  -  mask for IMB1    abs(1B)                         *
2428            *       %1526  -  mask for IMB2    abs(1C)                         *
2429            *       %1527  -  mask for IMB3    abs(1D)                         *
2430            *                                                                  *
2431            ********************************************************************
2432            *
2433            *                    save overlay locations
2434  0235            UBB  ADD            WRX3              JSL  CKPB          UNC   YREGA := %1515; Check BNKP                (2334)
2436  0236            UBB  ADD            DATA        SP2B  ADD           ROAD UNC   Write NIR; Read DRT0, skip                (2334)
2438  0237       0008      ADDL      SP4A                    ADD           RNOI       SP4A := adr of DRT BANK; inc YRGB, read DRTn
2440  0238                 ADD                     0003 RG   SUBL                ZERO  ; If RG = 3 then exit this loop
2442  0239            OPA  ADD            DATA        RG   JSBI *-2  RG        UNC   Write DRTn; Inc count & loop back
2444            *                    save DRT pointers
2445  023A                 ADD                         SP4A ADD           ROA3       ; Read DRT BANK (abs(8))
2447  023B       001A      ADDL      SP4A                    UBB  INC       ROA3       SP4A := mask 1 adr; Read DRT OFFSET (abs(9))
2449  023C            OPA  ADD            DATA              ADD                     Write (8) to %1522
2451  023D            OPA  ADD            DATA        SP4A  ADD           ROA3 UNC   Write (9) to %1523; Read mask 1
2453            *                    save I/O masks
2454  023E       1111      ADDL      RB                      ADD           RNOI       RB := CCPX mask delta; Increment YRGB, read
2456  023F                 INC                         RG    ADD                ZERO  UBA := 1; If RG = zero then exit this loop
2458  0240            OPA  ADD            DATA  RG    UBA   JSBS *-2  RG        UNC   Write mask n; Decrement count & loop back
2460  0241                 ADD                         XR8   REPC                      ; Repeat clearing of CPX bits
2462  0242       UBB  RB   INC            ZERO  UBB   RB    ADD           CCPX       Exit if CCPX mask = FFFF; Clear CPX1, CPX2
2464            *
2465            ********************************************************************
2466            *                                                                  *
2467            *      The LOAD and DUMP paths join at this point.                 *
2468            *                                                                  *
2469            *          1.  Init DRT pointers to bank 1, address 0.             *
2470            *          2.  Save DRT address in SM, (for use later).            *
2471            *          3.  Call #LTOP to translate logical IMB# to physical    *
2472            *              module# and store it in XRA9.                       *
2473            *                                                                  *
2474            ********************************************************************
2475            *
2476  0243  CLDO 0008      ADDL                              ADD       BKX3           UBA := adr of DRT ptrs; BKX3 := 0
2478  0244            UBA  ADD            WRX3        SP2B  ADD       SM    STFF     YRGA:=8; SM:= DRT ADR OF LOAD/DUMP DEVICE
2480  0245                 INC            DATA        RH    ADD       RG             ABS(8) (DRTBANK):=1; RG:= MCDEV#
2482  0246                 ADD            DATA        UBA   JSL  LTOP BNKD      UNC   ABS(9) (DRTOFFSET):=0; JSB TO MAP MOD #
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2485            ********************************************************************
2486            *                                                                  *
2487            *                  I/O INITIALIZATION, DEVICE IDENTICATION         *
2488            *                                                                  *
2489            *         This module will initialize the channel for the          *
2490            *         cold load / dump device.                                 *
2491            *                                                                  *
2492            *              1)  Issue I/O CLEAR of all channels.                *
2493            *              2)  Initialize the channel for the cold load        *
2494            *                  device.                                         *
2495            *              3)  Identify the type of device.  The ident byte    *
2496            *                  will determine whether the cold load device     *
2497            *                  is a magnetic tape or a disc.                   *
2498            *              4)  Re-initialize the channel for the cold          *
2499            *                  load device.                                    *
2500            *              5)  Issue DEVICE CLEAR.                             *
2501            *                                                                  *
2502            ********************************************************************
2503            *
2504    0247    CLD1 007F RH    ANDL      RH              XR7  JSL  IOMS SP3B      NZRO  RH:=CDEV #;ISSUE IOCL COMMAND
2506    0248         ADD        ADD       SP4A SF1        XR6  JSB  ECM  SP3B      NZRO  ENABLE ERROR CORRECTION MEMORY ;UBB:=1000
2508    0249    0007 ADD        ADDL      SP0A      5400       ADDL      RB            MASK ; RB:=UBB:=MASK TO SET IRQ AND CSRQ
2510    024A    0002 SP2B ADDL  ADDL      SP4A            UBB  JSL  IOMS RG          NZRO  DRT ADDR +2; WRITE TO REGISTER#0
2512    024B    0007 RH   ANDL  ADDL      XR1             RH   JSL  CINT BKX5      NZRO  DEVICE #; INITIALIZE CHANNEL
2514    024C    CRT1      ADDL  ADDL      RF        403E       ADDL                       CLEAR RF FOR DELAY COUNTER                (2635)
2516                                                                                      CHANNEL LISTEN COMMAND                    (2635)
2517    024D         UBB  ADD   ADDL      RG              XR6  JSL  IOMS SP3B      NZRO  RG:=CHANNEL LISTEN; DO IT                 (2635)
2519    024E              ADD   ADDL                405F       ADDL      RG            ; RG:=CHANNEL UNTALK COMMAND              (2635)
2521    024F         UBB  XR1   INC   ADDL      XR1             JSL  IOMS Z       UNC   XR1:=IDENTIFY COMMAND; CHANNEL UNTALK (2635)
2523    0250              XR1   ADD   ADDL      RG              JSL  IOMS BNKS    UNC   RG:=IDENTIFY COMMAND; IDENTIFY DEVICE CMD
2525    0251         C002       ADDL  ADDL      RG              JSL  IOMS CTR     UNC   RG:=GET TWO BYTES FROM DEVICE            (2635)
2527                                                                                      CLEAR CTR FOR RETRIES                     (2635)
2528    0252         0200       ADDL                      1300       ADDL      SP3B          FORMATS TO READ REG 2; WRITE REG 3.       (2635)
2530    0253         0002       ADDL  ADDL      RG              UBA  JSL  IOMS XR7       UNC   REG 3:=2 (DETECT OUT FIFO EMPTY)       (2635)
2532                                                                                      XRB7 := READ REG. 2 FORMAT                (2635)
2533    0254    CRTW            ADD                           RF   JSBI CRTW RF        NZRO  DELAY 13 MSECS.                     (2635)
2535    0255                    ADD                           XR7  JSL  IOMS SP3B      NZRO  READ REG 2                          (2635)
2537    0256              RG    JSB   CRUW      NZRO            CAD       XR8  ICTR          SKIP IF BYTES ARRIVED                     (2635)
2539                                                                                      INCREMENT CTR; XRB8 := INVALID ID         (2635)
2540    0257                    JSB   CRTW      ZERO            JSB  CRT2                CTR0  RETRY READ REG 2 UNTIL CTR = 0        (2635)
2542    0258    CRUW            ADD         RG  CF1             JSL  IOMS SP3B      ZERO  RG:=0; RETURN IDENT BYTE             (2635)
2544    0259         00E5       ADDL                      RG    ADD  RRZ  XR8             SAVE FIRST ID BYTE IN LSB OF XRB8    (2635)
2546    025A                    ADD                           UBA  ADD       CTR           SET CTR FOR XRB101                        (2635)
2548    025B                    ADD         RG  CF1             JSL  IOMS SP3B      ZERO  COMMAND TO READ SECOND BYTE           (2635)
2550    025C                    ADD                           INC       REGN                ;XRB101<>0 FOR NO DCU LOGGING       (2312)
2552    025D                    ADD                     405E  ADDL      RG            UNLISTEN COMMAND TO GIC
2554    025E                    ADD                     XR6   JSL  IOMS SP3B      NZRO
2555    025F                    ADD                     403F  ADDL      RG            UNTALK COMMAND TO DEVICE
2557    0260                    ADD                     XR6   JSL  IOMS SP3B      NZRO
2558    0261         0002 SP2B  ADDL        SP4A              JSL  CINT BKX3      UNC   SP4A:=DRT ADDR +2; REDO INITIALIZE CHANNEL
2560    0262    CRT2 0341       ADDL        XR1       0338    ADDL      BKX7            XR1:=%1501; BKX7:=%1470
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2563            *****************************************************************
2564            *                                                               *
2565            *          BOOTSTRAP SET UP MODULE                              *
2566            *                                                               *
2567            *          This module will set up the bootstrap I/O program    *
2568            *          that microcode builds.                               *
2569            *                                                               *
2570            *                          X = 1   for 1st bootstrap call       *
2571            *                          X = 0   for 2nd bootstrap call       *
2572            *                                                               *
2573            *               1)  Builds the disc or MT I/O program.          *
2574            *               2)  If loading rather than dumping then modify  *
2575            *                   the disc program to seek to sector 2 and    *
2576            *                   read to %7100.                              *
2577            *               3)  Set up the DRT entries for load/dump device *
2578            *               4)  Set interrupt system mask for channel.      *
2579            *               5)  Issue SIOP command for execution of I/O     *
2580            *                   program.                                    *
2581            *                                                               *
2582            *****************************************************************
2583            *
2584    0263          JSB  DSCP      UNC        XR8  JSL  NSDV          NZRO  Copy 13037 disc chan prog or check further
2586    0264  CLDR    XR15 JSB  DMP1 EVEN       BKX7 ADD            WRX3       SKIP LOAD SET UP IF DUMP; YREGB:=%1470
2588    0265          XR1  ADD       WRX3  0E40      ADDL                      YREGA:=%1501; UBB:=%7100
2590    0266               INC  LSL  DATA       UBB  ADD            DATA       SEEK TO SECTOR 2; READ TO %7100
2592    0267  DMP1 0003    ADDL      SP0A            BKX5 ADD       XR20       SP0A:=DRT INDEX; XRB20:=CDEV#
2594    0268       0313    ADDL                      SP2B ADD       XR31 WRD   CP ADDRESS; DRT ADDRESS WRITE TO BANK 1
2596                                                                          XRB31:= DRT0 ADDRESS
2597    0269       0317    ADDL      XR7             UBA  ADD            DATA       CPVA ADDRESS; WRITE(CP ADDRESS)
2599    026A       SP0A SP2B ADD            WRD      UBA  ADD            DATA       DRT+3 ADDRESS; WRITE(CPVA ADDRESS)
2601    026B       3000    ADDL                 002F SP1B SUBL      CTR        UBA:=3000; CTR:=BITMASK# FOR CHANNEL
2603    026C       UBA  UBA ADD  LSL  DATA  9000      ADDL      SP3B       UBA:=C000,WRITE DRT3; SP3B:=SMSK COMMAND
2605    026D                ADD                      REGN JSL  IOMS RG         UNC   ; RG:=CHANNEL MASK, SEND SMSK
2607    026E                INC       X         5000 BKX5 IORL      SP3B       UNC   X:=1 FOR 1ST BOOTSTRAP ; SP3B:= SIOP COMMAND
2609    026F       0007 UBB ANDL      RG                 JSL  IOMS DL         UNC   RG:=DEVICE #; ISSUE SIOP COMMAND
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2612            ************************************************************************
2613            *                                                                      *
2614            *          COLD LOAD AND DUMP SET UP MODULE                            *
2615            *                                                                      *
2616            *          This module will set up cold load running of the            *
2617            *          I/O bootstrap programs.                                     *
2618            *                                                                      *
2619            ************************************************************************
2620            *
2621    0270  CCL  0C00      ADDL                      F379      ADDL      PB              UBA:=GO UNBUSY CMD; SET TIMEOUT FOR 48 SECS
2623    0271                 ADD                       UBA       ADD             BUSC    ; GO UNBUSY
2625    0272  CLLP           JSB  *+1       TEST       XR20 ADD        RH              RESET CPUTIMER; COLDLOAD DEV#
2627    0273                 JSB  CLP       MSGI       DL   INC        DL        NZRO  JSB IF MSG INTERRUPT
2629                                                                                    DO NEXT LINE ONCE IN 2**16 TIMES
2630    0274                 JSB  CLLP      UNC        PB   JSZI CPT0 PB        ZERO  LOOP BACK IF NOT MSGI
2632                                                                                    ;ELSE SYSHALT IF TIMEOUT.
2633    0275  CLP  0200      ADDL     RG                       ADD       CTR   CCTX    RG:=UBA:=%0200; CTR:=0, CTX:=0
2635    0276       0090      ADDL                       UBA   INC  LSL         BUSC    SET CLEAR MESS. INT. FIELD;
2637                                                                                    READ FROM CODE (UBB:=%0402)
2638    0277                 ADD                       UBA   ADD             CCPX    ; Clear message interrupt bit
2640    0278                 ADD            SF1   0040 OPB   ANDL                ZERO  Set F1, Skip if it is an IRQ
2642    0279       X         JSB  LD2       ODD              JSL  CSRQ BKX5     UNC   2ND TIME IF NOT CSRQ AND X ODD; JSB IF CSRQ
2644    027A                 ADD      SPOA  F1HB       RG    ADD  LSL         BUSC    SPOA:=%8000; GO UNBUSY (UBB:=%0400)
2646    027B  CLFN 0007 RH   ANDL     RG              1C00 RH   IORL      SP3B        RG:=DEVICE#; SP3B:=INT. CLEAR MESSAGE
2648    027C           XR7   ADD                      ROX3             JSL  IOMS BKX5     UNC   READ(CPVA);
2650                                                                                    CLEAR INTERRUPT ON DEVICE, BKX5:=0
2651    027D           RB    ADD      RG                       XR6   JSL  IOMS SP3B        NZRO  RG:=%5400 TO ENABLE IRQ AND CSRQ;
2653                                                                                    SP3B:=%1000 (WRITE TO REG# 0 CHAN/DEV 0)
2654    027E       SPOA OPA  JSZS CPAB       NZRO  A401       ADDL      SP1B            IF CHANNEL PROGRAM ABORT THEN TRAP
2656                                                                                    SP1B:=EXTERNAL PROGRAM LABEL
2657    027F                 JSZ  ICS       SF1        XR16 ADD        RH              NO, SET UP ICS, SF1; RH:=MCDEV#
2659    0280                 INC      STA   F1HB  FFF7       ADDL                      STA:=PRIVILEGED, SEG# 1; -9 ON UBB      (CSTX)
2661    0281       UBB  Q    ADD            WRX3  4000       ADDL      XR12            WRITE AT Q-9;SET PHYS MAPPING           (CSTX)
2663    0282                 INC            DATA             ADD       BNKS            Q-9=1 (CST EXTENSN); BANKS=0            (CSTX)
2665    0283           SM    INC            WRS         SM   INC       SM              WRITE PARAMETER; SM:=SM + 1
2667    0284           RH    ADD            DATA             JSL  PCL3 BNKD        UNC   WRITE(CHANNEL/DEVICE); GO SET UP SEGMENT #1
```

```
RECORD  C.S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2670            ********************************************************************
2671            *                                                                  *
2672            *          COLD LOAD AND DUMP BOOTSTRAP MODULE                     *
2673            *                                                                  *
2674            *          This module will perform the bootstrapping of MPE's     *
2675            *          INITIAL and related segments.                           *
2676            *                                                                  *
2677            ********************************************************************
2678            *
2679  0285  LD2        XR15 JSB  LC2        ODD   0080       ADDL      RG              DUMP?; SET LENGTH OF CP PROGRAM
2681  0286       0358       ADDL       SP0A                  JSB  LD   BKX5       UNC   SET ADDRESS FOR READ (%1530);
2683                                                                                   LOAD BOOT, BKX5:=0
2684  0287  LC2  0E40       ADDL       SP0A                  ADD       BKX5            SET ADDRESS FOR READ (%7100); BKX5:=0
2686  0288  LD         UBA  ADD                ROX3  A72E     ADDL      SP3B            READ(CHECKSUM); SP3B:=%123456
2688  0289                  ADD                         XR31 ADD       SP2B            ;SP2B:=DRT0 ADDRESS
2690  028A       SP0A       INC        SP0A  ROB3  RG       CAD       RG        NZRO  READ(PROGRAM; DECREMENT WORD COUNT, DONE?
2692  028B            OPA   ADD        SP4A                  JSB  LDNE        UNC   SP4A:=CHECKSUM; YES!
2694  028C                  JSB  *-2        UNC   SP3B OPB  ADD       SP3B            LOOP BACK; ACCUMULATE SUM IN SP3B
2696  028D  LDNE 1C00 RH   IORL       SP4A        SP3B SP4A XOR                 ZERO  SP4A:=INT; CLEAR MASK; CHECKSUM CORRECT?
2698  028E       0007 RH   ANDL       RG                    JSZ  CPCS        UNC   RG:=DEVICE#; NO, SYSTEM HALT 007
2700  028F            XR15 ADD  LSR        ODD   035B       ADDL      RH              DUMP?; RH:=%1533
2702  0290       0E41       ADDL       RH                    SP4A JSL  IOMS SP3B  NZRO  RH:=%7101; RESET INTERRUPT ON GIC
2704  0291            RB    ADD        RG                    XR6  JSL  IOMS SP3B  NZRO  RG:=%5400 TO ENABLE IRQ AND CSRQ;
2706                                                                                   SP3B:=%1000 (WRITE TO REG# 0 CHAN/DEV 0)
2707  0292       0003 SP2B  ADDL                          SP2B ADD            WRD       UBA:=DRT ADDRESS+3; WRITE TO DRT(0)
2709  0293            UBA   ADD                WRD        RH   ADD            DATA      WRITE TO DRT(3); WRITE CP POINTER
2711  0294                  ADD        X     C000        ADDL                          X:=0; SET DRT CHANNEL RUN BIT MASK
2713  0295            UBB   ADD                DATA  5000 XR20 IORL      SP3B            WRITE TO DRT(3); SET UP SIOP COMMAND
2715  0296       0007 UBB  ANDL       RG                    JSL  IOMS              UNC   UBA:=DEV#; ISSUE SIOP COMMAND
2717  0297                  ADD                               JSB  CCL  DL        UNC   ; DL:=0, PERFORM BOOTSTRAP
```

```
RECORD  C.S.        ********** ALU A **********  ************* ALU B **************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2720          ************************************************************************
2721          *                                                                      *
2722          *          COLD LOAD AND DUMP CHANNEL INITIALIZATION                   *
2723          *                                                                      *
2724          *          This module will perform the initialization and the         *
2725          *          placing on-line of the HPIB channel on the GIC.             *
2726          *                                                                      *
2727          *          1)  Clear system mask.                                      *
2728          *          2)  Clear DRT3 entries.                                     *
2729          *          3)  Read register #1 for system controller flag             *
2730          *              (bit 12).                                               *
2731          *          4)  If not system controller then controller cannot         *
2732          *              be made to be controller-in-charge (system halt 14).    *
2733          *          5)  Enable the channel to respond to indentify              *
2734          *              sequences on the HPIB.                                  *
2735          *          6)  Go on-line to the HPIB and enable HPIB CRC              *
2736          *              capability.                                             *
2737          *          7)  Assert IFC, REN on the HPIB.                            *
2738          *          8)  Wait 100 ms for duration of IFC, REN.                   *
2739          *          9)  Remove IFC, keep REN.                                   *
2740          *         10)  Clear any HPIB interface errors.                        *
2741          *         11)  Check register #1 to verify that channel                *
2742          *              controller is now controller-in-charge.                 *
2743          *         12)  If not controller-in-charge take off-line and           *
2744          *              system halt 21.                                         *
2745          *                                                                      *
2746          ************************************************************************
2747          *
2748  0298  CINT 0080      ADDL      RG          3000 BKX5 IORL      SP3B          RG:=80; SET INIT COMMAND
2750  0299                ADD                           JSL  IOMS         UNC     ; SEND INIT COMMAND
2752  029A       0100     ADDL      SP4A                CAD  LLZ  XR9  CF2       REG#1; TIMER FOR ASSERTING IFC AND REN
2754  029B       0008     ADDL      SP0A                UBA  JSL  IOMS SP3B  NZRO MASK; READ REG#1 FOR SYS. CONTROLLER?
2756  029C       SP0A RG  AND             NZRO 1600          ADDL      RH          SKIP IF SYS. CONTROLLER; WRITE TO REG#6
2758  029D                JSZ  NSYC        UNC  UBB  SP4A ADD           XR13        SYSTEM HALT IF NOT SYSTEM CONTROLLER;REG#7
2760  029E       0008     ADDL      RG                  RH   JSL  IOMS SP3B  NZRO REG#6:=0008; PPOLL RESPONSE
2762  029F                ADD                           8080 ADDL                 ;ASSERT CRC WHILE GOING ON-LINE
2764  02A0           UBB  ADD       RG                  XR13 JSL  IOMS SP3B  NZRO REG#7:=4080; GO ONLINE ON HP-IB
2766  02A1       0030     ADDL      RG                  RH   JSL  IOMS SP3B  NZRO REG#6:=0030; ASSERT IFC, REN ON HP-IB
2768  02A2  CINR 1200     ADDL                          XR9  JSBI CINR XR9   NZRO REG#2; ALLOW 100 USEC TO ASSERT IFC AND REN
2770  02A3       0020     ADDL      RG                  UBA  JSL  IOMS XR9   UNC  REG#6:=0020; REMOVE IFC, KEEP REN
2772  02A4                CAD       RG                  XR9  JSL  IOMS SP3B  NZRO REG#2:=%FFFF; CLEAR ANY HP-IB INTERFACE ERR.
2774  02A5       0010     ADDL      RH                  SP4A JSL  IOMS SP3B  NZRO MASK; READ REG#1, FOR CIC OF HP-IB
2776  02A6       1700     ADDL                     RH   RG   AND              NZRO WRITE TO REG#7; SKIP IF HP-IB CONTROLLER
2778  02A7                ADD       RG                       JSZ  NCON         UNC  REGISTER #7:=0; SYSTEM HALT IF NOT SYS. CONT
2780  02A8       200A     ADDL                          XR6  ADD       SP3B        MASK TO BE SHIFTED LEFT;SP3B:=%1000
2782  02A9       UBA      ADD  LSL  RG                  XR8  JSL  IOMS            ZERO UBA:=!4014,DEVICE CLEAR;SEND MSG IF 792X
2784  02AA                JSB  CRT2        UNC               JSB  CRT1            F1   EXIT ROUTINE
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2787            ********************************************************************
2788            *                                                                  *
2789            *          DISC I/O PROGRAM BUILD MODULE                           *
2790            *                                                                  *
2791            *          This module will build the disc I/O program             *
2792            *          at location %7100.                                      *
2793            *                                                                  *
2794            ********************************************************************
2795            *
2796    02AB    DSCP CLDD       ADDL LBL  SPOA       FFF3      ADDL      RD              SETUP THE MSG POINTER; #OF MSG
2798    02AC         0313       ADDL                           ADD       BKX4            STARTING MEMORY ADDRESS; SET BANK:=0
2800    02AD                    JSB  UNPK      UNC   UBA       ADD             WRX4    GO UNPACK THE MSG & WRITE TO WCS; MEM ADDR
2802    02AE                    JSB  CLDR      UNC             ADD       BKX3            RETURN; SET BKX3:=0
2804    02AF    CLDD 0200 0000 0408 0002                       CON                       WAIT FOR 1st PPOLL
2806    02B0         0000 0000 033E 0200                       CON                       SET FILEMASK FOR DISC
2808    02B1         0000 0408 0002 0000                       CON                       ISSUE READ STATUS COMMAND
2810    02B2         0000 0343 0308 0004                       CON                       READ 4 STATUS BYTES
2812    02B3         0000 0000 0344 0200                       CON                       ISSUE WAIT
2814    02B4         0000 0408 0006 0000                       CON                       ISSUE SEEK COMMAND
2816    02B5         0000 033F 0200 0000                       CON                       ISSUE WAIT
2818    02B6         0408 0002 0000 0000                       CON                       ISSUE READ COMMAND
2820    02B7         0342 0300 0100 0000                       CON                       READ 256 BYTES TO ADDR 0358 (FOR DUMP)
2822    02B8         0000 0358 0200 0000                       CON                       ISSUE WAIT
2824    02B9         0180 0000 FFFF 0F05                       CON                       DONE - INTERRUPT HALT
2826    02BA         0200 0000 0003 0500                       CON                       SEEK COMMAND; READ COMMAND
2828    02BB         0300 0000 0000 0000                       CON                       STATUS COMMAND; STATUS BUFFERS
2830    02BC    UNPK            ADD                            ADD             PSHR      ADJUST RAR
2832    02BD    UNP1 0004       ADDL       RG          0038   ADDL      PL              # OF WCSs; STARTING OF WCS OPTION
2834    02BE    UNP2 SPOA       ADD             WRX3   0027   ADDL      CTR             WCS POINTER; WCS ADDR INHIBIT WCS PARITY
2836    02BF                    ADD        REGN RAR    PL     ADD       CTR             RAR:=YREGA (WCS ADDR); STARTING OF WCS
2838    02C0         RG         CAD        RG          PL     INC       PL              DEC # OF WCS; INC THE WCS POINTER
2840    02C1                    JSB  *+1        MEDJ        REGN ADD                RSB   VBUS:=WCS ADDR WHEN THIS LINE IS IN RANK 1
2842    02C2                    ADD             TEST        SREG ADD                NF2   RESET CLK; SKIP IF FROM COLD LOAD
2844    02C3                    ADD                         UBB  SP1B ADD       SP1B  UNC   IF F2 THEN ADD WCS DATA TO CKSUM, SKIP
2846    02C4         RG         JSB  UNP2       NZRO        RREG      ADD             DATA  LOOPBACK IF NOT FINISHED; WRITE TO MEMORY
2848    02C5         SPOA       INC        SPOA             0800      ADDL                  Inc the msg pointer; UBB := addr of MPL area
2850    02C6         UBA  UBB   SUB                NZRO              ADD                   Skip if new address not start of MPL area
2852    02C7         0B20       ADDL       SPOA                  RD  JSBI UNP1 RD        NZRO  Skip over MPL area; Do again if not done
2854    02C8                    ADD                NZRO              ADD             POPR NF2  Slow down RSB; Pop rtn adr, skip if from
2856                                                                                            cold load
2857    02C9                    ADD                RSB   FEC0 BKX5 ADDL      RG              RETURN; RG := precomputed WCS checksum -
2859                                                                                            MPL area checksum
2860    02CA            XR1     ADD                               ADD
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2862            ********************************************************************
2863            *                                                                  *
2864            *          MAGNETIC TAPE I/O PROGRAM BUILD MODULE                  *
2865            *                                                                  *
2866            *          This module will build the Mag Tape I/O program to      *
2867            *          read the bootstrap I/O program.                         *
2868            *                                                                  *
2869            ********************************************************************
2870            *
2871    02CB    MTCP CLDT      ADDL LBL   SPOA       0001 XR8  SUBL                ZERO  SETUP THE MSG POINTER
2873                                                                                     SKIP NEXT LINE IF ID VALID                (2635)
2874    02CC         FFF4      ADDL       RD                   JSL  CPTO            UNC   NUMBER OF MSG                        (2635)
2876                                                                                     GO TO CPTO                                (2635)
2877    02CD         0313      ADDL                            ADD        BKX4            MEM STARTING ADDR; SET BANK:=0
2879    02CE                   JSB  UNPK      UNC   UBA        ADD             WRX4       GO UNPACK THE MSG; STARTING MEM ADDR
2881    02CF         0070      ADDL                            ADD                        MASK TO SET FLUSH BIT IN CPX2             (SRF6)
2883    02D0                   ADD                       UBA   ADD             CCPX       SET THE BIT                               (SRF6)
2885    02D1         00F0      ADDL                            ADD             ROA3       MASK TO CLEAR FLUSH BIT IN CPX2           (SRF6)
2887                                                                                     DO A READ TO START FLUSHING               (SRF6)
2888    02D2                   ADD                       UBA   ADD             CCPX       CLEAR THE BIT                             (SRF6)
2890    02D3                   JSB  DMP1      UNC             ADD        BKX3            RETURN; BKX3:=0
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2893           ********************************************************************
2894           *                                                                  *
2895           *   CKSM:  Calculates WCS and LUT checksums and compares them      *
2896           *          against the values IML has read from the disc or tape.  *
2897           *          A system halt occurs if an error is found.  CKSM is     *
2898           *          executed upon cold-load or power-on, after the system   *
2899           *          microprogram has been loaded by IML.                    *
2900           *                                                                  *
2901           *          The following data has been left for us by IML:         *
2902           *                                                                  *
2903           *               BKX3 = number of WCS locations                     *
2904           *               BKX5 = WCS checksum (+ %123456)                    *
2905           *               BKX6 = LUT checksum (+ %123456)                    *
2906           *                                                                  *
2907           *   Entered with BNKD = 0                                          *
2908           *                                                                  *
2909           ********************************************************************
2910           *
2911    02D4   CKSM           ADD       SPOA SF2   A72E      ADDL      SP1B            Starting WCS adr (SPOA) := 0, set F2 for
2913                                                                                      #UNPK; SP1B := %123456
2914    02D5                  ADD                                  ADD       BNKP PSHR       ; BNKP := 0 (for #LTCK), Push return address
2916    02D6                  JSB  UNPK       MEDJ  0020 BKX3 SUBL      RD              Med jump; RD := -(# WCS locs) + !20 (# MPL)
2918    02D7       RG   SP1B JSZS WCCE       NZRO  A72E      ADDL      SP1B            Upon rtn RG = exp cksm, SP1B = act cksm,
2920                                                                                      Syshalt if <>; SP1B := %123456
2921           *                    now do LUT checksum
2922    02D8   LTCK           ADD       RC         0300      ADDL      SP1B            LUT adr (RC) := 0; SP1B := %1400
2924    02D9             SP1B ADD       X                    BKX6 ADD       RG              X := %123456; RG := exp LUT cksm
2926    02DA   LCK1      SP1B ADD                  WRD   RC   RC   ADD  LSL              Write adr = %1400; UBB := LUT adr & lsl(2)
2928    02DB       UBB   UBB  ADD  LSL            DATA       SP1B ADD            RONP    UBA := LUT adr & lsl(4), write to %1400;
2930                                                                                      Read LUT adr into NIR (write goes first)
2931    02DC                  ADD                                  ADD                     Wait for read
2933    02DD             RC   INC       RC                   OPB  ADD                     Inc LUT adr; Freeze if NIR is not valid
2935    02DE       X     LUT0 ADD                  1000       ADDL                         Add LUT0 to cksm; UBB := last LUT adr + 1
2937    02DF       UBA   LUT1 ADD       X          RREG RC    JSBS LCK1           NZRO    Add LUT1 to cksm; Loop if LUT adr <> !1000
2939    02E0       UBA   RG   JSZS LTCE       NZRO                ADD                     SYSHLT if expected cksum <> calculated cksum
2941                                                                                      pop return adr; slow down following RSB
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2943            ************************************************************************
2944            *                                                                      *
2945            *   MMAP -- Initialize the MODULE MAP (see #LTOP).                     *
2946            *           Entered from #CLDE or #PON with a bit set in RB            *
2947            *           for every IOA that exists on the CSB, i.e., bit 1          *
2948            *           set for IOA1, bit2 for IOA2, etc.                          *
2949            *                                                                      *
2950            *           If module exists then its module# is stored into the       *
2951            *           module map (MMAP) and a message is sent to that IOA        *
2952            *           to enable its CSRQ and IRQ masks, else a -1 is stored      *
2953            *           at this module's position in MMAP.                         *
2954            *                                                                      *
2955            *           RB should be set up by the DCU on LOAD, PON.               *
2956            *                                                                      *
2957            ************************************************************************
2958            *
2959     02E1   MMAP 0008       ADDL       SP4A       1000       ADDL       SP3B            SP4A := starting module #;
2961                                                                                          SP3B := !1000 (write IMBI reg 0)
2962     02E2         0020      ADDL       RF         00D0       ADDL       CTR             RF := max module#; CTR := REGN# for XRA80
2964     02E3   MAP1             ADD                              ADD        BKX5 SRF        ; BKX5 := 0, set RFLAG
2966     02E4                    CAD        REGN       RB   RB    JSB  MAP2  RB         POS  STORE  -1 INTO MMAP FOR THIS MODULE
2968                                                                                      ;RB:=RB*2, JUMP IF MODULE DOES NOT EXIST
2969     02E5              SP4A  ADD        REGN       5400       ADDL       RG              If module exists then store module# in
2971                                                                                          module map; RG := enable masks command
2972     02E6              SP4A  ADD        XR9                   JSL  IOMS             UNC  XR9 := module #; Go enable CSRQ & IRQ masks
2974     02E7   MAP2 0008 SP4A  ADDL       SP4A            CTRS  INC        CTR  CRF        Inc module #; Inc REGN#, clear RFLAG
2976     02E8                   ADD                        RF    SP4A JSBS MAP1       NZRO ; Do next module if last one not max (2324)
2978     02E9         D180      ADDL       SP0A                     ADD             POPR NZRO SP0A := param &lsr(1) (for #PON);
2980                                                                                          Pop return adr, slow down RSB    (2324)
2981     02EA                   ADD                                 ADD                  RSB  XR12=4000 CONST;RET TO PON OR CLDE (2324)
```

```
RECORD  C.S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL  RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

2984            ********************************************************************
2985            *                                                                  *
2986            *    BFD Cold Load Bootstrap Module                                *
2987            *                                                                  *
2988            ********************************************************************
2989            *
2990    02EB    CBFD 0600 0000 0200 0000                    CON                     SEND OUT DEVICE IDENTIFY; WAIT    <2511>
2992    02EC         0503 0000 0000 003C                    CON                     ISSUE DSJ
2994    02ED         FFF6 003C 0405 0009                    CON                     ;SET MASK
2996    02EE         0009 8000 033F 0200                    CON                     ISSUE STATUS MASK; WAIT
2998    02EF         0000 0502 0000 0000                    CON                     ISSUE DSJ
3000    02F0         0030 FFEA 0405 000D                    CON                     ; SEND WRITE
3002    02F1         000D C000 0343 0200                    CON                     WRITE LENGTH/DISC ADDRESS; WAIT
3004    02F2         0000 030E 0100 0000                    CON                     ISSUE DATA READ
3006    02F3         0000 0358 0200 0000                    CON                     READ IN CHANNEL PROGRAM; WAIT
3008    02F4         0503 0000 0000 001C                    CON                     ISSUE DSJ
3010    02F5         001C 001C 0180 0000                    CON                     ; INT/HALT (NORMAL)
3012    02F6         3E00 0000 0000 0008                    CON                     FILE MASK CMD
3014    02F7         7411 0000 0000 0003                    CON                     MASK OFF INFORM. ERRORS;SECTOR ADDRESS CMD
3016    02F8         1800 0001 0000 000D                    CON                     LENGTH COMMAND; READ STATUS COMMAND   (2330)
3018    02F9    SBFD 0405 0001 0000 C000                    CON                     SEND STATUS REQUEST CMD
3020    02FA         034A 0200 0000 030E                    CON                     WAIT
3022    02FB         0014 0000 0400 0367                    CON                     READ STATUS
3024    02FC         0180 0003 0000 0000                    CON                     INT/HALT (ABNORMAL)
3026    02FD    CLCT      XR15 JSB  DMP1      EVEN  0334      ADDL      BKX7      BRANCH BACK IF DUMP; SET ADDRESS TO %1464
3028    02FE              XR1  ADD            WRX3  0E40      ADDL      BKX7      WRITE TO SECTOR ADDRESS OF CP; UBB  = %7100
3030    02FF              XR7  ADD            DATA        BKX7 ADD           WRX3  WRITE LOAD SECTOR ADDRESS; SET WRITE
3032    0300                   ADD                        BKX7 ADD           DATA  LOOP BACK
3034    0301                   ADD                             JSL  DMP1          UNC
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3036            ************************************************************************
3037            *                                                                      *
3038            *      LTOP - Logical-to-Physical IMB translation.                     *
3039            *                                                                      *
3040            *             Enter with RG.(7:2) = logical IMB#;                      *
3041            *             Exits with XRA9.(10:3) = physical IOA module#.           *
3042            *                                                                      *
3043            *             Bits 7,8 of the 9-bit DRT# used in the I/O               *
3044            *             specify which one of four possible IMBs                  *
3045            *             contain the given CDEV#.  The IMBs are numbered          *
3046            *             0 thru 3, but the IOAs are addressed 1 thru 4.           *
3047            *             This routine will translate the logical IMB# to          *
3048            *             a physical IOA# in XRA9, ready for use with the          *
3049            *             BUSC command.  If the module does not exist then         *
3050            *             the Non-Responding Module (NRM) trap is taken.           *
3051            *                                                                      *
3052            *             The translation is accomplished by using the             *
3053            *             logical IMB# as an index into a MODULE MAPPING           *
3054            *             table that is kept in XRA80-83.  This table is           *
3055            *             set at initialization time to contain the physical       *
3056            *             module #s for those IOAs that exist and a -1 for         *
3057            *             those modules which don't exist.                         *
3058            *                                                                      *
3059            *          Logical       Physical       Table       Extended         *
3060            *           IMB#          IOA#        Contents    Register#           *
3061            *          -------       --------     --------    ----------          *
3062            *            0             1            8          XRA80             *
3063            *            1             2           10          XRA81             *
3064            *            2             3           18          XRA82             *
3065            *            3             4           20          XRA83             *
3066            *                                                                      *
3067            ************************************************************************
3068            *
3069            *
3070  0302  LTOP 0180 RG   ANDL                          ADD       XR8                 Mask out all but logical IMB #; XR8 := 0
3072  0303  L2PF UBA  UBA  ADD  LRZ            00D0      ADDL                          UBA := logical IMB#; UBB := XRA80 REGN#
3074                                                                                    (entry from FLSH)
3075  0304                 ADD                  UBA  UBB  ADD       CTR  SRF          ; CTR := index into module map, set RFLAG to
3077                                                                                    read from set 1
3078  0305                 ADD                            ADD             CRF         ; Clear RFLAG for store into XRA9 set 0
3080  0306            REGN ADD                            ADD                   F4B   UBA := physical mod#; Skip if from #FLSH
3082  0307            UBA  ADD       XR9  RSB        UBA  JSZ  NRM                NEG  XRA9 := physical mod#; If mod# < 0 then take
3084                                                                                    Non-Responding Module trap, else RSB
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3086           ************************************************************************
3087           *                                                                      *
3088           *        Multiple-IMB SET MASK Instruction (SMSK)                      *
3089           *                                                                      *
3090           *        This insruction will take 4 words from the TOS and send       *
3091           *        them to IOA's 1 thru 4 (IMB's 0-3) along with the set         *
3092           *        mask IMB command, and also write them to absolute             *
3093           *        locations 01A - 01D (hex).  For any IOAs not present,         *
3094           *        its mask is NOT sent and its word in mask locations is        *
3095           *        ZEROED.                                                       *
3096           *                                                                      *
3097           *        This algorithm utlizes the MODULE MAP in XRA80-83, set 1      *
3098           *        to determine the existance of each IOA.  ( See #LTOP )        *
3099           *                                                                      *
3100           *        Entered from #SCLK with SR=1, BKX3=0, priv mode checked.      *
3101           *                                                                      *
3102           ************************************************************************
3103           *
3104           *     Adjust TOS so SR is >=4; go BUSY.
3105           *
3106    0308   SMSK 0C02       ADDL                    00D0       ADDL       SP1B             UBA := GO BUSY command; SP1B := REGN# for
3108                                                                                            module map (XRA80)
3109    0309        0003       ADDL       XR0              UBA   REPN       XR1   BUSC 04    XRA0:= SR ADJUST-1
3111                                                                                        ;SEND CMD TO GO BUSY, XRB1:=BUSY CMD
3112    030A        0090       ADDL                        OPB   ADD              DCTR CTR0  UBA:=SMSK CMD (SWAPPED);WAIT HERE TO GO BUSY
3114    030B                   JSZ  IRDN       MSGI        UBA   ADD  SWAB SP3B        SRG2  HANDLE ANY PENDING MSGI; SKIP IF SR>2,
3116                                                                                        SP3B:=9000 (SMSK COMMAND)
3117    030C             XR0   INC        SP0A                    JSZ  PUL2             UNC   SP0A:=4 (LOOPCOUNT);PULL 2 IF SR WAS 1 OR 2
3119    030D        SR   XR0   JSZS PULM        ZERO  001A        ADDL       SP2B             PULL 1 IF SR=3; SP2B:=ABS ADR FOR MASKS
3121    030E                   JSZ  PSHM        SR7         SP1B  ADD        CTR              Empty RG if SR = 7; CTR := module map REGN#
3123           *
3124           *  inner loop:  if IOA exists send mask and write to memory
3125           *               else write 0 to memory.
3126           *
3127    030F   SMK1 SP0A       CAD        SP0A                    ADD        BKX5 SRF         DEC LPCNT;RFLAG:=1 TO READ MODULE MAP
3129    0310             UBA   JSZ  DIM2       NEG         XR1   ADD  LLZ  RH   CRF         IF LPCNT<0 THEN ALL DONE,EXIT THRU #DIM2;
3131                                                                                          RFLG:=0 TO STORE INTO XRA9,SETO,RH:=GOUNBSY
3132    0311             REGN  ADD        XR9   HBF2              ADD              ICTR      XR9:=PHYSICAL MOD#, IF NXM THEN F2=1
3134                                                                                        ;INC INDEX INTO MODULE MAP
3135    0312                   ADD              CCA         SP2B  INC        SP2B             Set CCE for successful completion; SP2B :=
3137                                                                                          mem adr at which to store next mask
3138    0313             SP2B  ADD              WRX3  RA    RA    XSUB       POP              YRGA:=ABS MEM ADR FOR THIS IOA'S MASK
3140                                                                                        IF NXM THEN UBB:=0 ELSE UBB:=MASK
3141    0314             UBB   ADD              DATA        UBB   JSL  IOMS RG        NF2   WRITE 0 OR MASK;RG:=MASK,SEND IF IOA EXISTS
3143    0315                   JSB  SMK1        SF1             ADD                        Loop back and do next IOA, set F1 for #DIM2
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3146          ************************************************************************
3147          *                                                                      *
3148          *       Multiple-IMB READ MASK Instruction (RMSK)                      *
3149          *                                                                      *
3150          *       Pushs absolute locations !01A thru !01D onto TOS.              *
3151          *                                                                      *
3152          *       Empties TOS to 3, reads !01A-01D into RE,RF,RG,RH and          *
3153          *       does 4 EPSHs.  (TOS adjusted to 3 because we will lose         *
3154          *       RD when the pushes are actually done )                         *
3155          *                                                                      *
3156          *       Entered from #RCLK with SR <= 6, BKX3 = 0.                     *
3157          *                                                                      *
3158          ************************************************************************
3159          *
3160  0316  RMSK 0019       ADDL      SP4A        0004       ADDL      SP3B            SP4A:=1ST MASK LOC -1; SP3B:=LOOPCOUNTER
3162  0317        SR   UBB  JSZS ETO3       POS         UBB  ADD       CTR             IF SR-4>=0 THEN EMPTY TOS TO 3;CTR:=RE REGN#
3164  0318             SP4A INC       SP4A ROB3  SP3B       JSZ  NEXT            ZERO  READ MASK, SP4A:=NEXT MASK ADR; NEXT IF DONE
3166  0319                  ADD                  SP3B        CAD       SP3B EPSH       ;DEC LOOPCOUNT, EPSH STACK
3168  031A                  JSB  *-2       UNC         OPB  ADD       REGN ICTR       LOOP BACK; STORE MASK TO TOS; INC CTR
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3171            ************************************************************************
3172            *                                                                      *
3173            *          HALT AND PAUSE INSTRUCTIONS                                 *
3174            *                                                                      *
3175            *          This module will perform a HALT or PAUSE of the             *
3176            *          CPU until an interrupt has occurred.                        *
3177            *                                                                      *
3178            *               1)  IF PAUS THEN SET PAUSE BIT IN CPX2.                *
3179            *               2)  IF HALT THEN SET RUN BIT OFF IN CPX2.              *
3180            *               3)  EMPTY TOS REGISTERS TO MEMORY.                     *
3181            *               4)  WAIT FOR INTERRUPT TO OCCUR.                       *
3182            *                                                                      *
3183            ************************************************************************
3184            *
3185            $LUT INSTR=HALT:0 011 000 011 11, ENTRY=HALT
3186            *
3187    031B    STOP            JSB  *+3        UNC   0E00      ADDL      RH              Skip priv test; UBB := clear RUNFF
3189    031C    HALT 0E00       ADDL      RH                    ADD                 UNC   RH := clear RUNFF mask; Skip
3191            *
3192            $LUT INSTR=PAUS:0 011 000 000 01, ENTRY=PAUS
3193            *
3194    031D    PAUS      STA   JSZ  TRP6       POS   6000      ADDL      RH              Trap if non-priv; RH := set PAUSEFF mask
3196    031E                    JSZ  PSHA       SRNZ            ADD                       Empty TOS if SR <> zero
3198    031F                    JSZ  TIME       UNC   00A4      ADDL      SP2B            JSB to measurement code; SP2B := normal
3200                                                                                        execution accumulator code
3201    0320                    ADD                         RH  REPC            CCPX      ; Set PAUSEFF or clear RUNFF, repeat next
3203                                                                                        line until an interrupt occurs
3204    0321                    ADD             TEST            ADD                       Resume when an interrupt occurs
3206    0322                    JSZ  TIME       UNC   00A3      ADDL      SP2B            JSB to measurement code; SP2B := PAUSE code
3208    0323                    JSZ  IRH        UNC       UBB   ADD             CCPX      Service interrupt; Clear PAUSEFF
```

```
RECORD  C S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3211          ************************************************************************
3212          *                                                                      *
3213          *          Run Mode Interrupt Module                                   *
3214          *                                                                      *
3215          *          This module will handle the following interrupts            *
3216          *          if they occur while in run mode:                            *
3217          *                                                                      *
3218          *                     CPX1(4)   -  Run/Halt Switch Interrupt           *
3219          *                     CPX1(8)   -  DCU Command Interrupt               *
3220          *                     CPX1(11)  -  Memory Breakpoint Interrupt         *
3221          *                                                                      *
3222          *          Entered with SP4A = CPX1 &csl(1)                            *
3223          *                                                                      *
3224          ************************************************************************
3225          *
3226  0324    RMIT 1000 STA  ANDL      SP0A       0100 SP4A ANDL                      Isolate right stackop bit;
3228                                                                                    UBB := DCU interrupt flag
3229  0325         0020 SP4A ANDL                       UBB  JSL  DCMD          NZRO  UBA := memory breakpoint interrupt;
3231                                                                                    Process DCU interrupt if present
3232  0326         0800      ADDL                       UBA  JSL  BKPT          NZRO  UBA := mask to clear R/HSWINT; Handle
3234                                                                                    memory breakpoint interrupt if present
3235  0327                   JSB  STOP      UNC        UBA  ADD            CCPX       Halt; Clear interrupt bit in CPX1
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 3238           ;*********************************************************************
 3239           ;   In the following memory reference instructions, prior to       *
 3240           ;   setting the target fetch flip flop, (STFF), READs using bank    *
 3241           ;   register BNKD in ALUA are switched to use BNKS for Q or S       *
 3242           ;   relative addressing.                                            *
 3243           ;*********************************************************************
 3244           ;
 3245           ;*********************************************************************
 3246           ;    LOAD:  DB, Q and S relative                                    *
 3247           ;*********************************************************************
 3248           ;
 3249           $LUT INSTR=LOAD(DB+):0 100 001 0xx xx, DSPL=8, DB, ENTRY=LWD
 3250           $LUT INSTR=LOAD(DB+):0 100 101 0xx xx, DSPL=8, DB, X, ENTRY=LWD
 3251           $LUT INSTR= LOAD(Q+):0 100 001 10x xx, DSPL=7, Q, ENTRY=LWD
 3252           $LUT INSTR= LOAD(Q+):0 100 101 10x xx, DSPL=7, Q, X, ENTRY=LWD
 3253           $LUT INSTR= LOAD(Q-):0 100 001 110 xx, DSPL=6, Q, F2, ENTRY=LWD
 3254           $LUT INSTR= LOAD(Q-):0 100 101 110 xx, DSPL=6, Q, F2, X, ENTRY=LWD
 3255           $LUT INSTR= LOAD(S-):0 100 001 111 xx, DSPL=6, SM, F2, ENTRY=LWS
 3256           $LUT INSTR= LOAD(S-):0 100 101 111 xx, DSPL=6, SM, F2, X, ENTRY=LWS
 3257           ;
 3258     0328  LWS  SR   UBA  ADD                              ADD                      UBA:=SR + (SM + XC);
 3260     0329  LWD  UBA  DSPL ADSB      RH   ROD   SR   SM    JSB  LPSH SP3B      SR7   RH:=UBA:=(BASE + XC) +/- DSPL, READ
 3262                                                                                    SP3B:=UBB:=S, JSZ IF SR=6
 3263     032A  LWD1 UBA  SM   CAD             EPSH  UBB  UBA   BNDE       CTR       TICB  ; BOUNDS CHECK S>=ADDR, CTR:=S-ADDR,
 3265                                                                                      SKIP NEXT IF ADDR>SM AND S>=ADDR AND
 3266                                                                                      NOT (FSS AND (DB REL ADDR OR TFF))
 3267     032B            OPA  ADD       RH   CCA   RH   DL    BNDE                NEXT  RH:=OPERAND, CCA; BOUNDS CHECK ADDR, NEXT
 3269     032C            REGN ADD       RH   CCA   RH   DL    BNDE                NEXT  RH:=WORD IN TOS, CCA; BOUNDS CHECK, NEXT
 3271           ;
 3272           ;*********************************************************************
 3273           ;    LOAD; Indirect DB, Q, and S relative                           *
 3274           ;*********************************************************************
 3275           ;
 3276           $LUT INSTR=LOAD(DB+):0 100 011 0XX XX,DSPL=8,DB,INDR,ENTRY=LWID
 3277           $LUT INSTR=LOAD(DB+):0 100 111 0XX XX,DSPL=8,DB,X,INDR,ENTRY=LWID
 3278           $LUT INSTR=LOAD(Q+):0 100 011 10X XX,DSPL=7,Q,INDR,ENTRY=LWID
 3279           $LUT INSTR=LOAD(Q+):0 100 111 10X XX,DSPL=7,Q,X,INDR,ENTRY=LWID
 3280           $LUT INSTR=LOAD(Q-):0 100 011 110 XX,DSPL=6,Q,INDR,F2,ENTRY=LWID
 3281           $LUT INSTR=LOAD(Q-):0 100 111 110 XX,DSPL=6,Q,X,INDR,F2,ENTRY=LWID
 3282           $LUT INSTR=LOAD(S-):0 100 011 111 XX,DSPL=6,SM,INDR,F2,ENTRY=LWIS
 3283           $LUT INSTR=LOAD(S-):0 100 111 111 XX,DSPL=6,SM,X,INDR,F2,ENTRY=LWIS
 3284           ;
 3285     032D  LWIS SR   SM   ADD                              ADD                      UBA:=SR + SM;
 3287     032E  LWID UBA  DSPL ADSB      RH   ROD   SR   SM    JSB  LPSH SP3B      SR7   RH:=UBA:=BASE +/- DSPL, READ;
 3289                                                                                    SP3B:=UBB:=(SR + SM), JSZ IF SR=6
 3290     032F       UBA  SM   CAD                   UBB  UBA   BNDE       CTR  TICB      ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
 3292                                                                                      S>=ADDR AND NOT (FSS AND DB REL ADDR)
 3293                                                                                      THEN (S-ADDR) ELSE CODE FOR OPERAND
 3294     0330       XC   DB   ADD                   RH   DL    BNDE            STFF      UBA:=XC + DB; BOUNDS CHECK ADDR>=DL, STFF
 3296     0331       UBA  REGN ADD       RH   ROD   SP3B       JSB  LWD1            UNC   RH:=UBA:=TARGET ADDR, READ; UBA:=S, JSB
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3299          ***************************************************************
3300          *    LOAD; P relative                                           *
3301          ***************************************************************
3302          *
3303          $LUT INSTR=LOAD(P+):0 100 000 0XX XX,DSPL=8,P,ENTRY=LWP
3304          $LUT INSTR=LOAD(P+):0 100 100 0XX XX,DSPL=8,X,P,ENTRY=LWP
3305          $LUT INSTR=LOAD(P-) 0 100 000 1XX XX,DSPL=8,P,F2,ENTRY=LWP
3306          $LUT INSTR=LOAD(P-).0 100 100 1XX XX,DSPL=8,X,F2,P,ENTRY=LWP
3307          *
3308    0332  LWP   UBA  DSPL ADSB      RH   ROP             JSZ  PSHM          SR7   RH:=UBA:=(P + XC) +/- DSPL, READ;
3310                                                                                   JSZ IF SR=6
3311    0333  LWP1 UBA   PB   BNDE           EPSH  PL   UBA  BNDE                     BOUNDS CHECK ADDR >= PB, EPSH;
3313                                                                                   BOUNDS CHECK PL >= ADDR
3314    0334  LWP2       OPA  ADD        RH   CCA             ADD                NEXT  RH:=OPERAND, CCA; NEXT
3316          *
3317          ***************************************************************
3318          *    LOAD; Indirect P relative                                  *
3319          ***************************************************************
3320          *
3321          $LUT INSTR=LOAD(P+):0 100 010 0XX XX,DSPL=8,INDR,P,ENTRY=LWIP
3322          $LUT INSTR=LOAD(P+):0 100 110 0XX XX,DSPL=8,X,INDR,P,ENTRY=LWIP
3323          $LUT INSTR=LOAD(P-):0 100 010 1XX XX,DSPL=8,INDR,F2,P,ENTRY=LWIP
3324          $LUT INSTR=LOAD(P-):0 100 110 1XX XX,DSPL=8,X,INDR,F2,P,ENTRY=LWIP
3325          *
3326    0335  LWIP UBA  DSPL ADSB      RH   ROP             JSZ  PSHM          SR7   RH:=UBA:=P +/- DSPL, READ; JSZ IF SR=6
3328    0336       XC   UBA  ADD                      UBA  PB   BNDE                  UBA:=XC + INDIRECT CELL ADDR;
3330                                                                                   BOUNDS CHECK INDIRECT CELL ADDR >= PB
3331    0337       UBA  OPA  JSB  LWP1     ROP   PL   RH   BNDE                     READ XC + CELL ADDR + CELL CONTENTS, JSB;
3333                                                                                   BOUNDS CHECK PL >= INDIRECT CELL ADDR
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3335            ************************************************************************
3336            *     LDX; DB, Q, and S relative                                       *
3337            ************************************************************************
3338            *
3339            $LUT INSTR=LDX(DB+):1 011 001 OXX XX,DSPL=8,DB,ENTRY=LXD
3340            $LUT INSTR=LDX(DB+):1 011 101 OXX XX,DSPL=8,DB,X,ENTRY=LXD
3341            $LUT INSTR=LDX(Q+):1 011 001 10X XX,DSPL=7,Q,ENTRY=LXD
3342            $LUT INSTR=LDX(Q+):1 011 101 10X XX,DSPL=7,Q,X,ENTRY=LXD
3343            $LUT INSTR=LDX(Q-):1 011 001 110 XX,DSPL=6,Q,F2,ENTRY=LXD
3344            $LUT INSTR=LDX(Q-):1 011 101 110 XX,DSPL=6,Q,F2,X,ENTRY=LXD
3345            $LUT INSTR=LDX(S-):1 011 001 111 XX,DSPL=6,SM,F2,ENTRY=LXS
3346            $LUT INSTR=LDX(S-):1 011 101 111 XX,DSPL=6,SM,F2,X,ENTRY=LXS
3347            *
3348    0338    LXS   SR   UBA  ADD                              ADD                    UBA:=SR + (SM + XC);
3350    0339    LXD   UBA  DSPL ADSB       RH    ROD    SR   SM  ADD       SP3B       RH:=UBA:=(BASE + XC) +/- DSPL, READ
3352                                                                                   SP3B:=UBB:=S
3353    033A    LXD1  UBA  SM   JSBC *+1         MEDJ   UBB  UBA BNDE      CTR  TICB  MEDIUM JSB TO WAIT ONE CLOCK;
3355                                                                                   BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3356                                                                                     S>=ADDR AND NOT (FSS AND (DB REL ADDR OR
3357                                                                                     TFF)) THEN (S-ADDR) ELSE CODE FOR OPERAND
3358    033B                REGN ADD       X     CCA    RH   DL  BNDE                  X:=OPERAND, CCA; BOUNDS CHECK ADDR
3360    033C                     ADD                              ADD           NEXT   ; NEXT AFTER WAITING ONE CLOCK
3362            *
3363            ************************************************************************
3364            *     LDX; Indirect DB, Q, and S relative                              *
3365            ************************************************************************
3366            *
3367            $LUT INSTR=LDX(DB+):1 011 011 OXX XX,DSPL=8,DB,INDR,ENTRY=LXID
3368            $LUT INSTR=LDX(DB+):1 011 111 OXX XX,DSPL=8,DB,X,INDR,ENTRY=LXID
3369            $LUT INSTR=LDX(Q+):1 011 011 10X XX,DSPL=7,Q,INDR,ENTRY=LXID
3370            $LUT INSTR=LDX(Q+):1 011 111 10X XX,DSPL=7,Q,X,INDR,ENTRY=LXID
3371            $LUT INSTR=LDX(Q-):1 011 011 110 XX,DSPL=6,Q,INDR,F2,ENTRY=LXID
3372            $LUT INSTR=LDX(Q-):1 011 111 110 XX,DSPL=6,Q,X,INDR,F2,ENTRY=LXID
3373            $LUT INSTR=LDX(S-):1 011 011 111 XX,DSPL=6,SM,INDR,F2,ENTRY=LXIS
3374            $LUT INSTR=LDX(S-):1 011 111 111 XX,DSPL=6,SM,X,INDR,F2,ENTRY=LXIS
3375            *
3376    033D    LXIS SR   SM   ADD                              ADD                    UBA:=SR + SM;
3378    033E    LXID UBA  DSPL ADSB       RH    ROD    SR   SM  ADD       SP3B       RH:=UBA:=BASE +/- DSPL, READ;
3380                                                                                   SP3B:=UBB:=(SR + SM)
3381    033F        UBA  SM   CAD                        UBB  UBA BNDE      CTR  TICB  ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3383                                                                                     S>=ADDR AND NOT (FSS AND DB REL ADDR)
3384                                                                                     THEN (S-ADDR) ELSE CODE FOR OPERAND
3385    0340        XC   DB   ADD                        RH   DL  BNDE           STFF   UBA:=XC + DB; BOUNDS CHECK ADDR>=DL, STFF
3387    0341        UBA  REGN ADD       RH    ROD    SP3B      JSB  LXD1            UNC  RH:=UBA:=TARGET ADDR, READ; UBA:=S, JSB
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3390          ************************************************************************
3391          *    LDX; P relative                                                   *
3392          ************************************************************************
3393          *
3394          $LUT INSTR=LDX(P+):1 011 000 0XX XX,DSPL=8,P,ENTRY=LXP
3395          $LUT INSTR=LDX(P+):1 011 100 0XX XX,DSPL=8,X,P,ENTRY=LXP
3396          $LUT INSTR=LDX(P-):1 011 000 1XX XX,DSPL=8,P,F2,ENTRY=LXP
3397          $LUT INSTR=LDX(P-):1 011 100 1XX XX,DSPL=8,X,F2,P,ENTRY=LXP
3398          *
3399    0342  LXP  UBA  DSPL ADSB      RH   ROP             ADD                       RH:=UBA:=(P + XC) +/- DSPL, READ;
3401    0343  LXP1 UBA  PB   BNDE                    PL   UBA  BNDE                  BOUNDS CHECK ADDR >= PB; CHECK PL >= ADDR
3403    0344       UBA  OPA  ADD       X    CCA             JSZ  NEXT           UNC  X:=OPERAND, CCA;
3405                                                                                 JSB TO WAIT ONE CLOCK FOR NEXT
3406          *
3407          ************************************************************************
3408          *    LDX; Indirect P relative                                          *
3409          ************************************************************************
3410          *
3411          $LUT INSTR=LDX(P+):1 011 010 0XX XX,DSPL=8,INDR,P,ENTRY=LXIP
3412          $LUT INSTR=LDX(P+):1 011 110 0XX XX,DSPL=8,X,INDR,P,ENTRY=LXIP
3413          $LUT INSTR=LDX(P-):1 011 010 1XX XX,DSPL=8,INDR,F2,P,ENTRY=LXIP
3414          $LUT INSTR=LDX(P-):1 011 110 1XX XX,DSPL=8,X,INDR,F2,P,ENTRY=LXIP
3415          *
3416    0345  LXIP UBA  DSPL ADSB      RH   ROP             ADD                       RH:=UBA:=P +/- DSPL, READ;
3418    0346       XC   UBA  ADD                       UBA  PB   BNDE                UBA:=XC + INDIRECT CELL ADDR;
3420                                                                                 BOUNDS CHECK INDIRECT CELL ADDR >= PB
3421    0347       UBA  OPA  JSB  LXP1      ROP   PL   RH   BNDE                    READ XC + CELL ADDR + CELL CONTENTS, JSB;
3423                                                                                 BOUNDS CHECK PL >= INDIRECT CELL ADDR
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3425          ***************************************************************
3426          *    LDD; DB, Q, and S relative                                 *
3427          ***************************************************************
3428          *
3429          $LUT INSTR=LDD(DB+):1 101 001 0XX XX,DSPL=8,DB,ENTRY=LDD
3430          $LUT INSTR=LDD(DB+):1 101 101 0XX XX,DSPL=8,DB,XSL,ENTRY=LDD
3431          $LUT INSTR=LDD(Q+):1 101 001 10X XX,DSPL=7,Q,ENTRY=LDD
3432          $LUT INSTR=LDD(Q+):1 101 101 10X XX,DSPL=7,Q,XSL,ENTRY=LDD
3433          $LUT INSTR=LDD(Q-):1 101 001 110 XX,DSPL=6,Q,F2,ENTRY=LDD
3434          $LUT INSTR=LDD(Q-):1 101 101 110 XX,DSPL=6,Q,F2,XSL,ENTRY=LDD
3435          $LUT INSTR=LDD(S-):1 101 001 111 XX,DSPL=6,SM,F2,ENTRY=LDS
3436          $LUT INSTR=LDD(S-):1 101 101 111 XX,DSPL=6,SM,F2,XSL,ENTRY=LDS
3437          *
3438  0348    LDS   SR   UBA  ADD                          ADD                        UBA:=SR + (SM + XC);
3440  0349    LDD   UBA  DSPL ADSB       RH   ROD   SR   SM   JSB  LDPH SP3B       SRG5  RH:=UBA:=(BASE + XC) +/- DSPL, READ;
3442                                                                                    SP3B:=UBB:=S, JSZ IF SR>4
3443  034A    LDD1  UBA        INC        SPOA ROBD       UBA  ADD        RG   EPSH      SPOA:=ADDR OF NEXT WORD, READ INTO OPB;
3445                                                                                    RG:=ADDR OF FIRST WORD, EPSH
3446  034B              OPA  ADD        RH   CCA   UBA  SM   CAD                 EPSH CRRY  RH:=FIRST WORD, CCA; SKIP NEXT IF ADDR OF
3448                                                                                      SECOND WORD > SM, EPSH
3449  034C        SPOA DL   BNDG                         OPB  ADD        RG   DCC  NEXT  BOUNDS CHECK ADDR OF SECOND WORD > DL;
3451                                                                                    RG:=SECOND WORD, DCC, NEXT
3452  034D        RG   SM   CAD                  SP3B RG   BNDE       CTR  TICB      ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3454                                                                                      S>=ADDR AND NOT (FSS AND (DB REL ADDR OR
3455                                                                                      TFF)) THEN (S-ADDR) ELSE CODE FOR OPERAND
3456  034E        RREG SM   SUB                  SP3B SREG BNDG       CTR  TICB      ; BOUNDS CHECK S>ADDR, CTR:=IF ADDR>=SM AND
3458                                                                                      S>ADDR AND NOT (FSS AND (DB REL ADDR OR
3459                                                                                      TFF)) THEN (S-ADDR-1)
3460                                                                                      ELSE CODE FOR OPERAND
3461  034F              REGN ADD        RH   CCA                ADD                   RH:=FIRST WORD, CCA;
3463  0350                   ADD             NEXT       REGN ADD        RG   DCC      NEXT; RG:=SECOND WORD, DCC
3465          *
3466          ***************************************************************
3467          *    LDD, Indirect DB, Q, and S relative                        *
3468          ***************************************************************
3469          *
3470          $LUT INSTR=LDD(DB+):1 101 011 0XX XX,DSPL=8,DB,INDR,ENTRY=LDID
3471          $LUT INSTR=LDD(DB+):1 101 111 0XX XX,DSPL=8,DB,XSL,INDR,ENTRY=LDID
3472          $LUT INSTR=LDD(Q+):1 101 011 10X XX,DSPL=7,Q,INDR,ENTRY=LDID
3473          $LUT INSTR=LDD(Q+):1 101 111 10X XX,DSPL=7,Q,XSL,INDR,ENTRY=LDID
3474          $LUT INSTR=LDD(Q-):1 101 011 110 XX,DSPL=6,Q,INDR,F2,ENTRY=LDID
3475          $LUT INSTR=LDD(Q-):1 101 111 110 XX,DSPL=6,Q,XSL,INDR,F2,ENTRY=LDID
3476          $LUT INSTR=LDD(S-):1 101 011 111 XX,DSPL=6,SM,INDR,F2,ENTRY=LDIS
3477          $LUT INSTR=LDD(S-):1 101 111 111 XX,DSPL=6,SM,XSL,INDR,F2,ENTRY=LDIS
3478          *
3479  0351    LDIS  SR   SM   ADD                          ADD                        UBA:=SR + SM;
3481  0352    LDID  UBA  DSPL ADSB       RH   ROD   SR   SM   JSB  LDPH SP3B       SRG5  RH:=UBA:=BASE +/- DSPL, READ;
3483                                                                                    SP3B:=UBB:=S, JSZ IF SR>4
3484  0353        UBA  SM   CAD                  UBB  UBA  BNDE       CTR  TICB      ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3486                                                                                      S>=ADDR AND NOT (FSS AND DB REL ADDR)
3487                                                                                      THEN (S-ADDR) ELSE CODE FOR OPERAND
3488  0354        XC   DB   ADD                  RH   DL   BNDE            STFF       UBA:=XC + DB; BOUNDS CHECK ADDR >= DL, STFF
3490  0355        UBA  REGN ADD        RH   ROD   SP3B       JSB  LDD1           UNC   RH:=UBA:=(XC+DB) + INDIRECT CELL, READ;
3492                                                                                    UBB:=S, JSB
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3494            ************************************************************
3495            *    LDPP and LDPN                                         *
3496            ************************************************************
3497            *
3498            $LUT INSTR=LDPP:0 011 100 0XX XX,DSPL=8,P,ENTRY=LDP
3499            $LUT INSTR=LDPN:0 011 100 1XX XX,DSPL=8,P,F2,ENTRY=LDP
3500            *
3501    0356  LDP  UBA  DSPL ADSB      RH   ROP            JSZ  PSM2          SRG5  RH:=UBA:=P +/- DSPL, READ; JSZ IF SR>5
3503    0357  LDP1 UBA  PB   BNDE           EPSH  UBA       INC       RH   ROP       BOUNDS CHECK ADDR >= PB, EPSH;
3505                                                                               RH:=UBB:=SECOND WORD ADDR, READ
3506    0358             OPA  ADD       RH   CCA   PL   UBB  BNDE           EPSH    RH:=FIRST WORD, CCA;
3508                                                                               BOUNDS CHECK PL >= SECOND ADDR, EPSH
3509    0359       RH    PB   BNDG                       OPB  ADD       RG   DCC  NEXT  BOUNDS CHECK SECOND ADDR > PB;
3511                                                                               RG:=SECOND WORD, DCC, NEXT
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3513            ********************************************************************
3514            *    LDB; DB, Q, and S relative                                    *
3515            ********************************************************************
3516            *
3517            $LUT INSTR=LDB(DB+):1 101 000 0XX XX,DSPL=8,DB,ENTRY=LBD
3518            $LUT INSTR=LDB(DB+):1 101 100 0XX XX,DSPL=8,DB,XSR,ENTRY=LBD
3519            $LUT INSTR=LDB(Q+):1 101 000 10X XX,DSPL=7,Q,ENTRY=LBQ
3520            $LUT INSTR=LDB(Q+):1 101 100 10X XX,DSPL=7,Q,XSR,ENTRY=LBQ
3521            $LUT INSTR=LDB(Q-):1 101 000 110 XX,DSPL=6,Q,F2,ENTRY=LBQ
3522            $LUT INSTR=LDB(Q-):1 101 100 110 XX,DSPL=6,Q,F2,XSR,ENTRY=LBQ
3523            $LUT INSTR=LDB(S-):1 101 000 111 XX,DSPL=6,SM,F2,ENTRY=LBS
3524            $LUT INSTR=LDB(S-):1 101 100 111 XX,DSPL=6,SM,F2,XSR,ENTRY=LBS
3525            *
3526    035A    LBS   SR   UBA  ADD                      ADD                        UBA:=SR + (SM + XC);
3527    035B    LBQ   UBA  DSPL ADSB      RH   ROBS  SR   SM   JSB  LBPH SP3B     SR7   RH:=UBA:=(BASE + XC) +/- DSPL, READ INTO B;
3530                                                                                     SP3B:=UBB:=SR + SM, JSB IF SR = 7
3531    035C          UBA  SM   CAD                      UBB  UBA  SUB       CTR  TICB CRRY  ; CTR:=IF ADDR>SM AND S>=ADDR THEN
3533                                                                                       (S-ADDR) ELSE CODE FOR OPERAND,
3534                                                                                       SKIP IF S >= ADDR
3535    035D          RH   DL   JSBS LB32       NCRY            JSB  LB32           UNC   JSB IF NOT (ADDR >= DL);
3537                                                                                     JSB IF NOT (S >= ADDR)
3538    035E          XC         ADD            ODD       REGN ADD  LRZ       EPSH       SKIP NEXT IF RIGHT BYTE;
3540                                                                                     UBB =LEFT BYTE, EPSH
3541    035F                     ADD            NEXT      UBB  ADD       RH   CCB       NEXT IF LEFT BYTE; NEW (S):=LEFT BYTE, CCB
3543    0360                     ADD            NEXT      REGN ADD  RRZ  RH   CCB       NEXT; NEW (S):=RIGHT BYTE, CCB
3545    0361  LBD    UBA  DSPL ADSB      RH   ROBD  SR   SM   JSB  LBPH SP3B     SR7   RH:=UBA:=(BASE + XC) +/- DSPL, READ INTO B;
3547                                                                                     SP3B:=UBB:=S, JSB IF SR = 7
3548    0362          DB   ADD       SPOA       UBA  DL   SUB            TICB NCRY      SPOA:=DB;
3550                                                                                     SKIP IF NOT (ADDR >= DL), TICB CAUSES CTR
3551                                                                                       TO GET %17 (CODE FOR OPERAND)
3552    0363          RH   SM   JSBC LBD1       FSS  SP3B RH   SUB       CTR  TICB CRRY  JSB IF SPLIT BANKS;
3554                                                                                     CTR:=IF ADDR>SM AND S>=ADDR AND
3555                                                                                       NOT (FSS AND (DB REL ADDR OR TFF)) THEN
3556                                                                                       (S-ADDR) ELSE CODE FOR OPERAND,
3557                                                                                       SKIP IF S >= ADDR
3558    0364          SPOA DL   SUB                      Z    DB   JSBS LB32      TICB  ; JSB IF NOT (DL <= ADDR <= S) AND NOT FSS
3560                                                                                       AND (DB >= DL) AND (Z >= DB)
3561                                                                                       (JSB IF NOT (DL <= ADDR <= S) AND NOT
3562                                                                                       SPLIT STACK)
3563    0365  LBD1   XC         ADD            ODD       REGN ADD  LRZ       EPSH       SKIP NEXT IF RIGHT BYTE;
3565                                                                                     UBB =LEFT BYTE, EPSH
3566    0366  LBDL             ADD            NEXT      UBB  ADD       RH   CCB       NEXT IF LEFT BYTE; NEW (S):=LEFT BYTE, CCB
3568    0367  LBDR             ADD            NEXT      REGN ADD  RRZ  RH   CCB       NEXT; NEW (S):=RIGHT BYTE, CCB
3570    0368  LB32             ADD                      8000 RH   ADDL              ; UBB:=32K + ADDR
3572    0369                UBB  ADD       RH   ROBD  UBB  DL   BNDE            TICB  RH:=ADDR, READ; BOUNDS CHECK ADDR >= DL
3574                                                                                     TICB TO SLOW DOWN RSB (ALWAYS FALSE)
3575    036A          UBA  SM   CAD                      SP3B UBA  BNDE      CTR  TICB RSB   ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3577                                                                                       S>=ADDR AND NOT (FSS AND DB REL ADDR) THEN
3578                                                                                       (S-ADDR) ELSE CODE FOR NOP, MEDIUM RSB
```

```
RECORD    C.S.          ********** ALU A **********  ********** ALU B **********
  NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3580            ***************************************************************
3581            *    LDB; Indirect DB, Q, and S relative                        *
3582            ***************************************************************
3583            *
3584            $LUT INSTR=LDB(DB+):1 101 010 0XX XX,DSPL=8,DB,INDR,ENTRY=LBID
3585            $LUT INSTR=LDB(DB+):1 101 110 0XX XX,DSPL=8,DB,X,INDR,ENTRY=LBID
3586            $LUT INSTR=LDB(Q+):1 101 010 10X XX,DSPL=7,Q,INDR,ENTRY=LBID
3587            $LUT INSTR=LDB(Q+):1 101 110 10X XX,DSPL=7,Q,X,INDR,ENTRY=LBID
3588            $LUT INSTR=LDB(Q-):1 101 010 110 XX,DSPL=6,Q,INDR,F2,ENTRY=LBID
3589            $LUT INSTR=LDB(Q-):1 101 110 110 XX,DSPL=6,Q,X,INDR,F2,ENTRY=LBID
3590            $LUT INSTR=LDB(S-):1 101 010 111 XX,DSPL=6,SM,INDR,F2,ENTRY=LBIS
3591            $LUT INSTR=LDB(S-):1 101 110 111 XX,DSPL=6,SM,X,INDR,F2,ENTRY=LBIS
3592            *
3593    036B    LBIS SR   SM   ADD                          ADD                      UBA:=SR + SM;
3595    036C    LBID UBA  DSPL ADSB      RH   ROD   SR   SM   JSB  LPSH SP3B      SR7  RH:=UBA:=(BASE + XC) +/- DSPL, READ;
3597                                                                                SP3B:=UBB:=SR + SM, JSB IF SR=7
3598    036D         UBA  SM   CAD                    UBB  UBA  BNDE      CTR  TICB      ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3600                                                                                  S>=ADDR AND NOT (FSS AND DB REL ADDR)
3601                                                                                  THEN (S-ADDR) ELSE CODE FOR OPERAND
3602    036E              DB   ADD            NFSS Z    DB   SUB            STFF NCRY UBA:=DB, SKIP IF NOT SPLIT BANKS;
3604                                                                                STFF, SKIP IF NOT (Z >= DB)
3605    036F         XC   REGN ADD  LSR       CF1  UBA  DL   SUB            CTF1     UBA:=(XC + INDIRECT CELL) & LSR(1);
3607                                                                                F1:=NOT SPLIT BANKS AND (Z >= DB) AND
3608                                                                                  (DB >= DL)
3609    0370         XC   REGN CSR            HBF2 UBA  DB   ADD        RG   ROD      SET F2 IF EFFECTIVE BYTE ADDRESS IS ODD;
3611                                                                                RG:=ABSOLUTE WORD ADDR, READ
3612    0371         UBB  SM   JSBC LBI1      NF1  SP3B UBB  SUB        CTR  TICB CRRY JSB IF SPLIT STACK;
3614                                                                                CTR:=IF ADDR>SM AND S>=ADDR AND NOT SPLIT
3615                                                                                  BANKS THEN (S-ADDR) ELSE CODE FOR OPERAND
3616    0372         RG   DL   JSBS LI32      NCRY             JSB  LI32           UNC  JSB IF NOT (ADDR >= DL);
3618                                                                                JSB IF NOT (S >= ADDR)
3619    0373    LBI1 RH   DL   BNDE           F2        REGN ADD  LRZ       EPSH     BOUNDS CHECK INDIRECT CELL >= DL,
3621                                                                                  SKIP IF RIGHT BYTE;
3622                                                                                UBB:=LEFT BYTE, EPSH
3623    0374                   ADD            NEXT      UBB  ADD       RH   CCB      NEXT: NEW (S):=LEFT BYTE, CCB
3625    0375                   ADD            NEXT      REGN ADD  RRZ  RH   CCB      NEXT: NEW (S):=RIGHT BYTE, CCB
3627    0376    LI32           ADD                 8000 RG   ADDL                    ; UBB:=32K + ADDR
3629    0377              UBB  ADD       RG   ROBD UBB  DL   BNDE                 TICB RG:=ADDR, READ; BOUNDS CHECK ADDR >= DL,
3631                                                                                  TICB TO SLOW DOWN RSB (ALWAYS FALSE)
3632    0378         UBA  SM   CAD                    SP3B UBA  BNDE      CTR  TICB RSB  ; BOUNDS CHECK S>=ADDR, CTR:= IF ADDR>SM AND
3634                                                                                  S>=ADDR AND NOT (FSS AND DB REL ADDR) THEN
3635                                                                                  (S-ADDR) ELSE CODE FOR NOP, MEDIUM RSB
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3637            ***************************************************************"
3638            *    LRA; Q and S relative                                      "
3639            ***************************************************************"
3640            *
3641            $LUT INSTR=LRA(Q+):1 111 001 10X XX,DSPL=7,Q,ENTRY=LAQ
3642            $LUT INSTR=LRA(Q+):1 111 101 10X XX,DSPL=7,Q,X,ENTRY=LAQ
3643            $LUT INSTR=LRA(Q-):1 111 001 110 XX,DSPL=6,Q,F2,ENTRY=LAQ
3644            $LUT INSTR=LRA(Q-):1 111 101 110 XX,DSPL=6,Q,F2,X,ENTRY=LAQ
3645            $LUT INSTR=LRA(S-):1 111 001 111 XX,DSPL=6,SM,F2,ENTRY=LAS
3646            $LUT INSTR=LRA(S-):1 111 101 111 XX,DSPL=6,SM,F2,X,ENTRY=LAS
3647            *
3648    0379    LAS  SR   UBA  ADD                              ADD                    UBA:=SR + (SM + XC);
3650    037A    LAQ  UBA  DSPL ADSB      RH                     JSZ  PSHM       SR7   RH:=UBA:=(BASE + XC) +/- DSPL;
3652                                                                                  JSB IF SR = 7
3653    037B                   JSZ  BNDV      NZRO  UBA  DB     SUB       RH    EPSH  JSB for BNDV if split banks;
3655                                                                                    New (S) := effective addr - DB, push
3656                                                                                  ******************************************
3657                                                                                  *    TEMP:  DON'T DO BOUNDS CHECK UNTIL  *
3658                                                                                  *           SOFTWARE IS FIXED - 12/11/81  *
3659                                                                                  ******************************************
3660    037C                   ADD                              ADD              NEXT  ; NEXT
3662            *
3663            ***************************************************************
3664            *    LRA; DB relative                                         *
3665            ***************************************************************
3666            *
3667            $LUT INSTR=LRA(DB+):1 111 001 0XX XX,DSPL=8,DB,ENTRY=LAD
3668            $LUT INSTR=LRA(DB+):1 111 101 0XX XX,DSPL=8,DB,X,ENTRY=LAD
3669            *
3670    037D    LAD  XC   DSPL ADD       RH                     JSZ  PSHM       SR7   RH:=XC + DSPL; JSB IF SR = 7
3672    037E                   ADD            EPSH              JSZ  NEXT       UNC   EPSH; NEXT
3674            *
3675            ***************************************************************
3676            *    LRA; Indirect DB, Q, and S relative                      *
3677            ***************************************************************
3678            *
3679            $LUT INSTR=LRA(DB+):1 111 011 0XX XX,DSPL=8,DB,INDR,ENTRY=LAID
3680            $LUT INSTR=LRA(DB+):1 111 111 0XX XX,DSPL=8,DB,X,INDR,ENTRY=LAID
3681            $LUT INSTR=LRA(Q+):1 111 011 10X XX,DSPL=7,Q,INDR,ENTRY=LAID
3682            $LUT INSTR=LRA(Q+):1 111 111 10X XX,DSPL=7,Q,X,INDR,ENTRY=LAID
3683            $LUT INSTR=LRA(Q-):1 111 011 110 XX,DSPL=6,Q,INDR,F2,ENTRY=LAID
3684            $LUT INSTR=LRA(Q-):1 111 111 110 XX,DSPL=6,Q,X,INDR,F2,ENTRY=LAID
3685            $LUT INSTR=LRA(S-):1 111 011 111 XX,DSPL=6,SM,INDR,F2,ENTRY=LAIS
3686            $LUT INSTR=LRA(S-):1 111 111 111 XX,DSPL=6,SM,X,INDR,F2,ENTRY=LAIS
3687            *
3688    037F    LAIS SR   SM   ADD                              ADD                    UBA:=SR + SM;
3690    0380    LAID UBA  DSPL ADSB      RH    ROD  SR   SM    JSB  LPSH SP3B  SR7   RH:=UBA:=BASE +/- DSPL, READ;
3692                                                                                  SP3B:=UBB:=SR + SM, JSZ IF SR = 7
3693    0381           UBA  SM   CAD            EPSH UBB  UBA   BNDE      CTR   TICB  ; BOUNDS CHECK S>=ADDR, CTR:=S-ADDR
3695                                                                                    SKIP NEXT IF ADDR>=SM AND S>=ADDR AND
3696                                                                                    NOT (FSS AND (DB REL ADDR OR TFF))
3697    0382           XC   OPA  ADD       RH         RH   DL    BNDE              NEXT  RH:=XC + INDIRECT CELL;
3699                                                                                  BOUNDS CHECK ADDR >= DL, NEXT
3700    0383           XC   REGN ADD       RH         RH   DL    BNDE              NEXT  RH:=XC + INDIRECT CELL;
3702                                                                                  BOUNDS CHECK ADDR >= DL, NEXT
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3704            ********************************************************************
3705            *     STOR; DB, Q, and S relative                                  *
3706            ********************************************************************
3707            *
3708            $LUT INSTR=STOR(DB+):0 101 001 0XX XX,SR=1,DSPL=8,DB,ENTRY=SWD
3709            $LUT INSTR=STOR(DB+):0 101 101 0XX XX,SR=1,DSPL=8,DB,X,ENTRY=SWD
3710            $LUT INSTR=STOR(Q+):0 101 001 10X XX,SR=1,DSPL=7,Q,ENTRY=SWD
3711            $LUT INSTR=STOR(Q+):0 101 101 10X XX,SR=1,DSPL=7,Q,X,ENTRY=SWD
3712            $LUT INSTR=STOR(Q-):0 101 001 110 XX,SR=1,DSPL=6,Q,F2,ENTRY=SWD
3713            $LUT INSTR=STOR(Q-):0 101 101 110 XX,SR=1,DSPL=6,Q,F2,X,ENTRY=SWD
3714            $LUT INSTR=STOR(S-):0 101 001 111 XX,SR=1,DSPL=6,SM,F2,ENTRY=SWS
3715            $LUT INSTR=STOR(S-):0 101 101 111 XX,SR=1,DSPL=6,SM,F2,X,ENTRY=SWS
3716            *
3717    0384    SWS   SR   UBA  ADD                          ADD                      UBA:=SR + (SM + XC);
3719    0385    SWD   UBA  DSPL ADSB       RH   WRD   SR   SM   ADD        SP3B             RH:=UBA:=(BASE + XC) +/- SR=1,DSPL, READ
3721                                                                                SP3B:=UBB:=SR + SM
3722    0386    SWD1  UBA  SM   CAD              EPOP  UBB  UBA  BNDE       CTR  TICB       ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3724                                                                                  S>=ADDR AND NOT ((FSS AND
3725                                                                                  (DB OR P REL ADDR)) OR TFF) THEN (S-ADDR)
3726                                                                                  ELSE %(8)17
3727    0387                RA   ADD        REGN DATA  RH   DL   BNDE                NEXT  WRITE (S); BOUNDS CHECK ADDR >= DL, NEXT
3729            *
3730            ********************************************************************
3731            *     STOR; Indirect DB, Q, and S relative                         *
3732            ********************************************************************
3733            *
3734            $LUT INSTR=STOR(DB+):0101 0110 XXXX,SR=1,DSPL=8,DB,INDR,ENTRY=SWID
3735            $LUT INSTR=STOR(DB+):0101 1110 XXXX,SR=1,DSPL=8,DB,X,INDR,ENTRY=SWID
3736            $LUT INSTR=STOR(Q+):0101 0111 0XXX,SR=1,DSPL=7,Q,INDR,ENTRY=SWID
3737            $LUT INSTR=STOR(Q+):0101 1111 0XXX,SR=1,DSPL=7,Q,X,INDR,ENTRY=SWID
3738            $LUT INSTR=STOR(Q-):0101 0111 10XX,SR=1,DSPL=6,Q,INDR,F2,ENTRY=SWID
3739            $LUT INSTR=STOR(Q-):0101 1111 10XX,SR=1,DSPL=6,Q,X,INDR,F2,ENTRY=SWID
3740            $LUT INSTR=STOR(S-):0101 0111 11XX,SR=1,DSPL=6,SM,INDR,F2,ENTRY=SWIS
3741            $LUT INSTR=STOR(S-):0101111111XX,SR=1,DSPL=6,SM,X,INDR,F2,ENTRY=SWIS
3742            *
3743    0388    SWIS  SR   SM   ADD                          ADD                      UBA:=SR + SM;
3745    0389    SWID  UBA  DSPL ADSB       RH   ROD   SR   SM   ADD        SP3B             RH:=UBA:=BASE +/- DSPL, READ;
3747                                                                                SP3B:=UBB:=SR + SM
3748    038A          UBA  SM   CAD                    UBB  UBA  BNDE       CTR  TICB       ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3750                                                                                S>=ADDR AND NOT (FSS AND (DB OR P REL ADDR))
3751                                                                                THEN (S-ADDR) ELSE %(8)17
3752    038B          XC   DB   ADD                    RH   DL   BNDE            STFF       UBA:=XC + DB; BOUNDS CHECK ADDR >= DL, STFF
3754    038C          UBA  REGN ADD        RH   WRD   SP3B      JSB  SWD1             UNC   RH:=UBA:=TARGET ADDR, WRITE; UBB:=S, JSB
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3757            ********************************************************************
3758            *    STD; DB, Q, and S relative                                    *
3759            ********************************************************************
3760            *
3761            $LUT INSTR=STD(DB+):1 110 001 0XX XX,SR=2,DSPL=8,DB,ENTRY=SDD
3762            $LUT INSTR=STD(DB+):1 110 101 0XX XX,SR=2,DSPL=8,DB,XSL,ENTRY=SDD
3763            $LUT INSTR=STD(Q+):1 110 001 10X XX,SR=2,DSPL=7,Q,ENTRY=SDD
3764            $LUT INSTR=STD(Q+):1 110 101 10X XX,SR=2,DSPL=7,Q,XSL,ENTRY=SDD
3765            $LUT INSTR=STD(Q-):1 110 001 110 XX,SR=2,DSPL=6,Q,F2,ENTRY=SDD
3766            $LUT INSTR=STD(Q-):1 110 101 110 XX,SR=2,DSPL=6,Q,F2,XSL,ENTRY=SDD
3767            $LUT INSTR=STD(S-):1 110 001 111 XX,SR=2,DSPL=6,SM,F2,ENTRY=SDS
3768            $LUT INSTR=STD(S-):1 110 101 111 XX,SR=2,DSPL=6,SM,F2,XSL,ENTRY=SDS
3769            *
3770    038D    SDS   SR   UBA  ADD                          ADD                      UBA:=SR + (SM + XC);
3772    038E    SDD   UBA  DSPL ADSB       RH   WRD   SR   SM   ADD       SP3B      RH:=UBA:=(BASE + XC) +/- DSPL, SEND ADDR
3774    038F    SDD1  UBA  SM   CAD            EPOP  UBB  UBA  BNDE      CTR  TICB  EPOP; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM
                                                                                     AND S>=ADDR AND NOT (FSS AND (DB REL ADDR
                                                                                     OR TFF)) THEN (S-ADDR) ELSE CODE FOR NOP
3776
3777
3778    0390                 RB   ADD        REGN DATA  RH   DL   BNDE               WRITE (S-1); BOUNDS CHECK ADDR>=DL
3780    0391          RH     SM   SUB             EPOP  SP3B RH   BNDG      CTR  TICB  EPOP; BOUNDS CHECK ADDR>S, CTR:=IF ADDR>=SM
                                                                                     AND S>ADDR AND NOT (FSS AND (DB REL ADDR
                                                                                     OR TFF)) THEN (S-ADDR-1) ELSE CODE FOR NOP
3782
3783
3784    0392                 RA   ADD        REGN DATA                ADD          NEXT  WRITE (S); NEXT
3786            *
3787            ********************************************************************
3788            *    STD; Indirect DB, Q, and S relative                           *
3789            ********************************************************************
3790            *
3791            $LUT INSTR=STD(DB+):1110 0110 XXXX,SR=2,DSPL=8,DB,INDR,ENTRY=SDID
3792            $LUT INSTR=STD(DB+):1110 1110 XXXX,SR=2,DSPL=8,DB,XSL,INDR,ENTRY=SDID
3793            $LUT INSTR=STD(Q+):1110 0111 0XXX,SR=2,DSPL=7,Q,INDR,ENTRY=SDID
3794            $LUT INSTR=STD(Q+):1110 1111 0XXX,SR=2,DSPL=7,Q,XSL,INDR,ENTRY=SDID
3795            $LUT INSTR=STD(Q-):1110 0111 10XX,SR=2,DSPL=6,Q,INDR,F2,ENTRY=SDID
3796            $LUT INSTR=STD(Q-):1110111110XX,SR=2,DSPL=6,Q,XSL,INDR,F2,ENTRY=SDID
3797            $LUT INSTR=STD(S-):1110 0111 11XX,SR=2,DSPL=6,SM,INDR,F2,ENTRY=SDIS
3798            $LUT INSTR=STD(S-):1110111111XX,SR=2,DSPL=6,SM,XSL,INDR,F2,ENTRY=SDIS
3799            *
3800    0393    SDIS  SR   SM   ADD                          ADD                      UBA:=SR + SM;
3802    0394    SDID  UBA  DSPL ADSB       RH   ROD   SR   SM   ADD       SP3B      RH:=UBA:=BASE +/- DSPL, READ; SP3B:=UBB:=S
3804    0395          UBA  SM   CAD                  UBB  UBA  BNDE      CTR  TICB  ; BOUNDS CHECK ADDR>S, CTR:=IF ADDR>=SM AND
                                                                                     S>ADDR AND NOT (FSS AND DB REL ADDR) THEN
3806                                                                                 (S-ADDR) ELSE CODE FOR NOP
3807
3808    0396          XC   DB   ADD                  RH   DL   BNDE          STFF  UBA:=XC + DB; BOUNDS CHECK ADDR >= DL, STFF
3810    0397          UBA  REGN ADD        RH   WRD   SP3B      JSB  SDD1          UNC  RH:=UBA:=XC + DB + INDIRECT CELL, SEND ADDR;
3812                                                                                 UBB:=S, JSB
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3814          ***************************************************************
3815          *    STB:  Q and S relative                                     *
3816          ***************************************************************
3817          *
3818          $LUT INSTR=STB(Q+):1 110 000 10X XX,SR=1,DSPL=7,Q,ENTRY=SBQ
3819          $LUT INSTR=STB(Q+):1 110 100 10X XX,SR=1,DSPL=7,Q,XSR,ENTRY=SBQ
3820          $LUT INSTR=STB(Q-):1 110 000 110 XX,SR=1,DSPL=6,Q,F2,ENTRY=SBQ
3821          $LUT INSTR=STB(Q-):1 110 100 110 XX,SR=1,DSPL=6,Q,F2,XSR,ENTRY=SBQ
3822          $LUT INSTR=STB(S-):1 110 000 111 XX,SR=1,DSPL=6,SM,F2,ENTRY=SBS
3823          $LUT INSTR=STB(S-):1 110 100 111 XX,SR=1,DSPL=6,SM,F2,XSR,ENTRY=SBS
3824          *
3825  0398  SBS  SR   UBA  ADD                          ADD                        UBA:=SR + (SM + XC);
3827  0399  SBQ  UBA  DSPL ADSB       RH   ROBD  SR   SM   ADD        SP3B       RH:=UBA:=(BASE + XC) +/- DSPL, READ;
3829                                                                             SP3B:=UBB:=S + SM
3830  039A       XC        CSR             HBF2  8000 UBA  ADDL       SP1B       F2:=XC(15); SP1B:=ADDR + 32K          (0118)
3832  039B       RH   SM   CAD             EPOP  SP3B RH   SUB        CTR  TICB  EPOP; CTR:=IF ADDR>SM AND S>=ADDR AND
3834                                                                               NOT (FSS AND (DB REL ADDR OR TFF)) THEN
3835                                                                               (S-ADDR) ELSE CODE FOR OPERAND
3836  039C       RH   DL   JSBS SB32       NCRY  SP3B RH   JSBS SB32       NCRY  JSB IF NOT (ADDR >= DL);
3838                                                                             JSB IF NOT (S >= ADDR)
3839  039D            RA   ADD  RRZ        NF2         REGN ADD  LLZ        NF2   UBA:=BYTE IN (S), SKIP IF NOT RIGHT BYTE;
3841                                                                             UBB.(0:8)=LEFT BYTE OF WORD AT STORE ADDR,
3842                                                                               SKIP IF NOT RIGHT BYTE
3843  039E       UBA  UBB  IOR        REGN DATA        SP1B ADD                  NEXT  MERGE BYTES, WRITE; UBB:=ADDR + 32K, NEXT
3845  039F            RA   ADD  RLZ                    REGN ADD  RRZ
3847                                                                             UBB:=RIGHT BYTE OF WORD AT MEMORY LOCATION
3848  03A0       UBA  UBB  IOR        REGN DATA             ADD                  NEXT  MERGE BYTES, WRITE; NEXT
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL  RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3851          ********************************************************************
3852          *    STB:  DB relative                                             *
3853          ********************************************************************
3854          *
3855          $LUT INSTR=STB(DB+):1 110 000 OXX XX,SR=1,DSPL=8,DB,ENTRY=SBD
3856          $LUT INSTR=STB(DB+):1 110 100 OXX XX,SR=1,DSPL=8,DB,XSR,ENTRY=SBD
3857          *
3858    03A1  SBD   UBA  DSPL ADSB      RH   ROBD  SR   SM   ADD       SP3B            RH:=UBA:=(BASE + XC) +/- DSPL, READ;
3860                                                                                    SP3B:=UBB:=S + SM
3861    03A2        XC         CSR           HBF2  8000 UBA  ADDL      SP1B            F2:=XC(15); SP1B:=ADDR + 32K          (0118)
3863    03A3              DB   ADD       SPOA EPOP  RH   DL   SUB           TICB NCRY  SPOA:=DB, EPOP;
3865                                                                                    SKIP IF NOT (ADDR >= DL), TICB CAUSES CTR
3866                                                                                      TO GET %17 (CODE FOR OPERAND)
3867    03A4        RH   SM   JSBC SBDR      FSS   SP3B RH   SUB       CTR  TICB CRRY  JSB IF SPLIT BANKS;
3869                                                                                    CTR:=IF ADDR>SM AND S>=ADDR AND
3870                                                                                      NOT (FSS AND (DB REL ADDR OR TFF)) THEN
3871                                                                                      (S-ADDR) ELSE CODE FOR OPERAND,
3872                                                                                      SKIP IF S >= ADDR
3873    03A5        SPOA DL   SUB                    Z    DB   JSBS SB32          TICB  ; JSB IF NOT (DL <= ADDR <= S) AND NOT FSS
3875                                                                                      AND (DB >= DL) AND (Z >= DB)
3876                                                                                      (JSB IF ((DL > ADDR) OR (ADDR > S)) AND
3877                                                                                      SPLIT STACK)
3878    03A6  SBDR        RA   ADD  RRZ       NF2        REGN ADD  LLZ           NF2   UBA:=BYTE IN (S), SKIP IF NOT RIGHT BYTE;
3880                                                                                    UBB:(0:8)=LEFT BYTE OF WORD AT STORE ADDR,
3881                                                                                      SKIP IF NOT RIGHT BYTE
3882    03A7        UBA  UBB  IOR       REGN DATA        SP1B ADD                 NEXT  MERGE BYTES, WRITE; UBB:=ADDR + 32K, NEXT
3884    03A8  SBDL        RA   ADD  RLZ                  REGN ADD  RRZ
3886                                                                                    UBA.(0:8):=BYTE IN (S);
3887    03A9        UBA  UBB  IOR       REGN DATA              ADD               NEXT  UBB:=RIGHT BYTE OF WORD AT MEMORY LOCATION
3889    03AA  SB32       SP1B ADD            ROBD  UBB  DL   BNDE              TICB  MERGE BYTES, WRITE; NEXT
3891                                                                                    READ ADDR + 32K;
3892                                                                                    BOUNDS CHECK (ADDR + 32K) >= DL, TICB SLOWS
3893    03AB        UBA  SM   JSBC SBDL      NF2   SP3B UBA  BNDE      CTR  TICB RSB   DOWN RSB (WILL ALWAYS BE FALSE)
3895                                                                                    JSB for storing into left byte if not F2:
3896                                                                                      Bounds check S >= addr, CTR := if addr >
3897                                                                                      SM and S >= addr and not (FSS and DB-rel
3898                                                                                      addr) then (S-addr) else code for NOP,
                                                                                        medium RSB
```

```
RECORD  C S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

3900            ********************************************************************
3901            *    STB:  Indirect, DB, Q and S relative                          *
3902            ********************************************************************
3903            *
3904            $LUT INSTR=STB(DB+):1110 0100 XXXX,SR=1,DSPL=8,DB,INDR,ENTRY=SBID
3905            $LUT INSTR=STB(DB+):1110 1100 XXXX,SR=1,DSPL=8,DB,X,INDR,ENTRY=SBID
3906            $LUT INSTR=STB(Q+):1110 0101 0XXX,SR=1,DSPL=7,Q,INDR,ENTRY=SBID
3907            $LUT INSTR=STB(Q+):1110 1101 0XXX,SR=1,DSPL=7,Q,X,INDR,ENTRY=SBID
3908            $LUT INSTR=STB(Q-):1110 0101 10XX,SR=1,DSPL=6,Q,INDR,F2,ENTRY=SBID
3909            $LUT INSTR=STB(Q-):1110 1101 10XX,SR=1,DSPL=6,Q,X,INDR,F2,ENTRY=SBID
3910            $LUT INSTR=STB(S-):1110 0101 11XX,SR=1,DSPL=6,SM,INDR,F2,ENTRY=SBIS
3911            $LUT INSTR=STB(S-):1110 1101 11XX,SR=1,DSPL=6,SM,X,INDR,F2,ENTRY=SBIS
3912            *
3913    03AC  SBIS SR   SM   ADD                          ADD                      UBA:=SR + SM;
3915    03AD  SBID UBA  DSPL ADSB      SPOA ROD   SR   SM   ADD       SP3B        SPOA:=UBA:=(BASE + XC) +/- DSPL, READ;
3917                                                                              SP3B:=UBB:=SR + SM
3918    03AE       UBA  SM   CAD                    UBB  UBA  BNDE      CTR  TICB  ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3920                                                                                S>=ADDR AND NOT (FSS AND DB REL ADDR)
3921                                                                                THEN (S-ADDR) ELSE CODE FOR OPERAND
3922    03AF            DB   ADD            NFSS  Z    DB   SUB             STFF NCRY  UBA:=DB, SKIP IF NOT SPLIT BANKS;
3924                                                                              STFF, SKIP IF NOT (Z >= DB)
3925    03B0       XC   REGN ADD  LSR       CF1   UBA  DL   SUB             CTF1       UBA:=(XC + INDIRECT CELL) & LSR(1);
3927                                                                              F1:=NOT SPLIT BANKS AND (Z >= DB) AND
3928                                                                                (DB >= DL)
3929    03B1       XC   REGN CSR            HBF2  UBA  DB   ADD       RH   ROD       SET F2 IF EFFECTIVE BYTE ADDR IS ODD;
3931                                                                              RH:=ABSOLUTE WORD ADDR, READ
3932    03B2       UBB  SM   JSBC SBI1      NF1   SP3B UBB  SUB       CTR  TICB CRRY  JSB IF SPLIT STACK;
3934                                                                              CTR:=IF ADDR>SM AND S>=ADDR AND NOT SPLIT
3935                                                                                BANKS THEN (S-ADDR) ELSE CODE FOR OPERAND
3936    03B3       RH   DL   JSBS SI32      NCRY                JSB  SI32            UNC  JSB IF NOT (ADDR >= DL);
3938                                                                              JSB IF NOT (S >= ADDR)
3939    03B4  SBI1     RA   ADD  RLZ       F2          REGN ADD  RRZ       EPOP F2  UBA.(0:8):=BYTE OUT OF (S);
3941                                                                              UBB:=RIGHT BYTE OF WORD AT MEMORY LOCATION,
3942                                                                                SKIP IF NOT STORING INTO LEFT BYTE
3943    03B5       SPOA DL   BNDE           NEXT  UBA  UBB  IOR       REGN DATA      BOUNDS CHECK INDIRECT CELL >= DL, NEXT;
3945                                                                              MERGE BYTES, WRITE
3946    03B6            RA   ADD  RRZ                 REGN ADD  LLZ                UBA:=BYTE OUT OF (S);
3948                                                                              UBB:=LEFT BYTE OF WORD AT MEMORY LOCATION
3949    03B7       SPOA DL   BNDE           NEXT  UBA  UBB  IOR       REGN DATA      BOUNDS CHECK INDIRECT CELL>=DL, NEXT;
3951                                                                              MERGE BYTES, WRITE
3952    03B8  SI32          ADD                 8000 RH   ADDL                      ; UBB:=32K + ADDR
3954    03B9       UBB  DL   BNDE                     UBB  ADD       RH   ROD  TICB  BOUNDS CHECK ADDR >= DL; RH:=ADDR, READ,
3956                                                                                TICB TO SLOW DOWN RSB (ALWAYS FALSE)
3957    03BA       UBB  SM   CAD                 SP3B UBB  BNDE      CTR  TICB RSB  ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
3959                                                                                S>=ADDR AND NOT (FSS AND DB REL ADR) THEN
3960                                                                                (S-ADDR) ELSE CODE FOR NOP, MEDIUM RSB
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 3962           ********************************************************************
 3963           *    The following instructions access the 5 bits, 11 through 15    *
 3964           *    from CIR for the value of DSPL.  F2 is initialized to the      *
 3965           *    value of bit 10 in the instruction.  Instructions IABZ and     *
 3966           *    DABZ require an SR preadjust of one.                           *
 3967           ********************************************************************
 3968           *
 3969           ********************************************************************
 3970           *    IABZ instruction, (Increment A, branch if zero)  -  IABZ is    *
 3971           *    the entry point for the direct case, and IABI is the entry     *
 3972           *    point for the indirect case.                                   *
 3973           ********************************************************************
 3974           *
 3975           $LUT INSTR=IABZ:0 001 000 111 0X,SR=1,DSPL=5,P,ENTRY=IABZ
 3976           $LUT INSTR=IABZ:0 001 100 111 0X,SR=1,DSPL=5,P,INDR,ENTRY=IABI
 3977           $LUT INSTR=IABZ:0 001 000 111 1X,SR=1,DSPL=5,P,F2,ENTRY=IABZ
 3978           $LUT INSTR=IABZ:0 001 100 111 1X,SR=1,DSPL=5,P,INDR,F2,ENTRY=IABI
 3979           *
 3980    03BB   IABZ RA         JSBI IABC      NZRO       P    ADD             CF2         JSB IF (S)+1<>0; UBB:=P BEFORE INCREMENTING
 3982    03BC        UBB  DSPL ADSB           RONP            JSB  DAB1            UNC   UBA:=TARGET P, READ INTO NIR; JSB
 3984    03BD   IABC RA         INC       RA   CCO            JSZ  NEXT            UNC   (S):=(S) + 1, CCO; NEXT
 3986    03BE   IABI UBA  DSPL ADSB      RH   ROP    RA         INC             CF2  ZERO  RH:=INDIRECT ADDR, READ; SKIP IF (S)+1=0
 3988    03BF        UBA  PB   SUB            CTF1            JSBI IABC SP1B       UNC   F1:=0 IF BNDV; JSB IF NOT BRANCH, SP1B =1
 3990    03C0   IAI1 RH   OPA  ADD            RONP  PL   RH    UBNE                       UBA:=TARGET P, READ INTO NIR; BOUNDS CHECK
 3992    03C1        UBA  PB   UBNE           F1    PL   UBA   UBNE                 NF1   BOUNDS CHECK TARGET P, SKIP IF NOT BNBV;
 3994    03C2             JSZ  BNDV      UNC            SREG JSB  DABC P           UNC   BNDV IF NOT F1; P:=TARGET P, JSB
 3996    03C3        RA   SP1B ADSB      RA   CCO            JSZ  NEXT            UNC   (S):=(S) +/- 1, CCO; NEXT
 3998           *
 3999           ********************************************************************
 4000           *    DABZ instruction, (Decrement A, branch if zero)  -  DABZ is    *
 4001           *    the entry point for the direct case, and DABI is the entry     *
 4002           *    point for the indirect case.                                   *
 4003           ********************************************************************
 4004           *
 4005           $LUT INSTR=DABZ:0 001 010 111 0X,SR=1,DSPL=5,P,ENTRY=DABZ
 4006           $LUT INSTR=DABZ:0 001 110 111 0X,SR=1,DSPL=5,P,INDR,ENTRY=DABI
 4007           $LUT INSTR=DABZ:0 001 010 111 1X,SR=1,DSPL=5,P,F2,ENTRY=DABZ
 4008           $LUT INSTR=DABZ:0 001 110 111 1X,SR=1,DSPL=5,P,F2,INDR,ENTRY=DABI
 4009           *
 4010    03C4   DABZ RA         JSBC DABC      NZRO       P    ADD             SF2         JSB IF (S)-1<>0; UBB:=P BEFORE INCREMENTING
 4012    03C5        UBB  DSPL ADSB           RONP            ADD                         UBA:=TARGET P, READ INTO NIR
 4014    03C6   DAB1 UBA  PB   UBNE           F2    PL   UBA   UBNE                       SKIP IF FROM IABZ; BOUNDS CHECK TARGET P
 4016    03C7             ADD              RSB            SREG ADD        P             RSB IF IABZ; P:=TARGET P AFTER BOUNDS CHECK
 4018    03C8   DABC RA         CAD       RA   CCO            JSZ  NEXT            UNC   (S):=(S)-1, CCO; NEXT
 4020    03C9   DABI UBA  DSPL ADSB      RH   ROP    RA         CAD             SF2  ZERO  RH:=INDIRECT ADDR, READ; SKIP IF (S)-1=0
 4022    03CA        UBA  PB   JSBS IAI1      CTF1            JSBI DABC SP1B       UNC   F1:=0 IF BNDV; JSB IF NOT BRANCH, SP1B:=1
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4025            ************************************************************************
4026            *    IXBZ instruction, (Increment X, branch if zero)  -  IXBZ is      *
4027            *    the entry point for the direct case, and IXBI is the entry       *
4028            *    point for the indirect case.                                     *
4029            ************************************************************************
4030            *
4031            $LUT INSTR=IXBZ:0 001 001 010 0X,DSPL=5,P,ENTRY=IXBZ
4032            $LUT INSTR=IXBZ:0 001 101 010 0X,DSPL=5,P,INDR,ENTRY=IXBI
4033            $LUT INSTR=IXBZ:0 001 001 010 1X,DSPL=5,P,F2,ENTRY=IXBZ
4034            $LUT INSTR=IXBZ:0 001 101 010 1X,DSPL=5,P,F2,INDR,ENTRY=IXBI
4035            *
4036    03CB    IXBZ X          JSBI IXBC       NZRO        P    ADD             CF2         JSB IF X+1 <> 0; UBB:=P BEFORE INCREMENTING
4038    03CC         UBB  DSPL ADSB            RONP             JSB  DXBI            UNC    UBA:=TARGET P, READ; JSB
4040    03CD    IXBC X          INC       X     CCO              JSZ  NEXT            UNC    X:=X+1, CCO; JSB TO WAIT ONE CLOCK FOR NEXT
4042    03CE    IXBI X          JSBI IXBC       NZRO        P    ADD                         JSB IF X+1 <> 0; UBB:=P BEFORE INCREMENTING
4044    03CF         UBB  DSPL ADSB       SPOA ROP              JSBI DXII SP1B       UNC    SPOA:=INDIRECT ADDR, READ; SP1B:=1, JSB
4046            *
4047            ************************************************************************
4048            *    DXBZ instruction, (Decrement X, branch if zero)  -  DXBZ is      *
4049            *    the entry point for the direct case, and DXBI is the entry       *
4050            *    point for the indirect case.                                     *
4051            ************************************************************************
4052            *
4053            $LUT INSTR=DXBZ:0 001 001 011 0X,DSPL=5,P,ENTRY=DXBZ
4054            $LUT INSTR=DXBZ:0 001 101 011 0X,DSPL=5,P,INDR,ENTRY=DXBI
4055            $LUT INSTR=DXBZ:0 001 001 011 1X,DSPL=5,P,F2,ENTRY=DXBZ
4056            $LUT INSTR=DXBZ:0 001 101 011 1X,DSPL=5,P,F2,INDR,ENTRY=DXBI
4057            *
4058    03D0    DXBZ X          JSBC DXBC       NZRO        P    ADD             SF2         JSB IF X-1 <> 0; UBB:=P BEFORE INCREMENTING
4060    03D1         UBB  DSPL ADSB            RONP             ADD                         UBA:=TARGET P, READ
4062    03D2    DXB1 UBA  PB   UBNE            F2     PL   UBA  UBNE                        SKIP IF NOT FROM IABZ, BOUNDS CHECK TARGET P
4064    03D3              ADD                  RSB          SREG ADD        P              RSB IF FROM IABZ; P:=TARGET P
4066    03D4    DXBC X          CAD       X     CCO              JSZ  NEXT            UNC    X:=X-1, CCO; JSB TO WAIT ONE CLOCK FOR NEXT
4068    03D5    DXBI X          JSBC DXBC       NZRO        P    ADD                         JSB IF X-1 <> 0; UBB:=P BEFORE INCREMENTING
4070    03D6         UBB  DSPL ADSB       SPOA ROP              CAD         SP1B           SPOA:=INDIRECT ADDR, READ; SP1B:=-1
4072    03D7    DXI1 UBA  PB   UBNE                 PL   UBA  UBNE                        BOUNDS CHECK INDIRECT CELL ADDRESS
4074    03D8         SPOA OPA  ADD                  RONP             ADD                 UBA:=TARGET P, READ INTO NIR
4076    03D9         UBA  PB   UBNE                 PL   UBA  UBNE                        BOUNDS CHECK TARGET P
4078    03DA         X    SP1B ADD             X     CCO          SREG JSZ  NEXT P       UNC    X:=X +/- 1, CCO; P:=TARGET P, JSB
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4081            ;*****************************************************************
4082            ;     LOAD instructions use special TOS push routines because      *
4083            ;                                                                  *
4084            ;          1.  STOV is handled differently ( #STO2 )               *
4085            ;          2.  Must re-read memory in case reference was (S-6) and  *
4086            ;              TOS registers were full                             *
4087            ;*****************************************************************
4088            ;
4089    03DB  LPSH      SM   INC            WRS          SM   INC       SM   DCSR        Write at (SM + 1); SM := SM + 1, DCSR
4091    03DC          QDWN ADD            DATA  Z      SREG JSZC STO2            NEG   WRITE QDWN; STOV IF NEW SM > Z & WITHIN 32K
4093    03DD          RH   ADD            ROD   SP3B       ADD                  RSB   UBA:=ADDR, REREAD; UBB:=S, RSB
4095    03DE  LDPH      SM   INC            WRS              ADD           DCSR       WRITE AT (SM + 1); DCSR
4097    03DF          QDWN ADD            DATA         UBA  INC       SM   DCSR        WRITE QDWN; STOV IF NEW SM > Z & WITHIN 32K
4099    03E0          QDWN ADD            DATA  Z      SREG JSZC STO2            NEG   WRITE QDWN; STOV IF NEW SM > Z & WITHIN 32K
4101    03E1          RH   ADD            ROD   SP3B       ADD                  RSB   UBA:=ADDR, REREAD; UBB:=S, RSB
4103    03E2  LBPH      SM   INC            WRS              SM   INC       SM   DCSR        WRITE AT (SM + 1); SM:=SM + 1, DCSR
4105    03E3          QDWN ADD            DATA  Z      SREG JSZC STO2            NEG   WRITE QDWN; STOV IF NEW SM > Z & WITHIN 32K
4107    03E4          RH   ADD            ROBD  SP3B       ADD                  RSB   UBA:=ADDR, REREAD; UBB:=S, RSB
```

```
RECORD  C_S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4110            ********************************************************************
4111            *    Set Process Clock (SCLK) instruction                          *
4112            ********************************************************************
4113            *
4114            $LUT INSTR=SCLK/SMSK:0 011 000 001 00x xxx, SR=1, ENTRY=SCLK
4115            *
4116    03E5    SCLK      CIR  JSB  SMSK      EVEN            JSZ  TRP6 BKX3      NPRV  Trap if not privileged else jump if SMSK
4118                                                                                       instruction;  BKX3 := 0
4119    03E6                   JSZ  NEXT      UNC        RA   ADD       XR20 EPOP       Jump for NEXT; Process clock := (S), pop 1
4121            *
4122            ********************************************************************
4123            *    Read Process Clock (RCLK) instruction                         *
4124            ********************************************************************
4125            *
4126            $LUT INSTR=RCLK/RMSK:0 011 000 001 01x xxx, ENTRY=RCLK
4127            *
4128    03E7    RCLK           ADD                             JSZ  PSHM BKX3      SR7   ; Empty RG if SR = 7, BKX3 := 0
4130    03E8               CIR  JSB  RMSK      EVEN            ADD                       Jump if RMSK instruction
4132    03E9                   JSZ  NEXT      UNC        XR20 ADD       RH   EPSH       Jump for NEXT; New (S) := process clk, push
```

```
RECORD   C.S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4135           ***************************************************************
4136           *     Store B into X (STBX) Instruction                         *
4137           ***************************************************************
4138           *
4139           $LUT INSTR=STBX:0 000 010 110 xxx xxx, ENTRY=STBX, SR=2
4140           *
4141     03EA  STBX      RB   ADD       X    CCA             JSZ  NEXT          UNC   X := (S-1), set CCA; Jump for NEXT
4143           *
4144           ***************************************************************
4145           *     Add A to X (ADAX) Instruction                             *
4146           ***************************************************************
4147           *
4148           $LUT INSTR=ADAX:0 000 011 110 xxx xxx, ENTRY=ADAX, SR=1
4149           *
4150     03EB  ADAX X    RA   ADD       X    CCO             JSL  DEL           UNC   X := X + (S), set CCO;
4152                                                                                   Jump to delete (S) and for NEXT
4153           ***************************************************************
4154           *     Add X to A (ADXA) Instruction                             *
4155           ***************************************************************
4156           *
4157           $LUT INSTR=ADXA:0 000 011 111 xxx xxx, ENTRY=ADXA, SR=1
4158           *
4159     03EC  ADXA X    RA   ADD       RA   CCO             JSZ  NEXT          UNC   (S) := X + (S), set CCO; Jump for NEXT
4161           *
4162           ***************************************************************
4163           *     Load X into B (LDXB) Instruction                          *
4164           ***************************************************************
4165           *
4166           $LUT INSTR=LDXB:0 000 100 010 xxx xxx, ENTRY=LDXB, SR=2
4167           *
4168     03ED  LDXB X          ADD       RB   CCA             JSZ  NEXT          UNC   (S-1) := X, set CCA; Jump for NEXT
4170           *
4171           ***************************************************************
4172           *     Store A into X (STAX) Instruction                         *
4173           ***************************************************************
4174           *
4175           $LUT INSTR=STAX:0 000 100 011 xxx xxx, ENTRY=STAX, SR=1
4176           *
4177     03EE  STAX      RA   ADD       X    CCA             JSL  DEL           UNC   X := (S), set CCA;
4179                                                                                   Jump to delete (S) and for NEXT
4180           *
4181           ***************************************************************
4182           *     Load X into A (LDXA) Instruction                          *
4183           ***************************************************************
4184           *
4185           $LUT INSTR=LDXA:0 000 100 100 xxx xxx, ENTRY=LDXA
4186           *
4187     03EF  LDXA X          ADD       RH   CCA             JSZ  PSHM          SR7   New (S) := X, set CCA; Empty RG if SR = 7
4189     03F0                  ADD            EPSH            JSZ  NEXT          UNC   Push; Jump for NEXT
```

```
RECORD  C.S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4192          ***************************************************************************
4193          *     Add B to X (ADBX) Instruction                                       *
4194          ***************************************************************************
4195          *
4196          $LUT INSTR=ADBX:0 000 111 110 xxx xxx, ENTRY=ADBX, SR=2
4197          *
4198    03F1  ADBX X    RB   ADD       X    CCO             JSZ  NEXT          UNC   X := (X) + (S-1), set CCO; Jump for NEXT
4200          *
4201          ***************************************************************************
4202          *     Add X to B (ADXB) Instruction                                       *
4203          ***************************************************************************
4204          *
4205          $LUT INSTR=ADXB:0 000 111 111 xxx xxx, ENTRY=ADXB, SR=2
4206          *
4207    03F2  ADXB X    RB   ADD       RB   CCO             JSZ  NEXT          UNC   (S-1) := X + (S-1), set CCO; Jump for NEXT
```

```
RECORD  C S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4210            ****************************************************************
4211            *     ZROX (Zero X) Instruction                                *
4212            ****************************************************************
4213            *
4214            $LUT INSTR=ZROX:0 000 000 011 xxx xxx, ENTRY=ZROX, SR=0
4215            *
4216    03F3    ZROX            ADD       X                   JSZ  NEXT          UNC   X := 0, JSB for NEXT
4218            *
4219            ****************************************************************
4220            *     ZERO (Push zero) Instruction                             *
4221            ****************************************************************
4222            *
4223            $LUT INSTR=ZERO:0 000 000 110 xxx xxx, ENTRY=ZERO, SR=0
4224            *
4225    03F4    ZERO            ADD                           JSZ  PSHM          SR7   ; JSB to push 1 TOS if SR = 7
4227    03F5    ZER1            ADD       RH   EPSH           JSZ  NEXT          UNC   New (S) := 0, push; JSB for NEXT
4229            *
4230            ****************************************************************
4231            *     DZRO (Push double zero) Instruction                      *
4232            ****************************************************************
4233            *
4234            $LUT INSTR=DZRO:0 000 000 111 xxx xxx, ENTRY=DZRO, SR=0
4235            *
4236    03F6    DZRO            JSZ  PSHM       SRG5          JSZ  PSM2          SR7   JSB to push 2 TOS if SR = 7 else push 1 TOS
4238                                                                                       if SR = 6
4239    03F7                    JSB  ZER1       UNC           ADD       RG   EPSH      Jump to finish; New (S) := 0, push 1 once
4241            *
4242            ****************************************************************
4243            *     Zero B (ZROB) Instruction                                *
4244            ****************************************************************
4245            *
4246            $LUT INSTR=ZROB:0 000 100 001 xxx xxx, ENTRY=ZROB, SR=2
4247            *
4248    03F8    ZROB            ADD       RB                  JSZ  NEXT          UNC   (S-1) := 0; Jump to NEXT
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4251            ************************************************************************
4252            *    OR - Logical OR instruction                                       *
4253            ************************************************************************
4254            *
4255            $LUT INSTR=OR:0 000 110 101 xxx xxx, ENTRY=OR, SR=2
4256            *
4257    03F9    OR    RB   RA   IOR       RB   CCA             JSL  DEL            UNC   (S-1) := (S-1) IOR (S), CCA;
4259                                                                                      Jump to delete (S) and for NEXT
4260            ***
4261            ************************************************************************
4262            *    XOR - Logical Exclusive OR instruction                            *
4263            ************************************************************************
4264            *
4265            $LUT INSTR=XOR:0 000 110 110 xxx xxx, ENTRY=XOR, SR=2
4266            *
4267    03FA    XOR   RB   RA   XOR       RB   CCA             JSL  DEL            UNC   (S-1) := (S-1) XOR (S), CCA;
4269                                                                                      Jump to delete (S) and for NEXT
4270            ****************
4271            ************************************************************************
4272            *    AND - Logical AND instruction                                     *
4273            ************************************************************************
4274            *
4275            $LUT INSTR=AND:0 000 110 111 xxx xxx, ENTRY=AND, SR=2
4276            *
4277    03FB    AND   RB   RA   AND       RB   CCA             JSL  DEL            UNC   (S-1) := (S-1) AND (S), CCA,
4279                                                                                      Jump to delete (S) and for NEXT
4280            *
4281            ************************************************************************
4282            *    NOT - One's Complement instruction                                *
4283            ************************************************************************
4284            *
4285            $LUT INSTR=NOT:0 000 110 100 xxx xxx, ENTRY=NOT, SR=1
4286            *
4287    03FC    NOT         RA   CAD       RA   CCA             JSZ  NEXT           UNC   (S) := NOT (S); Jump for NEXT
4289    03FD                     ADD                            ADD                       NOP to keep system happy              (2555)
4291    03FE                     ADD                            ADD                       NOP to keep system happy              (2555)
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4294            ************************************************************************
4295            *    MVBP is the entry point for the MVB, (Move Bytes), instruc-       *
4296            *    tion with P relative source addressing.  MVBD is the entry        *
4297            *    point for DB relative addressing.  There are four general         *
4298            *    cases and two special cases in which a move may fail.  For        *
4299            *    the general cases, the move may be in the forward direction,      *
4300            *    increasing addresses), or in the reverse direction, (decreas-     *
4301            *    ing addresses).  The move may also be aligned, (the source        *
4302            *    and target pointers are both even or both odd), or skewed,        *
4303            *    (the source and target are not both even or odd). The two         *
4304            *    special cases are where the source and target are overlapped      *
4305            *    by one such that the first byte of the source is propogated       *
4306            *    throughout.                                                       *
4307            ************************************************************************
4308            *
4309            $LUT INSTR=MVB(PB+):0 010 000 000 10,P,SR=3,ENTRY=MVBP
4310            $LUT INSTR=MVB(DB+):0 010 000 000 11,DB,SR=3,ENTRY=MVBD
4311            *
4313            %0400
4314            *
4315  0400  MVBP           JSZ  ETO3      NFSS       RA   JSL  D03G           ZERO  JSB TO EMPTY TOS TO 3 IF NOT SPLIT BANKS;
4317                                                                                 JSB FOR STACK DECREMENT IF BYTE COUNT = 0
4318  0401       RA        CAD       SP4A POS        DB   ADD       SP3B           SP4A:=BYTE COUNT - 1 IF > 0; SP3B:=DB
4320  0402       RA        INC       SP4A SF3A       BNKP ADD       BKX4           SP4A:=BYTE COUNT + 1, SF3A, BKX4:=BNKP
4322  0403            RB   ADD  LSR             0001 RB   ANDL      SP2B           UBA:=WORD PTR; SP2B:=UBB:=SOURCE PTR(15)
4324  0404       UBA  PB   ADD       RH   ROP   UBB  SP4A ASR            SF4B      RH:=UBA:=ABS SOURCE WORD ADDR, READ;
4326                                                                                UBB:=ADJUSTED WORD COUNT, SF4B FOR PB REL
4327  0405       UBA  UBB  ADD                  PL   UBA  UBNE                      UBA:=ENDING SOURCE WORD ADDR;
4329                                                                                BOUNDS CHECK PL >= STARTING WORD ADDR
4330  0406       UBA  PB   UBNE                 PL   UBA  UBNE                      BOUNDS CHECK ENDING WORD ADDR >= PB;
4332                                                                                BOUNDS CHECK PL >= ENDING WORD ADDR
4333  0407       RH   PB   UBNE                      JSB  MVB1          UNC   BOUNDS CHECK STARTING WORD ADDR >= PB; JSB
4335  0408  MVBD           JSZ  ETO3      NFSS       RA   JSB  D03G           ZERO  JSB TO EMPTY TOS TO 3 IF NOT SPLIT BANKS;
4337                                                                                JSB FOR STACK DECREMENT IF BYTE COUNT = 0
4338  0409       RA        CAD       SP4A POS        DB   ADD       SP3B           SP4A:=<BYTE COUNT>-1 IF >0; SP3B:=DB
4340  040A       RA        INC       SP4A SF3A       BNKD JSB  DBBC BKX4      UNC  SP4A:=<BYTE COUNT>+1, SF3A FOR REVERSE MOVE;
4342                                                                                BKX4:=SOURCE BANK, JSB TO FORM SOURCE ADDR
4343  040B  MVB1      SP2B JSBS DBCC            HBF2       RH   ADD       SP3B     F2:=SOURCE PTR(15), CHECK TARGET ADDR;
4345                                                                                SP3B:=STARTING SOURCE WORD ADDRESS
4346  040C           RA   SUB       RH   F3A   RB   RC   XOR            LBF5      RH:=-(WORD COUNT) IF REVERSE MOVE;
4348                                                                                F5B:=SOURCE AND TARGET ADDRESSES SKEWED
4349  040D           RA   ADD       RH   UNC             JSBI MVBA XR0      NF5B  RH:=(WORD COUNT) IF FORWARD MOVE; ALIGNED
4351                                                                                XR0:=1, JSB IF ALIGNED MOVE
4352  040E  MVBS           JSB  MBSR      UNC   SP3B       ADD            RAB4      JSB IF REVERSE SKEWED MOVE; READ SOURCE WORD
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL  RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4355            $NOWARN
4356            ********************************************************************
4357            *    move bytes skewed forward                                     *
4358            ********************************************************************
4359            *
4360    040F    MBSF        OPA  ADD  LLZ  XR0        RH         ADD  LSR  RH        F2    XR0:=LEFT TARGET BYTE;
4362                                                                                        RH:=WORD COUNT, SKIP IF ODD SOURCE PTR
4363    0410                OPA  ADD  RLZ  SP4A       RH         CAD  LSR  RH        UNC   SP4A.(0:8):=RIGHT SOURCE BYTE;
4365                                                                                        RH:=WORD COUNT IF EVEN SOURCE PTR
4366    0411                OPA  ADD  LRZ                         JSB  BBFC SP2B      UNC   UBA.(8:8):=LEFT SOURCE BYTE;
4368                                                                                        SP2B:=0 AND JSB IF ODD SOURCE PTR
4369    0412        UBA     XR0  IOR       XR0  DATA             INC       SP2B            XR0:=FIRST WORD, WRITE; SP2B:=1
4371    0413    BBFC RC     RB   JSBC MBBF      ZERO   SP3B       INC       SP3B RAB4       JSB IF SOURCE AND TARGET PTRS OVERLAP BY ONE
4372                                                                                        SP3B:=SP3B+1, READ SOURCE WORD
4374    0414    BSF0                JSB  BAF4     TEST   RH   XR0  JSBS BSF2 RH          NEG   JSB IF INTERRUPT; JSB IF (WORD COUNT - 1)<0
4376    0415                OPA  ADD  RLZ  SP4A             OPB  ADD  LRZ                   SP4A.(0:8):=RIGHT SOURCE BYTE;
4378                                                                                        UBB.(8:8):=LEFT SOURCE BYTE
4379    0416        UBB     SP4A IOR                DATA   0002 SP2B ADDL      SP2B            WRITE TARGET WORD; SP2B:=SP2B+2
4381    0417                     JSB  BSF0          UNC    SP3B       INC       SP3B RAB4       JSB BACK; SP3B:=SP3B+1, READ SOURCE WORD
4383    0418    BSF2        SP1B ADD               ROD    RA   SP2B JSBS BAF4 RA          ZERO  READ LAST TARGET BYTE;
4385                                                                                        (S):=(S) - #BYTES MOVED, JSB IF 0
4386    0419    BSF3                JSB  BAF3     UNC    RA   SP2B CAD       RA              JSB UNC; (S):=(S) + #BYTES MOVED + 1
4388            *
4389            ********************************************************************
4390            *    move bytes blanking buffer forward                            *
4391            ********************************************************************
4392            *
4393    041A    MBBF        SP4A ADD  SWAB       NF2    RH         ADD       CTR       NF4B  UBA:=RIGHT SOURCE BYTE, SKIP IF EVEN SOURCE
4395                                                                                        CTR:=(WORD COUNT).(8:8)
4396    041B        UBA     SP4A IOR       XR0                     ADD                 RSB   XR0:=RIGHT SOURCE BYTE IN BOTH BYTES, JSB;
4398                                                                                        RSB IF SOURCE WAS PB RELATIVE
4399    041C    BBF0                JSB  BAF4     TEST        SP2B REPC           DCTR CTR0  JSB IF INTERRUPT;
4401                                                                                        UBB:=SP2B, REPC IF WORD COUNT > 0
4402    041D                XR0  ADD               DATA   UBB  XR0  INC       SP2B DCTR CTR0  WRITE DUPLICATED SOURCE BYTE;
4404                                                                                        SP2B:=UBB:=SP2B+2, CTR:=CTR-1, CTR=0?
4405    041E                SP2B ADD  LSR                RA   SP2B JSBS BAF4 RA          ZERO  UBA:=NUMBER OF WORDS MOVED;
4407                                                                                        (S):=(S) - #BYTES MOVED, JSB IF 0
4408    041F        RH      UBA  SUB  RRZ        NZRO   RA         ADD       RA              UBA:=(REMAINING WORD COUNT).(8:8),
4410                                                                                        SKIP IF TBUSA <> 0;  RESTORE RA
4411    0420                SP1B JSB  BBF1       ROD         UBA  ADD       CTR       NZRO  JSB & READ LAST TARGET WORD IF WORD COUNT=0;
4413                                                                                        CTR:=(WORD COUNT).(8:8), SKIP IF <> 0
4414    0421                     JSB  BBF0       UNC                ADD            DCTR      JSB BACK; CTR:=%FF IF (WORD COUNT).(8:8)=0
4416    0422    BBF1        XR0  ADD  LLZ  SP4A                     JSB  BAF3 RA        UNC   SP4A.(0:8):=SOURCE BYTE; RA:=0, JSB
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4419            ********************************************************************
4420            *    move bytes skewed reverse                                     *
4421            ********************************************************************
4422            *
4423    0423    MBSR       OPA  ADD  RRZ  XR0         RH         ADD  LSR  RH         NF2   XR0:=LEFT TARGET BYTE;
4425                                                                                         RH:=WORD COUNT, SKIP IF EVEN SOURCE PTR
4426    0424               OPA  ADD  LRZ  SP4A        RH         CAD  LSR  RH         UNC   SP4A.(0:8):=RIGHT SOURCE BYTE;
4428                                                                                         RH:=WORD COUNT IF ODD SOURCE PTR
4429    0425               OPA  ADD  RLZ                         JSB  BBRC SP2B       UNC   UBA.(8:8):=LEFT SOURCE BYTE;
4431                                                                                         SP2B:=0 AND JSB IF EVEN SOURCE PTR
4432    0426       UBA     XR0  IOR       XR0   DATD            INC       SP2B               XR0:=FIRST WORD, WRITE; SP2B:=1
4434    0427    BBRC RB    RC   JSBC MBBR      ZERO  SP3B        CAD       SP3B RAB4          JSB IF SOURCE AND TARGET PTRS OVERLAP BY ONE
4436                                                                                         SP3B:=SP3B-1, READ SOURCE WORD
4437    0428    BSR0               JSB  BAR4      TEST  RH   XR0  JSBS BSR2 RH         NEG   JSB IF INTERRUPT; JSB IF (WORD COUNT - 1)<0
4439    0429               OPA  ADD  LRZ  SP4A            OPB  ADD  RLZ                      SP4A.(0:8):=RIGHT SOURCE BYTE;
4441                                                                                         UBB.(8:8):=LEFT SOURCE BYTE
4442    042A       UBB     SP4A IOR                 DATD  0002 SP2B ADDL      SP2B          WRITE TARGET WORD; SP2B:=SP2B+2
4444    042B                    JSB  BSR0          UNC   SP3B       CAD       SP3B RAB4     JSB BACK, SP3B:=SP3B-1, READ SOURCE WORD
4446    042C    BSR2       SP1B ADD                 ROD   RA   SP2B JSB  BAR4 RA         ZERO  READ LAST TARGET BYTE;
4448                                                                                         (S):=(S) + #BYTES MOVED, JSB IF 0
4449    042D    BSR3               JSB  BAR3      UNC   RA   SP2B INC       RA                 JSB; RA:=RA + #BYTES MOVED + 1
4451            *
4452            ********************************************************************
4453            *    move bytes blanking buffer reverse                            *
4454            ********************************************************************
4455            *
4456    042E    MBBR       SP4A ADD  SWAB      F2    RH         ADD       CTR        NF4B  UBA:=LEFT SOURCE BYTE, SKIP IF ODD SOURCE;
4458                                                                                         CTR:=(WORD COUNT).(8:8)
4459    042F       UBA     SP4A IOR       XR0                   ADD                  RSB   XR0:=LEFT SOURCE BYTE IN BOTH BYTES;
4461                                                                                         JSB IF SOURCE WAS PB RELATIVE > 0
4462    0430    BBR0               JSB  BAR4      TEST       SP2B REPC           DCTR CTR0  JSB IF INTERRUPT;
4464                                                                                         UBB:=SP2B, REPC IF WORD COUNT > 0
4465    0431               XR0  ADD                 DATD  UBB  XR0  INC       SP2B DCTR CTR0  WRITE DUPLICATED SOURCE BYTE;
4467                                                                                         SP2B:=UBB:=SP2B+2, CTR:=STR-1, CTR=0?
4468    0432               SP2B ADD  LSR                 RA   SP2B JSB  BAR4 RA         ZERO  UBA:=NUMBER OF WORDS MOVED;
4470                                                                                         (S):=(S) - #BYTES MOVED, JSB IF 0
4471    0433       RH      UBA  SUB  RRZ         NZRO  RA         ADD       RA                 UBA:=(REMAINING WORD COUNT).(8:8),
4473                                                                                         SKIP IF TBUSA <> 0;  RESTORE RA
4474    0434               SP1B JSB  BBR1        ROD         UBA  ADD       CTR        NZRO  JSB & READ LAST TARGET WORD IF WORD COUNT=0;
4476                                                                                         CTR:=(WORD COUNT).(8:8), SKIP IF <> 0
4477    0435                    JSB  BBR0        UNC               ADD            DCTR       JSB BACK; CTR:=%FF IF (WORD COUNT).(8:8)=0
4479    0436    BBR1       XR0  ADD  RRZ  SP4A                     JSB  BAR3 RA         UNC   SP4A.(8:8):=SOURCE BYTE; RA:=0, JSB
4481    0437    MVBA RH           ADD  LSR  SP4A  NF3A  SP3B       ADD            RAB4       SP4A:=WORD COUNT, SKIP IF NOT REVERSE MOVE
4483                                                                                         READ SOURCE WORD INTO OPA AND OPB
4484    0438               OPA  JSB  MBAR        UNC               ADD       SP2B               UBA:=TARGET WORD; SP2B:=0
4486            $WARN
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4488            ********************************************************************
4489            *    move bytes aligned forward                                    *
4490            ********************************************************************
4491            *
4492    0439    MBAF       UBA  ADD  LLZ  SP4A           SP4A JSB  BAF0 RH         NF2   SP4A:=LEFT TARGET BYTE;
4494                                                                                      JSB IF SOURCE PTR EVEN, RH:=WORD COUNT
4495    043A               OPA  ADD  RRZ             RH        CAD  LSR  RH               UBA:=RIGHT SOURCE BYTE; RH:=WORD COUNT - 1
4497    043B       UBA     SP4A IOR                  DATA      JSBI BAF1 SP2B       UNC   WRITE FIRST WORD; SP2B:=1, JSB TO READ
4499    043C    BAF0               JSB  BAF4        TEST RH   XR0  JSBS BAF2 RH         NEG   JSB IF INTERRUPT; JSB IF (WORD COUNT - 1)<0
4501    043D               OPA  ADD                  DATA 0002 SP2B ADDL      SP2B             WRITE TARGET WORD; SP2B:=SP2B+2
4503    043E    BAF1               JSB  BAF0        UNC  SP3B      INC       SP3B RAB4        JSB BACK, SP3B:=SP3B+1, READ SOURCE WORD
4505    043F    BAF2       SP1B ADD                  ROD  RA   SP2B JSBS BAF4 RA         ZERO  READ LAST TARGET BYTE;
4507                                                                                      (S):=(S) - #BYTES MOVED, JSB IF 0
4508    0440               OPA  ADD  LLZ  SP4A       RA   SP2B CAD       RA               SP4A:=LEFT SOURCE BYTE;
4510                                                                                      RA:=RA - #BYTES MOVED - 1
4511    0441    BAF3       OPA  ADD  RRZ             RB   SP2B INC       RB               UBA:=RIGHT TARGET BYTE;
4513                                                                                      RA:=RB + #BYTES MOVED + 1
4514    0442       UBA     SP4A JSB  D03G       DATA RC   SP2B INC       RC               WRITE LAST WORD, JSB FOR STACK DECREMENT;
4516                                                                                      RC:=RC + #BYTES MOVED + 1
4517    0443    BAF4       RA   ADD                  NZRO RB   SP2B ADD       RB               SKIP IF NOT FINISHED;
4519                                                                                      (S-1):=(S-1) + #BYTES MOVED
4520    0444                    JSB  D03G       UNC  RC   SP2B ADD       RC               JSB TO FINISH; (S-2):=(S-2) + #BYTES MOVED
4522    0445                    JSZ  IRDN       UNC  RA   SP2B SUB       RA   CF4B       JSB FOR INTERRUPT;
4524                                                                                      (S):=(S) - #BYTES MOVED, CF4B
4525            ********************************************************************
4526            *    move bytes aligned reverse                                    *
4527            ********************************************************************
4528            *
4529    0446    MBAR       UBA  ADD  RRZ  SP4A           SP4A JSB  BAR0 RH         F2    SP4A:=RIGHT TARGET BYTE;
4531                                                                                      JSB IF SOURCE PTR ODD, RH:=WORD COUNT
4532    0447               OPA  ADD  LLZ             RH        CAD  LSR  RH               UBA:=LEFT SOURCE BYTE; RH:=WORD COUNT - 1
4534    0448       UBA     SP4A IOR                  DATD      JSBI BAR1 SP2B       UNC   WRITE FIRST TARGET WORD; SP2B:=1, JSB
4536    0449    BAR0               JSB  BAR4        TEST RH   XR0  JSBS BAR2 RH         NEG   JSB IF INTERRUPT; JSB IF (WORD COUNT - 1)<0
4538    044A               OPA  ADD                  DATD 0002 SP2B ADDL      SP2B             WRITE TARGET WORD; SP2B:=SP2B+2
4540    044B    BAR1               JSB  BAR0        UNC  SP3B      CAD       SP3B RAB4        JSB BACK, SP3B:=SP3B-1, READ SOURCE WORD
4542    044C    BAR2       SP1B ADD                  ROD  RA   SP2B JSB  BAR4 RA         ZERO  READ LAST TARGET WORD;
4544                                                                                      (S):=(S) + #BYTES MOVED, JSB IF 0
4545    044D               OPA  ADD  RRZ  SP4A       RA   SP2B INC       RA               SP4A:=RIGHT SOURCE BYTE;
4547                                                                                      RA:=RA + #BYTES MOVED + 1
4548    044E    BAR3       OPA  ADD  LLZ             RB   SP2B CAD       RB               UBA:=LEFT TARGET BYTE
4550                                                                                      RB:=RB - #BYTES MOVED - 1
4551    044F       UBA     SP4A JSB  D03G       DATD RC   SP2B CAD       RC               WRITE LAST WORD, JSB FOR STACK DECREMENT;
4553                                                                                      RC:=RC - #BYTES MOVED - 1
4554    0450    BAR4       RA   ADD                  NZRO RB   SP2B SUB       RB               SKIP IF NOT FINISHED;
4556                                                                                      (S-1):=(S-1) - #BYTES MOVED
4557    0451                    JSB  D03G       UNC  RC   SP2B SUB       RC               JSB TO FINISH; (S-2):=(S-2) - #BYTES MOVED
4559    0452                    JSZ  IRDN       UNC  RA   SP2B ADD       RA   CF4B       JSB FOR INTERRUPT;
4561                                                                                      (S):=(S) + #BYTES MOVED, CF4B
```

```
RECORD   C S                ********** ALU A **********  ************** ALU B **************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4563           $LUT INSTR=CMPB(PB+):0 010 000 010 10,P,SR=3,ENTRY=CMBP
4564           $LUT INSTR=CMPB(DB+):0 010 000 010 11,DB,SR=3,ENTRY=CMBD
4565           *
4566     0453  CMBP            JSZ  ETO3       NFSS       RA   JSB  CMB0           ZERO  JSB TO EMPTY TOS TO 3 IF NOT SPLIT BANKS;
4568                                                                                      JSB IF BYTE COUNT = 0
4569     0454       RA          CAD       SP4A POS        DB   ADD       SP3B            SP4A:=BYTE COUNT - 1 IF >0; SP3B:=DB
4571     0455       RA          INC       SP4A SF3A       BNKP JSB  DBCC BKX4       UNC   SP4A:=BYTE COUNT + 1, SF3A;
4573                                                                                      BKX4:=BNKP, JSB TO FORM TARGET ADDR
4574     0456            RH     ADD       SP0A ROBD  0001 RB   ANDL      SP2B            SP0A:=TARGET WORD ADDR, READ;
4576                                                                                      UBB:=SP2B:=SOURCE POINTER(15)
4577     0457            RB     ADD  LSR  SP4A       UBB  SP4A ASR       SP1B            SP4A:=UBA:=SOURCE WORD POINTER;
4579                                                                                      UBB:=SP1B:=ADJUSTED WORD COUNT
4580     0458       UBA  PB     ADD       RH   ROP    PL   PB   SUB                     RH:=UBA:=ABSOLUTE SOURCE ADDR, READ;
4582                                                                                      UBB:=PL - PB
4583     0459       UBB  SP4A   UBNE                   UBA  SP1B ADD                    BOUNDS CHECK WORD POINTER >= (PL - PB);
4585                                                                                      UBB:=ENDING WORD ADDR
4586     045A       UBB  PB     UBNE                   PL   UBB  UBNE                   BOUNDS CHECK ENDING ADDR >= PB;
4588                                                                                      BOUNDS CHECK PL >= ENDING WORD ADDR
4589     045B                   JSB  CMB1       UNC             INC       XR0             JSB; XR0:=1
4591     045C  CMBD             JSZ  ETO3       NFSS       RA   JSB  CMB0           ZERO  JSB TO EMPTY TOS TO 3 IF NOT SPLIT BANKS;
4593                                                                                      JSB IF BYTE COUNT = 0
4594     045D       RA          CAD       SP4A POS        DB   ADD       SP3B            SP4A:=<BYTE COUNT>-1 IF >0; SP3B:=DB
4596     045E       RA          INC       SP4A SF3A       BNKD JSB  DBCC BKX4       UNC   SP4A:=<BYTE COUNT)+1, SF3A;
4598                                                                                      BKX4:=SOURCE BANK, JSB TO FORM TARGET ADDR
4599     045F            RH     ADD       SP0A ROBD             JSBI DBBC XR0       UNC   SP0A:=ABS TARGET WORD ADDR, READ INTO OPB
4601                                                                                      XR0:=1, JSB TO CHECK SOURCE ADDR
4602     0460  CMB1      RA     ADD       XR0  NF3A  RC   RB   XOR            LBF5        SAVE BYTE COUNT IN XR0;
4604                                                                                      F5:=SOURCE AND TARGET POINTERS SKEWED
4605     0461                   JSB  CMBR       UNC        SP2B SUB            HBF2 F5B  JSB IF REVERSE COMPARISON;
4607                                                                                      F2:=SOURCE PTR(15), SKIP IF NOT ALIGNED
4608     0462                   ADD             NF2             JSB  CBAF           UNC   SKIP IF EVEN SOURCE PTR AND JSB NOT TAKEN
4610                                                                                      JSB IF ALIGNED COMPARISON
4611     0463  CBSF             JSB  CSF2       UNC             ADD                       JSB IF ODD SOURCE PTR;
4613     0464  CSF1       OPA   ADD  LRZ                  OPB  ADD  RRZ                  UBA:=SOURCE BYTE; UBB:=TARGET BYTE
4615     0465       SPOA        INC       SP0A ROBD  UBB  UBA  JSBS CMBN SP1B       NZRO  READ NEXT TARGET WORD; COMPARE & SAVE SP1B
4617     0466                   JSB  CMBI       TEST  RA   XR0  JSBS CMBE RA         ZERO  INTERRUPT?, DEC <BYTE COUNT>, JSB IF 0
4619     0467  CSF2       OPA   ADD  RRZ                  OPB  ADD  LRZ                  UBA:=SOURCE BYTE; UBB:=TARGET BYTE
4621     0468            RH     INC       RH   ROX4 UBB  UBA  JSBS CMBN SP1B       NZRO  READ NEXT SOURCE WORD; COMPARE & SAVE SP1B
4623     0469                   JSB  CSF1       UNC   RA   XR0  JSBS CMBE RA         ZERO  LOOP BACK; DEC <BYTE COUNT>, JSB IF 0
4625     046A  CBAF       OPA   ADD  RRZ  SP4A                  JSB  CAF2               F2  SP4A:=SOURCE BYTE; JSB IF ODD SOURCE PTR
4627     046B  CAF1       OPA   ADD  LRZ                  OPB  ADD  LRZ                  UBA:=SOURCE BYTE; UBB:=TARGET BYTE
4629     046C            OPA    ADD  RRZ  SP4A       UBB  UBA  JSBS CMBN SP1B       NZRO  SP4A:=SOURCE BYTE; COMPARE & SAVE IN SP1B
4631     046D                   JSB  CMBI       TEST  RA   XR0  JSBS CMBE RA         ZERO  INTERRUPT?, DEC <BYTE COUNT>, JSB IF 0
4633     046E  CAF2 RH          INC       RH   ROX4 OPB  ADD  RRZ                       READ NEXT SOURCE WORD; UBB:=TARGET BYTE
4635     046F       SPOA        INC       SP0A ROBD  UBB  SP4A JSBS CMBN SP1B       NZRO  READ NEXT TARGET WORD; COMPARE & SAVE SP1B
4637     0470                   JSB  CAF1       UNC   RA   XR0  JSBS CMBE RA         ZERO  LOOP BACK; DEC <BYTE COUNT>, JSB IF 0
4639     0471  CMBR             ADD             F2              JSB  CBAR           UNC   SKIP IF ODD SOURCE PTR AND JSB NOT TAKEN
4641                                                                                      JSB IF ALIGNED COMPARISON
4642     0472  CBSR             JSB  CSR2       UNC             ADD                       JSB IF EVEN SOURCE PTR;
4644     0473  CSR1       OPA   ADD  RRZ                  OPB  ADD  LRZ                  UBA:=SOURCE BYTE, UBB:=TARGET BYTE
4646     0474       SPOA        CAD       SP0A ROBD  UBB  UBA  JSBS CMBN SP1B       NZRO  READ NEXT SOURCE WORD; COMPARE & SAVE SP1B
4648     0475                   JSB  CMBI       TEST  RA   XR0  JSB  CMBE RA         ZERO  INTERRUPT?, INC <BYTE COUNT>, JSB IF 0
4650     0476  CSR2       OPA   ADD  LRZ                  OPB  ADD  RRZ                  UBA:=SOURCE BYTE; UBB:=TARGET BYTE
4652     0477            RH     CAD       RH   ROX4 UBB  UBA  JSBS CMBN SP1B       NZRO  READ NEXT TARGET WORD; COMPARE & SAVE SP1B
4654     0478            UBA    JSB  CSR1       UNC   RA   XR0  JSB  CMBE RA         ZERO  LOOP BACK; INC <BYTE COUNT>, JSB IF 0
4656     0479  CBAR       OPA   ADD  LRZ  SP4A                  JSB  CAR2               NF2  SP4A:=SOURCE BYTE; JSB IF EVEN SOURCE PTR
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
4658	047A	CAR1		OPA	ADD	RRZ				OPB	ADD	RRZ				UBA:=SOURCE BYTE; UBB:=TARGET BYTE
4660	047B			OPA	ADD	LRZ	SP4A		UBB	UBA	JSBS	CMBN	SP1B		NZRO	SP4A:=SOURCE BYTE; COMPARE & SAVE IN SP1B
4662	047C				JSB	CMBI		TEST	RA	XRO	JSB	CMBE	RA		ZERO	INTERRUPT?, INC <BYTE COUNT>, JSB IF 0
4664	047D	CAR2	RH		CAD		RH	ROX4		OPB	ADD	LRZ				READ NEXT SOURCE WORD; UBB:=TARGET BYTE
4666	047E		SPOA		CAD		SPOA	ROBD	UBB	SP4A	JSBS	CMBN	SP1B		NZRO	READ NEXT TARGET WORD; COMPARE & SAVE SP1B
4668	047F				JSB	CAR1		UNC	RA	XRO	JSB	CMBE	RA		ZERO	LOOP BACK; INC <BYTE COUNT>, JSB IF 0
4670	0480	CMBE	RB	XRO	ADD		RB				ADD			CCA		SOURCE PTR:=SOURCE PTR + BYTE COUNT; SET CCE
4672	0481		RC	XRO	ADD		RC				JSB	D03S			UNC	TARGET PTR:=TARGET PTR + BYTE COUNT;TR
4674																JSB TO STACK DECREMENT
4675	0482	CMBN	RA	XRO	SUB					SP1B	ADD			CCA		UBA:=MINUS NUMBER OF BYTES COMPARED
4677																SET CONDITION CODES ON SP1B
4678	0483		RB	UBA	SUB		RB				ADD					SOURCE PTR:=SOURCE PTR +/- #BYTES COMPARED
4680	0484		RC	SREG	SUB		RC				JSB	D03S			UNC	TARGET PTR:=TARGET PTR +/- #BYTES COMPARED
4682																JSB FOR STACK DECREMENT
4683	0485	CMBI	RA	XRO	SUB						ADD					UBA:=NUMBER OF BYTES COMPARED
4685	0486		RB	UBA	SUB		RB				ADD					SOURCE PTR:=SOURCE PTR +/- #BYTES COMPARED
4687	0487		RC	SREG	SUB		RC				JSL	IRDN			UNC	TARGET PTR:=TARGET PTR +/- #BYTES COMPARED
4689																JSB TO INTERRUPT
4690	0488	CMBO			ADD			CCA			JSB	D03S			UNC	CCA ON 0, JSB STACK DECREMENT

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4693            ***************************************************************
4694            *    DBBC checks the DB relative starting byte address and the  *
4695            *    ending address for the source pointer in byte moves.  The  *
4696            *    absolute starting word address is returned in RH.  Bit 15  *
4697            *    of the source byte pointer is returned in SP2B.  Upon entry,*
4698            *    SP4A should contain the byte count minus 1 if positive and  *
4699            *    plus one if negative.  Addresses are checked against SM, and*
4700            *    if SM is exceeded, routine GSMC is entered for further      *
4701            *    checking.  Entered with SP3B = DB.                          *
4702            ***************************************************************
4703            *
4704    0489    DBBC      RB   ADD  LSR  RH   NFSS  SP3B DL   SUB            CF1  NCRY  RH:=SOURCE WORD PTR, SKIP IF NOT SPLIT BNKS;
4706                                                                                    SKIP IF NOT (DB >= DL), CF1
4707    048A              SM   ADD            CF1   Z    DB   SUB            CTF1       UBA:=SM, OVERRIDE CLEAR F1 IF SPLIT BANKS;
4709                                                                                    SET F1 IF NOT SPLIT STACK (SPLIT STACK IF
4710                                                                                      FSS OR (Z < DB) OR (DB < DL))
4711    048B      UBA  DB   SUB            NF1   0001 RB   ANDL      SP2B       UBA:=SM - DB, SKIP IF SPLIT STACK;
4713                                                                                    SP2B:=UBB:=SOURCE PTR(15)
4714    048C      RH   UBA  CAD            CTF1  UBB  SP4A ASR                   F1:=(SOURCE PTR > (SM - DB));
4716                                                                                    UBB:=ADJUSTED WORD COUNT
4717    048D            RB   ADD  LSR      F1HB  UBB  DB   ADD                   UBA:=WORD PTR + 32K IF NOT SPLIT STACK;
4719                                                                                    UBB:=WORD COUNT + DB
4720    048E      UBA  DB   ADD       RH   ROX4  UBA  UBB  ADD       SP1B       RH:=ABS START ADDR; SP1B:=ABS ENDING ADDR
4722    048F      UBB  SM   JSBC GSMC     CRRY  UBB  DL   BNDE            PSHR       JSB IF ENDING ADDR > SM;
4724                                                                                    BOUNDS CHECK ENDING ADDR >= DL,
4725                                                                                      PSHR AFTER RSB AND BEFORE POSSIBLE JSB
4726    0490      RH   SM   JSBC GSMC     CRRY  RH   DL   BNDE                  JSB IF STARTING ADDR > SM;
4728                                                                                    BOUNDS CHECK STARTING ADDR >= DL      (1002)
4729                                                                                    MOVED RSB DOWN TO AVOID S KILL   XXXX (1002)
4730    0491            ADD                                ADD            POPR TICB  (1002)
4732    0492            ADD                                ADD            RSB        (1002)
4734            ***************************************************************
4735            *    DBCC checks the DB relative starting byte address and the  *
4736            *    ending address for the target pointer in byte moves.  The  *
4737            *    absolute starting word address is returned in RH, and the  *
4738            *    ending address in SP1B.  A read is initiated into OPA for  *
4739            *    the starting address.  Upon entry, SP4A should contain the  *
4740            *    byte count minus 1 if positive and plus one if negative.    *
4741            *    Addresses are checked against SM, and if SM is exceeded,    *
4742            *    routine GSMC is entered for further checking.               *
4743            ***************************************************************
4744            *
4745    0493    DBCC      RC   ADD  LSR  RH   NFSS  SP3B DL   SUB            CF1  NCRY  RH:=TARGET WORD PTR, SKIP IF NOT SPLIT BNKS;
4747                                                                                    SKIP IF NOT (DB >= DL), CF1
4748    0494              SM   ADD            CF1   Z    DB   SUB            CTF1       UBA:=SM, OVERRIDE CLEAR F1 IF SPLIT BANKS;
4750                                                                                    SET F1 IF NOT SPLIT STACK (SPLIT STACK IF
4751                                                                                      FSS OR (Z < DB) OR (DB < DL))
4752    0495      UBA  DB   SUB            NF1   0001 RC   ANDL                  UBA:=SM - DB, SKIP IF SPLIT STACK;
4754                                                                                    UBB:=TARGET PTR(15)
4755    0496      RH   UBA  CAD            CTF1  UBB  SP4A ASR                   F1:=(TARGET PTR > (SM - DB));
4757                                                                                    UBB:=ADJUSTED WORD COUNT
4758    0497            RC   ADD  LSR      F1HB  UBB  DB   ADD                   UBA:=WORD PTR + 32K IF NOT SPLIT STACK;
4760                                                                                    UBB:=WORD COUNT + DB
4761    0498      UBA  DB   ADD       RH   ROD   UBA  UBB  ADD       SP1B       RH:=ABS STARTING ADDR; SP1B:=ABS ENDING ADDR
4763    0499      UBB  SM   JSBC GSMC     CRRY  UBB  DL   BNDE            PSHR       JSB IF ENDING ADDR > SM;
4765                                                                                    BOUNDS CHECK ENDING ADDR >= DL
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4766                                                                                    PSHR AFTER RSB AND BEFORE POSSIBLE JSB
4767     049A        RH   SM   JSBC GSMC      CRRY  RH   DL   BNDE                     JSB IF STARTING ADDR > SM;
4769                                                                                    BOUNDS CHECK STARTING ADDR >= DL
4770                                                                                     PUT RSB BACK (ORIGINALLY 2511)        (1002)
4771     049B                    ADD                              ADD           POPR TICB  RESTORE WHERE TO RETURN               (1002)
4773                                                                                    TICB SLOWS RSB TO RANK2 SO POPR TAKES (1002)
4774     049C                    ADD                              ADD                RSB  AND RETURN WITH OUT S TRASH            (1002)
4776             ********************************************************************
4777             *    This routine jumps to BNDV if not priviledged or if not split  *
4778             *    stack and the starting and ending addresses "span" (SM + 1).   *
4779             *    In other words, the move would go through its parameters.      *
4780             *    The problem with this occurs in the event an interrupt is      *
4781             *    serviced immediately after moving the bytes at SM.  After      *
4782             *    returning from the interrupt, then effective byte address      *
4783             *    would be mapped into the DL to DB area.                        *
4784             ********************************************************************
4785             *
4786     049D    GSMC            JSB  FSSR       FSS   Z    RH   SUB           POPR CRRY  JSB IF SPLIT BANKS;
4788                                                                                    POPR, SKIP IF Z >= STARTING ADDR
4789     049E            STA  JSZ  BNDV       POS   Z    SP1B SUB                NCRY  JSB FOR BNDV IF NOT PRIVILEDGED;
4791                                                                                    SKIP IF NOT (Z >= ENDING ADDR)
4792     049F        RH  DL   BNDE                                 JSZ  BNDV      UNC   BOUNDS CHECK STARTING ADDR >= DL;
4794                                                                                    JSB FOR BNDV IF NOT ((STARTING ADDR > Z)
4795                                                                                      AND (ENDING ADDR > Z)); DROPPED RSB (2527)
4796     04A0    FSSR            ADD                 RSB           ADD                  RSB IF SPLIT BANKS TO LINE AFTER CALL
4798                                                                                      TO DBBC OR DBCC
```

RECORD NO	C S ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
4800																
4801		*	A new trap (#MWOL) is implemented in this instruction. The													*
4802		*	trap occurs if the source byte pointer ever equals the													*
4803		*	starting value of the target pointer, except where the two													*
4804		*	are the same to begin with. In the event of this trap, the													*
4805		*	last word pointed at by the value in S-1 is not moved													*
4806																
4807		*														
4808		$LUT INSTR=MVBW:0 010 000 010 0xx xxx, DB, SR=2, ENTRY=MVBW														
4809		*														
4810	04A1	MVBW		DB	ADD			NFSS	Z	DB	SUB				NCRY	UBA := DB, skip if not split banks;
4812																Skip if DB > Z
4813	04A2		SR	SM	ADD			CF1	UBA	DL	SUB			CTF1		UBA := S, clr F1 if split banks, (overrides
4815																set F1 in ALUB); Set F1 if not split stack
4816	04A3		UBA	DB	SUB		RH	NF1		RA	ADD	LSR				RH := UBA := S - 1 - DB, skip if split stack
4818																; UBB := source word pointer
4819	04A4		UBB	UBA	SUB			CTF1		RB	ADD	LSR			NF1	If not split stack F1 := swptr > S - 2;
4821																UBB := target wrd ptr, skip if split stack
4822	04A5			RA	ADD	LSR		F1HB	UBB	RH	SUB			CTF1		UBA := source word ptr + 32K in necessary;
4824																If not split stack F1 := twptr > S - 2
4825	04A6			RB	ADD	LSR	SPOA	F1HB	UBA	DB	ADD		SP1B	ROD		SPOA := target word ptr + 32K if necessary;
4827																SP1B := absolute source address, read
4828	04A7				JSZ	ETO2		NFSS	UBB	DL	BNDE					Empty TOS to 2 if not split banks;
4830																Bounds check source address >= DL
4831	04A8		SPOA	DB	ADD		SP4A	ROD	RA	RB	XOR			CTF1	LBF5	SP4A := absolute target address, read;
4833																Set F5 if move is skewed
4834	04A9		0004	CIR	ANDL				UBA	DL	BNDE					UBA := upshift bit;
4836																Bounds check target address >= DL
4837	04AA			OPA	ADD	LRZ	RH		UBA		CAD			CTF1	F5B	RH := left target byte;
4839																F1 := upshift bit, skip if skewed move
4840	04AB			SM	INC		SPOA				JSB	MBWA			UNC	SPOA := SM + 1; Jump if aligned move

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4843          ********************************************************************
4844          *    skewed case                                                   *
4845          ********************************************************************
4846          *
4847    04AC  MBWS       RA   JSB  MWSE      EVEN       RA   ADD       SP2B           Jump if source pointer is even;
4849                                                                                     SP2B := source pointer
4850    04AD       FFFF SP4A ADDL      SP4A             OPB  ADD  RRZ  RH   CCB        SP4A := absolute target addr - 1;
4852                                                                                     RH := right source byte, set CCB
4853    04AE  MWSO            JSB  MBWI      TEST       SP1B INC       SP1B ROD       Jump if interrupt;
4855                                                                                     SP1B := addr of next source word, read
4856    04AF            SP4A INC       SP4A NF1   RB   RA   JSZC MWOL           ZERO  SP4A := addr of next target word, skip if
4858                                                                                     not upshift; Jump if moving into source
4859    04B0       RA        JSB  USHT      UNC        RA   JSBI MBWX RA         NWIL  Jump if 'while' not satisfied, else jsb if
4861                                                                                     upshift; Increment source address pointer
4862    04B1  MWSE SPOA SP1B JSBC WBC1      NCRY       OPB  ADD  LRZ  RH              Jsb if src addr > SM + 1 (check before left
4864                                                                                     byte; RH := UBB := left source byte
4865    04B2       RH        ADD  RLZ  XRO  NF1        UBB  ADD            CCB       XRA0.(0:8) := old rt src byte, skip if not
4867                                                                                     upshift; Set CCB on new left source byte
4868    04B3            JSB  USHT      UNC   RB   RA   JSZC MWOL           ZERO  Jump if overlapped move else
4870                                                                                     jsb if upshift
4871    04B4       SPOA SP4A JSBS WBC4      NCRY       RA   JSBI MBWX RA         NWIL  Jump if 'while' not satisfied, else jsb if
4873                                                                                     tar addr > SM + 1, inc src addr ptr
4874    04B5       RH   XRO  JSB  MWSO      DATA       OPB  ADD  RRZ  RH   CCB        Write target word, loop back;
4876                                                                                     RH := right source byte, set CCB
4877          *
4878          ********************************************************************
4879          *    aligned case                                                  *
4880          ********************************************************************
4881          *
4882    04B6  MBWA       RA   JSB  MAOE      ODD        RA   ADD       SP2B           Jump if source pointer is odd;
4884                                                                                     SP2B := source pointer
4885    04B7       FFFF SP4A ADDL      SP4A             OPB  ADD  LRZ  RH              Decrement target address;
4887                                                                                     RH := UBB := left source byte
4888    04B8  MWAE SPOA SP1B JSBC WBC1      NCRY       UBB  ADD            CCB       Jump if src addr > SM + 1 (check before left
4890                                                                                     byte); Set CCB on left source byte
4891    04B9            SP4A INC       SP4A NF1   RB   RA   JSZC MWOL           ZERO  Increment target word addr, skip if not
4893                                                                                     upshift; Jump if overlapped move
4894    04BA                 JSB  USHT      UNC        RA   JSBI MBWX RA         NWIL  Jump if 'while' not satisfied else jsb if
4896                                                                                     upshift;  Increment source pointer
4897    04BB                 JSB  MBWI      TEST       OPB  ADD  RRZ  RH   CCB        Jump if interrupt pending;
4899                                                                                     RH := UBB := right source byte, set CCB
4900    04BC  MWAO RH             ADD  RLZ  XRO  NF1        SP1B INC       SP1B ROD       XRA0.(0:8) := right source byte, skip if not
4902                                                                                     upshift; Increment source addr ptr, read
4903    04BD       RA        JSBC USHT      UNC   RB   RA   JSZC MWOL           ZERO  UBA := (S) in case TEST; Jump if overlapped
4905                                                                                     else jsb if upshift
4906    04BE       SPOA SP4A JSBS WBC4      NCRY       RA   JSBI MBWX RA         NWIL  Jump if 'while' not satisfied else jsb if
4908                                                                                     next tar addr > SM + 1; Inc source address
4909    04BF       RH   XRO  JSB  MWAE      DATA       OPB  ADD  LRZ  RH              Write target word, loop back;
4911                                                                                     RH := UBB := left source byte
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
4913			**********													
4914			* common routines													*
4915			**********													
4916			*													
4917	04C0	MBWX	RA	SP2B	CAD						ADD					UBA := number of bytes moved
4919	04C1			SP4A	ADD			ROBD	RB	UBA	ADD		RB	LBF5		Read last target word; Inc target ptr by # of bytes moved, set F5 if odd
4921																
4922	04C2		RA		CAD		RA			SM	JSBI	D03S	SP3B		NF5B	Dec source ptr; SP3B := SM + 1, jump to stack decrement if target pointer is even
4924																
4925	04C3		SP0A	SP4A	BNDG					OPB	ADD	RRZ				Bounds violation if last tar addr > SM + 1; UBB := right target byte
4927																
4928	04C4		UBB	XR0	JSB	D03S		DATA	SP3B	SP1B	BNDE					Write last byte, jump for stack decrement; Bounds violation if last src addr > SM + 1
4930																
4931	04C5	MAOE			JSB	MWAO		UNC		OPB	ADD	RRZ	RH	CCB		Enter here for aligned/odd case to skip 1st test for interrupts; RH := right source byte, set CCB
4933																
4934																
4935	04C6	MBWI	UBA	SP2B	SUB					UBA	ADD		RA			UBA := number of bytes moved; (S) := updated (S)
4937																
4938	04C7		RB	UBA	ADD		RB				JSZ	IRDN			UNC	Inc target ptr by # of bytes moved; Enter interrupt handler after FAKENEXT
4940																
4941	04C8	USHT	FFDF	RH	ANDL				0040	RH	ANDL				ZERO	UBA := upshifted character; Skip if not alpha, i.e., <= %100
4943																
4944	04C9				ADD			RSB		UBA	ADD		RH			Return; RH := upshifted character if alpha
4946																
4947	04CA	WBC1			CAD		SP0A	FSS	Z		JSZ	BNDV			NPRV	SP0A := -1 as new limit, skip if split banks UBB := Z, bounds violation if non-priv
4949																
4950	04CB		UBB	SP1B	JSZS	BNDV		CRRY		RH	ADD				RSB	Bounds viol'n if Z >= src addr & not split banks (and addr > SM); UBB := RH, return
4952																
4953	04CC	WBC4			CAD		SP0A	FSS	Z		JSZ	BNDV			NPRV	SP0A := -1 as new limit, skip if split banks UBB := Z, bounds violation if non-priv
4955																
4956	04CD		UBB	SP1B	JSZS	BNDV		CRRY			ADD				RSB	Bounds viol'n if Z >= tar addr & not split banks (and addr > SM); Return
4958																

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

4960            ****************************************************************
4961            *    Scan While (SCW) Instruction - $MTSW is the entry point for     *
4962            *    the Scan While and Move to Data Segment (MTDS) instructions     *
4963            *    The SR preadjust is set to 2 for SCW.  The TOS registers        *
4964            *    are emptied unless DB-bank is not equal to S-bank (FSS).        *
4965            *    The address to be scanned may be greater than SM in             *
4966            *    privileged mode if also greater than Z.  Allowing the           *
4967            *    address to "go through" SM + 1 except in split stack            *
4968            *    could cause indeterminate results if an interrupt is            *
4969            *    serviced after the word at SM is scanned.  Upon return          *
4970            *    from interrupt the next address would be mapped into the        *
4971            *    DL to DB area.                                                  *
4972            ****************************************************************
4973            *
4974    04CE    SCW        DB   ADD             NFSS  Z    DB   SUB                NCRY  UBA:=DB, SKIP IF NOT SPLIT BANKS;
4976                                                                                      SKIP IF NOT (Z >= DB)
4977    04CF         SR    SM   ADD             CF1   UBA  DL   SUB          CTF1        UBA:=SR + SM, CF1 IF SPLIT BANKS;
4979                                                                                      SET F1 IF NOT SPLIT STACK (SPLIT STACK
4980                                                                                        IF FSS OR (Z < DB) OR (DB < DL))
4981    04D0             RB   ADD  LSR       NF1   UBA  DB   CAD                NF1   UBA:=WORD POINTER, SKIP IF SPLIT STACK;
4983                                                                                      UBB:=(SR + SM) - DB - 1, skip if split stack
4984    04D1                  JSZ  ETO2      NFSS  UBA  UBB  SUB          CTF1        JSB TO EMPTY TOS TO 2 IF NOT SPLIT BANKS;
4986                                                                                      F1:=WORD POINTER >= (S - 1 - DB)
4987                                                                                        (WORD ADDR > (S - 2));
4988    04D2             RB   ADD  LSR       F1HB       RA   ADD  RRZ  SP2B          UBA:=WORD PTR + 32K IF NOT SPLIT STACK;
4990                                                                                      SP2B:=TEST CHARACTER
4991    04D3             RA   ADD  LRZ  XR0  FSS   UBA  DB   ADD       SP3B ROAD     XR0:=TERMINAL CHARACTER, SKIP IF FSS;
4993                                                                                      SP3B:=ABSOLUTE WORD ADDR, READ
4994    04D4        UBB  SM   JSBC SWUB      CRRY  UBB  DL   BNDE                     JSB IF (ADDR > SM) AND NOT SPLIT BANKS;
4996                                                                                      BOUNDS CHECK ADDR >= DL
4997    04D5             RB   JSB  SCW1      ODD        SM   INC       RH            UBA:=SOURCE BYTE POINTER, JSB IF ODD;
4999                                                                                      RH=SM + 1
5000    04D6    SCW0     OPA  ADD  LRZ  SP4A            UBA  ADD       RB       NF2  SP4A:=UBA:=LEFT SOURCE BYTE;
5002                                                                                      INCREMENT BYTE PTR, SKIP IF SPLIT BANKS
5003    04D7        UBA  SP2B JSBS SCW2      NZRO  SP3B RH   JSBS SBCK          CRRY  JSB IF SOURCE BYTE <> TEST CHARACTER;
5005                                                                                      JSB IF WORD ADDR >= (SM + 1) AND
5006                                                                                        NOT SPLIT BANKS
5007    04D8             JSZ  IRDN      TEST  RB         INC       RB                 JSB IF INTERRUPT; INCREMENT BYTE PTR
5009    04D9    SCW1     OPA  ADD  RRZ  SP4A       SP3B       INC       SP3B ROAD     SP4A:=UBA:=RIGHT SOURCE BYTE,
5011                                                                                      INCREMENT SOURCE WORD ADDR, READ
5012    04DA             RB   JSBI SCW0      UNC   UBA  SP2B JSBS SCW2          NZRO  UBA:=INCREMENTED BYTE PTR;
5014                                                                                      JSB IF SOURCE BYTE <> TEST CHARACTER
5015    04DB    SCW2     XR0  ADD                       SP4A ADD            CCB       UBA:=TERMINAL CHARACTER;
5017                                                                                      SET CCB ON LAST CHARACTER SCANNED
5018    04DC             ADD                       UBA  SP4A XOR                SCRY ZERO  ; SKIP IF TERMINAL CHAR = LAST CHAR, SCRY
5020    04DD             JSB  D03G      UNC                 ADD                 CCRY       JSB FOR STACK DECREMENT;
5022                                                                                      CLEAR CARRY IF TERMINAL CHAR <> LAST CHAR
5023    04DE    SWUB          ADD             RSB                 JSZ  BNDV          NPRV  RSB UNLESS...; JSB FOR BNDV IN NPRV
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5026           ********************************************************************
5027           *    Entered from #MFSU, which is the entry point for both the     *
5028           *    Move From Data Segment (MFDS) and the Scan Until (SCU)        *
5029           *    instructions.  The SR preadjust is set to 2.  The TOS         *
5030           *    registers are emptied unless in split-bank mode (FSS).  The   *
5031           *    address to be scanned may be greater than SM in privileged    *
5032           *    mode if also greater than Z.  Allowing the address to 'go     *
5033           *    through' SM + 1 except in split stack mode could cause        *
5034           *    indeterminate results if an interrupt is serviced after the   *
5035           *    word at SM is scanned.  Upon return from the interrupt the    *
5036           *    next address would be mapped into the DL to DB area.          *
5037           ********************************************************************
5038           *
5039    04DF   SCU        DB   ADD             NFSS  Z     DB   SUB                NCRY  UBA := DB, skip if not split banks;
5041                                                                                      Skip if not (Z >= DB)
5042    04E0        SR    SM   ADD             CF1   UBA   DL   SUB          CTF1        UBA := S, clear F1 if split banks;  Set F1
5044                                                                                      if not split stack - not (FSS or (Z < DB)
5045                                                                                      or (DB < DL))
5046    04E1              RB   ADD   LSR       NF1   UBA   DB   CAD                NF1   UBA := word pointer, skip if split stack;
5048                                                                                      UBB := S - DB - 1, skip if split stack
5049    04E2                   JSZ   ETO2      NFSS  UBA   UBB  SUB          CTF1        Empty TOS to 2 if not split banks;
5051                                                                                      F1 := word pointer > (S - 2)
5052    04E3              RB   ADD   LSR       F1HB        RA   ADD   LRZ  XRO         UBA := word pointer + 32K if not split stack
5054                                                                                      XRBO := terminal character
5055    04E4              RA   ADD   RRZ  XRO  FSS   UBA   DB   ADD         SP3B ROAD   XRAD := test character, skip if split banks;
5057                                                                                      SP3B := absolute word address, read
5058    04E5        UBB   SM   JSBC  SWUB      CRRY        SM   INC         RH          Check for BNDV if address > SM and not split
5060                                                                                      banks;  RH := SM + 1
5061    04E6              RB   JSB   SCU1      ODD   SP3B  DL   BNDE                NF2   Jump if right byte;  Bounds check
5063                                                                                      address >= DL, skip if split banks
5064    04E7   SCU0       OPA  ADD   LRZ             SP3B  RH   JSBS  SBCK         CRRY  UBA := left source byte;  Check for BNDV if
5066                                                                                      address >= SM + 1 and not split banks
5067    04E8        UBA   XRO  JSBS  SCU2      ZERO  UBA   XRO  JSBS  SCU3         ZERO  JSB IF SOURCE BYTE = TEST CHARACTER,
5069                                                                                      JSB IF SOURCE BYTE = TERMINAL CHARACTER,
5070                                                                                      TERMINAL CHARACTER HAS PRECEDENCE
5071    04E9                   JSZ   IRDN      TEST  RB          INC         RB          JSB IF INTERRUPT; INCREMENT BYTE PTR
5073    04EA   SCU1       OPA  ADD   RRZ             SP3B        INC         SP3B ROAD   UBA:=RIGHT SOURCE BYTE;
5075                                                                                      INCREMENT WORD ADDR, READ
5076    04EB        UBA   XRO  JSBS  SCU2      ZERO  UBA   XRO  JSBS  SCU3         ZERO  JSB IF SOURCE BYTE = TEST CHARACTER;
5078                                                                                      JSB IF SOURCE BYTE = TERMINAL CHARACTER,
5079                                                                                      TERMINAL CHAR TEST HAS PRECEDENCE
5080    04EC                   JSB   SCU0      UNC   RB          INC         RB    NF2   LOOP BACK;
5082                                                                                      INCREMENT BYTE PTR, SKIP IF SPLIT BANKS
5083    04ED   SCU2             JSB   D03S      UNC               ADD               CCRY  JSB FOR STACK DECREMENT; CCRY
5085    04EE   SCU3             JSB   D03S      UNC               ADD               SCRY  JSB FOR STACK DECREMENT; SCRY
5087    04EF   SBCK             CAD        RH         Z         JSZ   BNDV          NPRV  RH:=%FFFF AS NEW LIMIT;
5089                                                                                      UBB:=Z, BOUNDS VIOLATION IF NOT PRIVILEGED
5090    04F0              OPA  ADD   LRZ       RSB   SP3B  UBB  JSZC  BNDV         NCRY  UBA:=LEFT SOURCE BYTE, RSB;
5092                                                                                      BOUNDS VIOLATION IF NOT (ADDR > Z)
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5094           ********************************************************************
5095           *    Stack Decrement Routines for Moves                            *
5096           *                                                                  *
5097           *    Decrements from 0 to 3 words from TOS after move instructions. *
5098           ********************************************************************
5099           *
5100     04F1  D03S 0003 CIR  ANDL                          JSB  D03G           SRG2  UBA:=STACK DECREMENT BITS OUT OF CIR;
5102                                                                                   JSB IF SR > 2
5103     04F2       SR   UBA  JSBS D03L      NCRY          ADD                       JSB IF NOT (SR >= STACK DECREMENT);
5105     04F3  D03G      CIR  ADD            EVEN      UBA ADD                       SKIP IF NOT STACK DECREMENT OF 1;
5107                                                                                   UBB:=(SR - STACK DECREMENT)
5108     04F4            UBB  ADD            EPOP      UBA ADD  LSR          EVEN  UBA:=(SR - STACK DECREMENT), EPOP;
5110                                                                                   SKIP IF NOT STACK DECREMENT BY 2
5111     04F5                 JSZ  NEXT      UNC           ADD           EPP2        JSB FOR NEXT, EPP2
5113     04F6  D03L UBA  SM   ADD       SP4A               ADD                       SP4A:=(SR + SM) - STACK DECREMENT;
5115     04F7       UBA  Q    JSZS STUN      NCRY          ADD                       JSB FOR STUN IF NOT (NEW SM >= Q)
5117     04F8                 JSZ  NEXT      UNC       SP4A ADD       SM   CLSR        JSB FOR NEXT IN ONE CLOCK;
5119                                                                                   SM:=S - STACK DECREMENT, CLSR
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5121            ************************************************************
5122            *                                                          *
5123            *    BFD or MT Channel Program Build Module                *
5124            *                                                          *
5125            ************************************************************
5126            *
5127    04F9    NSDV CBFD      ADDL LBL  SP0A       0002 XR8  SUBL                ZERO  GO TO MTCP IF NOT BFD OR MT CH. PROG  (2635)
5129    04FA         0341      ADDL      XR1                 JSL  MTCP            UNC   COPY BFD DISC PROGRAM; COPY MT CHANNEL PROG
5131    04FB         0313      ADDL                          ADD       BKX4              SET PROGRAM STARTING ADDRESS; BKX4 := 0
5133    04FC         FFF2      ADDL      RD        UBA       ADD            WRX4         SET LENGTH; SET WRITE ADDRESS
5135    04FD         0002      ADDL      XR7                 JSL  UNPK            UNC   SET SECTOR ADDRESS; UNPACK CHANNEL PROGRAM
5137    04FE         0331      ADDL                    XR5   ADD  LSL                   UBA <- MEM. ADR. FOR READ SIZE         (2635)
5139                                                                                    UBB <- XFER SIZE IN BYTES              (2635)
5140    04FF         UBB       ADD                   UBA     ADD            WRS          SET UP FOR MEM. WRITE                  (2635)
5142    0500                   ADD                   UBA     ADD            DATA         WRITE CORRECT SIZE IN MEM.             (2635)
5144    0501         0359      ADDL                  FFFC    ADDL      RD               SET PROGRAM ADDRESS FOR DISC; SET LENGTH
5146                                                                                    WRITE OUT BFD STATUS MASK
5147    0502         SBFD      ADDL LBL  SP0A        UBA     ADD            WRX4         SET MESSAGE ADDRESS;SET ADDRESS
5149    0503         0346      ADDL      XR1                 JSL  UNPK            UNC   SET DISC ADDRESS; UNPACK MESSAGES
5151    0504                   ADD                           JSL  CLCT BKX3       UNC   FINISH BUILDING CHANNEL PROGRAM
5153            ************************************************************
5154            *    MWP is the entry point for a P relative source, and MWD is     *
5155            *    the entry point for a DB relative source.  The instruction     *
5156            *    breaks into four cases, two for moves in the direction of      *
5157            *    ascending addresses and two for moves in the direction of      *
5158            *    descending addresses.  For each direction, there are two       *
5159            *    cases, the normal case and the case where the move over-       *
5160            *    writes itself.  In the latter case, the source address of      *
5161            *    the next word moved is the target address of the preceding     *
5162            *    word.  The first word of an overlapped move is propogated      *
5163            *    throughout the length of the move without additional reads.    *
5164            *    Pending interrupts are tested for every 255 words moved.       *
5165            ************************************************************
5166            *
5167            $LUT INSTR=MOVE(PB+):0 010 000 000 00x xxx, P, SR=3, ENTRY=MWP
5168            $LUT INSTR=MOVE(DB+):0 010 000 000 01x xxx, DB, SR=3, ENTRY=MWD
5169            *
5170            *
5171    0505    MWP  RA        CAD       SP4A POS            RA   JSL  MD03 XR0       ZERO  SP4A:=WORD COUNT - 1, SKIP IF > 0;
5173                                                                                    XR0:=WORD COUNT, JSB IF = 0
5174    0506         RA        INC       SP4A SF2  RB   PB   ADD       SP3B              IF WORD COUNT < 0 THEN SP4A:=COUNT + 1, SF2;
5176                                                                                    SP3B:=ABSOLUTE SOURCE ADDR
5177    0507         UBB  PB   BNDE                    PL   UBB  BNDE                  BOUNDS CHECK ADDR >= PB; CHECK PL >= ADDR
5179    0508         RC   DB   ADD       RH          SP3B SP4A ADD                   RH:=ABSOLUTE TARGET ADDR;
5181                                                                                    UBB:=ENDING SOURCE ADDR
5182    0509         UBB  PB   BNDE                    PL   UBB  BNDE                  BOUNDS CHECK ADDR >= PB; CHECK PL >= ADDR
5184    050A         RH   SP4A ADD       RH          RH   BNKP ADD       BKX3         RH:=ENDING TARGET ADDR; BKX3:=BNKP
5186    050B         RH   SM   JSBC MWE3        CRRY  RH   DL   BNDE      SP2B         JSB IF ADDR > SM;
5188                                                                                    BOUNDS CHECK ADDR >= DL, SP2B:=ADDR - DL
5189    050C         RH   SM   JSBC MWE3        CRRY  RH   DL   BNDE      SP2B         DITTO; DITTO
5191    050D         RC   DB   ADD             WRD   SP3B      ADD            ROA3 NF2  WRITE AT (S-2) + DB;
5193                                                                                    READ FIRST WORD, SKIP IF FORWARD MOVE
5194    050E                   JSB  MWF1        UNC                 JSB  MWR1            UNC   JSB FOR FORWARD MOVE; JSB FOR REVERSE MOVE
5196    050F    MWD  RA        CAD       SP4A POS            RA   JSB  MD03 XR0       ZERO  SP4A:=WORD COUNT - 1, SKIP IF > 0;
5198                                                                                    XR0:=WORD COUNT, JSB IF = 0
5199    0510         RA        INC       SP4A SF2  RB   DB   ADD       SP3B              IF WORD COUNT < 0 THEN SP4A:=COUNT + 1, SF2;
```

RECORD NO	C S ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
5201																SP3B:=ABSOLUTE SOURCE ADDR
5202	0511		UBB	SM	JSBC	MWE3		CRRY	UBB	DL	BNDE		SP2B			JSB IF ADDR > SM;
5204																BOUNDS CHECK ADDR >= DL, SP2B:=ADDR - DL
5205	0512		RC	DB	ADD		RH		SP3B	SP4A	ADD					RH:=ABS TARGET ADDR; UBB:=ENDING SOURCE ADDR
5207	0513		UBB	SM	JSBC	MWE3		CRRY	UBB	DL	BNDE		SP2B			JSB IF ADDR > SM;
5209																BOUNDS CHECK ADDR >= DL, SP2B:=ADDR - DL
5210	0514		RH	SP4A	ADD		RH			BNKD	ADD		BKX3			RH:=ENDING TARGET ADDR; BKX3:=BNKD
5212	0515	MW0	RH	SM	JSBC	MWE3		CRRY	RH	DL	BNDE		SP2B			JSB IF ADDR > SM;
5214																BOUNDS CHECK ADDR >= DL, SP2B:=ADDR - DL
5215	0516		RH	SM	JSBC	MWE3		CRRY	RH	DL	BNDE		SP2B			DITTO; DITTO
5217	0517	MW1	RC	DB	ADD			WRD			JSB	MWR			F2	SEND WRITE ADDR; JSB IF REVERSE MOVE

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5220           ****************************************************************
5221           *    Move Words Forward, (in the direction of ascending addresses)  *
5222           ****************************************************************
5223           *
5224     0518  MWF   RC   RB   JSBC MWBF       ZERO  SP3B       ADD            ROA3        JSB IF SOURCE AND TARGET PTRS OVERLAP BY 1;
5226                                                                                        READ FIRST SOURCE WORD
5227     0519  MWF1                JSBI MWIX       TEST  SP3B       INC        SP3B ROA3   UBA:=1, JSB IF INTERRUPT;
5229                                                                                        READ NEXT SOURCE WORD, SP3B:=SP3B + 1
5230     051A              OPA   JSB  *-1         DATA  RA   UBA   JSBS MWX  RA          ZERO  WRITE TARGET WORD, LOOP BACK;
5232                                                                                        WORD COUNT:=WORD COUNT - 1, JSB IF = 0
5233     051B  MWX           ADD                        RB   XR0   ADD        RB              ; (S-1):=(S-1) + WORD COUNT (MAY BE < 0)
5235     051C                JSB  MD03       UNC   RC   XR0   ADD        RC              JSB FOR STACK DECREMENT;
5237                                                                                        (S-2):=(S-2) + WORD COUNT
5238     051D  MWIX          ADD                        RA   XR0   SUB                  ; UBB:=WORD COUNT - ORIGINAL WORD COUNT
5240     051E                ADD                        RB   UBB   SUB        RB        ; (S-1):=(S-1) + NUMBER OF WORDS MOVED
5242     051F                JSZ  IRDN       UNC   RC   SREG  SUB        RC              JSB TO INTERRUPT HANDLER;
5244                                                                                        (S-2):=(S-2) + NUMBER OF WORDS MOVED
5245           ****************************************************************
5246           *    Move Words Blanking Forward, (overlapped forward move)        *
5247           ****************************************************************
5248           *
5249     0520  MWBF          ADD                        RA   ADD        CTR             ; CTR:=WORD COUNT.(8:8)
5251     0521                JSZ  IRDN       TEST                 REPC       SP1B DCTR CTR0  JSB IF INTERRUPT;
5253                                                                                        SP1B:=0, REPC IF CTR <> 0, DCTR
5254     0522              OPA   ADD          DATA  UBB        INC        SP1B DCTR CTR0  WRITE SOURCE WORD;
5256                                                                                        SP1B.=UBB + 1, DCTR, END REPEAT IF CTR = 0
5257     0523        RB    SP1B ADD    RB           RA   SP1B SUB  RRZ  CTR             (S-1):=(S-1) + NUMBER OF WORDS MOVED;
5259                                                                                        UBB:=CTR;=REMAINING WORD COUNT.(8:8)
5260     0524        RC    SP1B ADD    RC                UBB  ADD                  NZRO  (S-2):=(S-2) + NUMBER OF WORDS MOVED;
5262                                                                                        SKIP IF CTR <> 0
5263     0525        RA    SP1B SUB    RA                     ADD            DCTR        (S):=(S) - # OF WORDS MOVED;
5265                                                                                        IF CTR=0 THEN CTR:=FF
5266     0526                JSB  MD03       UNC        UBA  JSB  *-5             NZRO  JSB FOR STACK DEC; JSB IF REMAINING COUNT
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5269            ********************************************************************
5270            *    Move Words Reverse, (in the direction of descending addresses) *
5271            ********************************************************************
5272            *
5273    0527    MWR   RB   RC   JSBC MWBR       ZERO  SP3B       ADD            ROA3        JSB IF SOURCE AND TARGET PTRS OVERLAP BY 1,
5275                                                                                        READ FIRST SOURCE WORD
5276    0528    MWR1                JSBI MWIX   TEST  SP3B       CAD       SP3B ROA3        UBA:=1, JSB IF INTERRUPT;
5278                                                                                        READ NEXT SOURCE WORD, SP3B:=SP3B - 1
5279    0529              OPA  JSB  *-1       DATD  RA   UBA  JSB  MWX  RA         ZERO    WRITE TARGET WORD, LOOP BACK;
5281                                                                                        WORD COUNT:=WORD COUNT - 1, JSB IF = 0
5282            ********************************************************************
5283            *    Move Words Blanking Reverse, (overlapped reverse move)        *
5284            ********************************************************************
5285            *
5286    052A    MWBR                ADD                       RA   ADD       CTR             ; CTR:=WORD COUNT.(8:8)
5288    052B                        JSZ  IRDN       TEST                    REPC      SP1B ICTR CTR0  JSB IF INTERRUPT;
5290                                                                                        SP1B:=0, REPC IF CTR <> 0, ICTR
5291    052C              OPA  ADD              DATD  UBB        CAD       SP1B ICTR CTR0  WRITE SOURCE WORD;
5293                                                                                        SP1B:=UBB - 1, ICTR, END REPEAT IF CTR = 0
5294    052D        RB    SP1B ADD        RB          RA   SP1B SUB  RRZ  CTR             RB:=RB - NUMBER OF WORDS MOVED;
5296                                                                                        UBB:=CTR:=REMAINING WORD COUNT.(8:8)
5297    052E        RC    SP1B ADD        RC                   UBB  ADD            NZRO  RC:=RC - NUMBER OF WORDS MOVED;
5299                                                                                        SKIP IF CTR <> 0
5300    052F        RA    SP1B SUB        RA                        ADD       ICTR      RA:=RA + NUMBER OF WORDS MOVED;
5302                                                                                        IF CTR=0 THEN CTR:=01
5303    0530                        JSB  MD03       UNC         UBA  JSB  *-5       NZRO  JSB FOR STACK DEC; JSB IF REMAINING COUNT
5305            *
5306            ********************************************************************
5307            *    The following code emptys the TOS registers to 3 in the event  *
5308            *    the starting or ending address is greater than SM.  It then    *
5309            *    bounds check the address against the new value of SM.          *
5310            ********************************************************************
5311            *
5312    0531    MWE3      DL   ADD        SPOA NFSS  Z         ADD       SP1B            SPOA:=DL, SKIP IF NOT SPLIT BANKS; SP1B:=Z
5314    0532                ADD                  RSB   FFFC      ADDL                    RSB IF SPLIT BANKS; UBB:=-4
5316    0533              SM   INC              WRS   SR   UBB  REPC      RG   DCSR NCRY  WRITE TOS REGISTERS AT SM + 1;
5318                                                                                        RG:=(SR - 4), REPEAT IF NOT (SR > 4), DCSR
5319                                                                                          RG HAS BEEN READ IN RANK1 ALREADY IF SR=7
5320    0534              QDWN ADD              DATA  UBB        CAD             DCSR NEG  WRITE QDWN;
5322                                                                                        END REPEAT WHEN (SR - 4) COUNTS DOWN TO < 0
5323    0535        SPOA  SP2B ADD                    RG   SM   INC       SM    INSR      UBA:=DL + (ADDR - DL);
5325                                                                                        SM:=(SR - 4) + SM + 1 (SM:=S - 3), INSR
5326    0536        UBB   UBA  BNDE                   UBB  SP1B JSZC STO2             POS   BOUNDS CHECK SM >= ADDR, NO RSB HERE (2527)
5328                                                                                        JSB STOV IF SM > Z AND WITHIN 32K
5329    0537                   ADD              RSB                 ADD                     THIS LINE ADDED TO FIX BNDE-RSB BUG (2527)
5331    0538    MD03      CIR  ADD              EVEN                ADD                     CIR(15) INDICATES STACK DECREMENT OF 1
5333    0539                   ADD              EPOP        UBA     ADD  LSR            EVEN  EPOP; SKIP IF NOT STACK DECREMENT OF 2
5335    053A                   JSZ  NEXT       UNC                 ADD            EPP2     JSZ FOR NEXT; EPP2
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5338            ***************************************************************
5339            *    MBLA is the entry point for the Move DB to DL (MVBL)       *
5340            *    instruction, the Move Absolute (MABS) instruction, and the *
5341            *    two-word clock and hardware dependent instructions.  The   *
5342            *    SR preadjust is set to 0 because of the clock instructions.*
5343            ***************************************************************
5344            *
5345            $LUT INSTR=MVBL/MABS:0 010 000 001 00x xxx, DSPL=4, DB, ENTRY=MBLA
5346            *
5347     053B   MBLA FFF8 DSPL ADDL                          BNKD JSZ  TRP6 BKX3       NPRV  UBA.(0:1):=0 IF MABS, UBA:=CIR.(13:3);
5349                                                                                          BKX3:=BNKD FOR MVBL, TRAP IF NOT PRIVILEGED
5350     053C          UBA   JSB  MABS       POS   0004 UBA  ANDL                      ZERO  JSB IF MABS, SKIP IF NOT CIR.(13:1)
5352     053D          UBA   INC  LSL                      UBA  JSL  CLKI BKX3       UNC   UBA:=2; BKX3:=0, JSB if 2-word instruction
5354     053E   MVBL SR      UBA  JSZC ETO3       CRRY                JSZ  PULM       SRZ   EMPTY TOS TO 3 IF SR > 2 (#ETO3 CLEAR F1);
5356                                                                                          PULL ONE TOS IF SR = 0
5357     053F          RC    JSZ  PULM       NF2                     JSZ  PUL2       SRL2  UBA:=(S-2), PULL ANOTHER TOS IF SR <= 2,
5359                                                                                            (F2 SET IF #ETO3 WAS EXECUTED)
5360                                                                                          PULL TWO TOS IF SR < 2
5361     0540       UBA   DL   ADD        SP4A WRS      .     RA   JSB  MBLR XRO        NEG   SP4A:=UBA:=(S-2) + DL (ABSOULTE TARGET ADDR)
5363                                                                                            WRITE USING BNKS
5364                                                                                          XRO:=WORD COUNT, JSB IF REVERSE MOVE
5365     0541          UBA   ADD            FSS   RB   DB   ADD        SP3B ROA3         UBA:=TARGET ADDR, SKIP IF SPLIT BANKS
5367                                                                                          SP3B:=UBB:=ABSOLUTE SOURCE ADDR, READ INTO A
5368     0542       UBA   UBB  JSBC MWBF       ZERO        RA   JSL  D03G       ZERO  JSB IF TARGET AND SOURCE ADDRS OVERLAP BY 1
5370                                                                                            AND NOT SPLIT BANKS;
5371                                                                                          JSB FOR STACK DECREMENT IF WORD COUNT = 0
5372     0543                JSBI MWIX       TEST  SP3B        INC        SP3B ROA3         UBA:=1, JSB IF INTERRUPT;
5374                                                                                          READ NEXT SOURCE WORD, SP3B:=SP3B + 1
5375     0544          OPA   JSB  *-1        DATA  RA   UBA  JSBS MWX  RA         ZERO  WRITE TARGET WORD, LOOP BACK;
5377                                                                                          WORD COUNT:=WORD COUNT - 1, JSB IF = 0
5378     0545   MBLR        SP4A ADD            FSS,  RB   DB   ADD        SP3B ROA3         UBA:=TARGET ADDR, SKIP IF SPLIT BANKS;
5380                                                                                          SP3B:=UBB:=ABSOLUTE SOURCE ADDR, READ INTO A
5381     0546       UBB   UBA  JSBC MWBR       ZERO        RA   JSL  D03G       ZERO  JSB IF SOURCE AND TARGET ADDRS OVERLAP BY 1
5383                                                                                            AND NOT SPLIT BANKS;
5384                                                                                          JSB FOR STACK DECREMENT IF WORD COUNT = 0
5385     0547                JSBI MWIX       TEST  SP3B        CAD        SP3B ROA3         UBA:=1, JSB IF INTERRUPT;
5387                                                                                          READ NEXT SOURCE WORD, SP3B:=SP3B - 1
5388     0548          OPA   JSB  *-1        DATD  RA   UBA  JSB  MWX  RA         ZERO  WRITE TARGET WORD, LOOP BACK;
5390                                                                                          WORD COUNT:=WORD COUNT + 1, JSB IF = 0
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5392          ********************************************************************
5393          *    Move Absolute (MABS) Instruction                              *
5394          *                                                                  *
5395          *    Entered from #MBLA with privileged mode already checked.      *
5396          ********************************************************************
5397          *
5398    0549  MABS       SREG JSZ  PULM      SRZ   0005      ADDL      SP1B          UBA := 2, pull one TOS if SR = 0; SP1B := 5
5400    054A        SR   UBA  JSZC PULM      NCRY              JSZ  PUL2        SRL2  Pull one if SR = 2; Pull two if SR = 1
5402    054B        SR   SP1B JSZS PULM      NCRY  SR   SP1B JSZC PSHM        CRRY  Pull one if SR = 3 or 4; Push one if SR > 5
5404    054C        SR   SP1B JSZS PULM      NCRY  SR   SP1B JSZC PSHM        CRRY  Pull one if SR = 4; Push one if SR = 6
5406    054D             RD   ADD            WRX4        RA   JSB  D07S XRO   ZERO  SEND ADDR USING BKX4;
5408                                                                               XRO:=WORD COUNT, JSB IF ZERO
5409    054E             RA   JSB  MAR       NEG   00FF  RE   ANDL      BKX4        JSB IF WORD COUNT < 0; BKX4:=BANK ADDR
5411    054F  MAF  RE    RC   SUB            NZRO  00FF  RC   ANDL      BKX3        SKIP IF BANK ADDRESSES NOT THE SAME;
5413                                                                               BKX3:=SOURCE BANK
5414    0550       RD    RB   JSBC MABF      ZERO        RB   ADD           ROA3   JSB IF TARGET ADDR ONE GREATER THAN SOURCE
5416                                                                                 AND BANK ADDRESSES ARE THE SAME;
5417                                                                               READ FIRST WORD INTO OPA
5418    0551                  JSBI AIXF      TEST  RB         INC       RB   ROA3   UBA:=1, JSB IF INTERRUPT;
5420                                                                               READ NEXT SOURCE WORD, RB:=RB + 1
5421    0552             OPA  JSB  *-1       DATA  RA   UBA  JSBS MAX  RA    ZERO  WRITE TARGET WORD, LOOP BACK;
5423                                                                               WORD COUNT:=WORD COUNT - 1, JSB IF = 0
5424    0553  MAR  RE    RC   SUB            NZRO  00FF  RC   ANDL      BKX3        SKIP IF BANK ADDRESSES NOT THE SAME;
5426                                                                               BKX3:=SOURCE BANK
5427    0554       RB    RD   JSBC MABR      ZERO        RB   ADD           ROA3   JSB IF TARGET ADDR ONE LESS THAN SOURCE
5429                                                                                 AND BANK ADDRESSES ARE THE SAME
5430                                                                               READ FIRST WORD INTO OPA
5431    0555                  JSBI AIXR      TEST  RB         CAD       RB   ROA3   UBA:=1, JSB IF INTERRUPT;
5433                                                                               READ NEXT SOURCE WORD, ADDR:=ADDR - 1
5434    0556             OPA  JSB  *-1       DATD  RA   UBA  JSB  MAX  RA    ZERO  WRITE TARGET WORD, LOOP BACK;
5436                                                                               WORD COUNT:=WORD COUNT + 1, JSB IF = 0
5437    0557  AIXR RB        INC       RB          RA   XRO  JSBS AIX1        UNC   SOURCE ADDR:=SOURCE ADDR + 1;
5439                                                                               UBB:=WORD COUNT - ORIGINAL WORD COUNT, JSB
5440    0558  MAX              JSB  D07S      UNC   RD   XRO  ADD       RD   SF5B  JSB FOR STACK DECREMENT;
5442                                                                               (S-3):=(S-3) + WORD COUNT, SET F5B TO
5443                                                                                 INHIBIT ADJUSTING (S-1)
5444    0559  AIXF RB        CAD       RB          RA   XRO  SUB                 SOURCE ADDR:=SOURCE ADDR - 1;
5446                                                                               UBB:=WORD COUNT - ORIGINAL WORD COUNT
5447    055A  AIX1           JSZ  IRDN      UNC   RD   UBB  SUB       RD        JSB FOR INTERRUPT;
5449                                                                               TARGET ADDR:=TARGET ADDR + # WORDS MOVED
```

```
RECORD   C S.                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5451           ***************************************************************
5452           *    Move Absolute Blanking Forward -  This routine is used by      *
5453           *    the MABS, MDS, MTDS and MFDS instructions for moves in         *
5454           *    the direction of ascending addresses where the target          *
5455           *    address is one greater than the source.  Word pointer RC       *
5456           *    is adjusted for the MTDS instruction instead of RD.            *
5457           ***************************************************************
5458           *
5459    055B   MABF           ADD                          RA   ADD       CTR  SF5B       ; CTR:=WORD COUNT.(8:8),
5461                                                                                       SET F5B TO INHIBIT ADJUSTING (S-1)
5462    055C                  JSZ  IRDN      TEST               REPC      SP1B DCTR CTR0  JSB IF INTERRUPT;
5464                                                                                       SP1B:=0, REPC IF CTR <> 0, DCTR
5465    055D           OPA    ADD            DATA  UBB          INC       SP1B DCTR CTR0  WRITE SOURCE WORD;
5467                                                                                       SP1B:=UBB + 1, DCTR, END REPEAT IF CTR = 0
5468    055E     RB    SP1B ADD       RB     NF3A  RA   SP1B SUB  RRZ  CTR               RB:=RB + NUMBER OF WORDS MOVED;
5470                                                                                       UBB:=CTR:=REMAINING WORD COUNT.(8:8)
5471    055F     RC    SP1B ADD       RC     UNC         UBB  ADD                  NZRO  RC:=RC + NUMBER OF WORDS MOVED IF MTDS;
5473                                                                                       SKIP IF CTR <> 0
5474    0560     RD    SP1B ADD       RD                        ADD            DCTR       RD:=RD + NUMBER OF WORDS MOVED IF NOT MTDS;
5476                                                                                       IF CTR=0 THEN CTR:=FF
5477    0561                  JSB  D07S      UNC   RA   SP1B JSBS *-5  RA         NZRO  JSB FOR STACK DEC;
5479                                                                                       RA:=RA - NUMBER WORDS MOVED, JSB IF <> 0
5480           *
5481           ***************************************************************
5482           *    Move Absolute Blanking Reverse -  This routine is used by      *
5483           *    the MABS and MDS instructions for moves in the direction       *
5484           *    of descending addresses where the target address is one        *
5485           *    less than the source.                                          *
5486           ***************************************************************
5487           *
5488    0562   MABR           ADD                          RA   ADD       CTR  SF5B       ; CTR:=WORD COUNT.(8:8),
5490                                                                                       SET F5B TO INHIBIT ADJUSTING (S-1)
5491    0563                  JSZ  IRDN      TEST               REPC      SP1B ICTR CTR0  JSB IF INTERRUPT;
5493                                                                                       SP1B:=0, REPC IF CTR <> 0, ICTR
5494    0564           OPA    ADD            DATD  UBB          CAD       SP1B ICTR CTR0  WRITE SOURCE WORD;
5496                                                                                       SP1B:=UBB - 1, ICTR, END REPEAT IF CTR = 0
5497    0565     RB    SP1B ADD       RB           RA   SP1B SUB  RRZ  CTR               RB:=RB - NUMBER OF WORDS MOVED;
5499                                                                                       UBB:=CTR:=REMAINING WORD COUNT.(8:8)
5500    0566     RD    SP1B ADD       RD                 UBB  ADD                  NZRO  RD:=RD - NUMBER OF WORDS MOVED;
5502                                                                                       SKIP IF CTR <> 0
5503    0567     RA    SP1B SUB       RA                        SUB            ICTR       RA:=RA + NUMBER OF WORDS MOVED;
5505                                                                                       IF CTR=0 THEN CTR:=01
5506    0568                  JSB  D07S      UNC         UBA  JSB  *-5  RA         NZRO  JSB FOR STACK DEC; JSB IF REMAINING COUNT
```

```
RECORD   C S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5509           ********************************************************************
5510           *    MLBS is the entry point for both the Move DL to DB (MVLB)     *
5511           *    instruction and the Move Data Segments (MDS) instruction.     *
5512           *    The SR preadjust is set to 3 for MVLB                         *
5513           ********************************************************************
5514           *
5515           $LUT INSTR=MVLB/MDS:0 010 000 001 10x xxx, SR=3, ENTRY=MLBS
5516           *
5517     0569  MLBS 0008 CIR  ANDL                          JSZ  TRP6 BKX3       NPRV  UBA := CIR.(12:1); BKX3 := 0, trap if NPRIV
5519     056A       0005      ADDL      SP4A        UBA   JSB  MVLB              ZERO  SP4A := 5; JSB if MVLB instruction
5521     056B  MDS  SR   UBA  JSZS PULM      NCRY  SR    UBA  JSZC PSHM         CRRY  Pull 1 if NOT (SR >= 5); Push 1 if SR > 5
5523     056C       SR   SP4A JSZS PULM      NCRY              INC  LSL  SP1B ROX3      Pull 1 if NOT (SR >= 5); SP1B := 2, read DST
5525                                                                                     pointer @ abs (2)
5526     056D       SR   SP4A JSZC PSHM      CRRY  RC    RC   ADD  LSL  RG             Push 1 if SR > 5; RG := src DS# * 4
5528     056E       RE   RE   ADD  LSL               OPB  ADD       RF   ROA3         UBA := tar DS# * 4; RF := adr of DST, read
5530     056F       UBA  UBB  ADD            ROX3        RE   JSZ  DSTV RH       ZERO  Read 1st word of tar DST entry; RH := tar
5532                                                                                     DS#, DST violation if it's 0
5533     0570       RE   OPA  JSZC DSTV      CRRY  UBA   SP1B ADD              ROX3  DST violation if tar DS# > DST length;
5535                                                                                     Read 3rd word of tar DST entry
5538     0571            OPA  JSB  ADSE      NEG         UBB  INC              ROX3  Trap if tar DS absent; Read 4th word of tar
5538                                                                                     DST entry
5539     0572       2000      ADDL      SP4A        OOFF OPB  ANDL       BKX4       SP4A := UBA := mask for referenced bit;
5541                                                                                     BKX4 := tar DS bank # (8 bits)
5542     0573       UBA  OPA  IOR             DATA        UBB  ADD        SP2B       Set ref'd bit for tar; SP2B := tar DS bank #
5544     0574            RF   ADD            ROX3  RD    OPB  ADD        RF         Read DST length; RF := tar addr
5546     0575       RF   RG   ADD            ROX3        RC   JSZ  DSTV SP3B  ZERO  Read 1st word of src DST entry; SP3B := src
5548                                                                                     DS#, DST violation if it's 0
5549     0576       RC   OPA  JSZC DSTV      CRRY  UBA   SP1B ADD              ROX3  DST violation if src DS# > DST length;
5551                                                                                     Read 3rd word of src DST entry
5552     0577            OPA  JSB  ADSC      NEG         UBB  INC              ROX3  Trap if src DS absent; Read 4th word of src
5554                                                                                     DST entry
5555     0578       UBA  SP4A IOR             DATA  RA    OPB  XFRR       XRO  HBF2  Set ref'd bit for src; SREGB := src bank #
5557                                                                                     XRO := word count, set F2 if reversed move
5558     0579            RF   JSB  FDS1      WRX4  OOFF  SREG ANDL       BKX3       Set up write to tar adr, JSB;
5560                                                                                     UBB := BKX3 := src bank # (8 bits)
5561     057A  ADSE      RE   ADD       SPOA                      JSZ  ABDS       UNC  SPOA := param (DS#); Trap for absent tar DS
5563     057B  ADSC      RC   ADD       SPOA                      JSZ  ABDS       UNC  SPOA := param (DS#); Trap for absent src DS
```

```
RECORD  C S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5566            ********************************************************************
5567            *    Move DL to DB (MVLB) Instruction                              *
5568            *                                                                  *
5569            *    Entered from #MLBS with privileged mode already checked.      *
5570            ********************************************************************
5571            *
5572    057C    MVLB           JSZ  ETO3      UNC         BNKS JSZ  TRP6 BKX3      NPRV  JSB TO EMPTY TOS TO 3;
5574                                                                                      JSB IF NON-PRIVILEDGED, BKX3:=SOURCE BANK
5575    057D        RC   DB    ADD       SP4A WRD         RA   JSB  MLBR XRO       NEG   SP4A:=UBA:=(S-2) + DB (ABSOLUTE TARGET ADDR)
5577                                                                                        WRITE USING BNKD;
5578                                                                                      XRO:=WORD COUNT, JSB IF REVERSE MOVE
5579    057E            UBA  ADD            FSS   RB   DL   ADD       SP3B ROA3         UBA:=TARGET ADDR, SKIP IF SPLIT BANKS;
5581                                                                                      SP3B:=UBB:=ABSOLUTE SOURCE ADDR, READ INTO A
5582                                                                                        USING BNKS
5583    057F        UBA  UBB   JSBC MWBF       ZERO        RA   JSL  D03G            ZERO  JSB IF TARGET AND SOURCE ADDRS OVERLAP BY 1
5585                                                                                        AND NOT SPLIT BANKS;
5586                                                                                      JSB FOR STACK DECREMENT IF WORD COUNT = 0
5587    0580            JSBI MWIX      TEST  SP3B       INC       SP3B ROA3         UBA:=1, JSB IF INTERRUPT;
5589                                                                                      READ NEXT SOURCE WORD, SP3B:=SP3B + 1
5590    0581            OPA  JSB  *-1  DATA  RA   UBA  JSBS MWX  RA           ZERO  WRITE TARGET WORD, LOOP BACK;
5592                                                                                      WORD COUNT:=WORD COUNT - 1, JSB IF = 0
5593    0582  MLBR      SP4A ADD       FSS   RB   DL   ADD       SP3B ROA3         UBA:=TARGET ADDR, SKIP IF SPLIT BANKS;
5595                                                                                      SP3B:=UBB:=ABSOLUTE SOURCE ADDR, READ INTO A
5596                                                                                        USING BNKS
5597    0583        UBB  UBA   JSBC MWBR       ZERO        RA   JSL  D03G            ZERO  JSB IF SOURCE AND TARGET ADDRS OVERLAP BY 1
5599                                                                                        AND NOT SPLIT BANKS;
5600                                                                                      JSB FOR STACK DECREMENT IF WORD COUNT = 0
5601    0584            JSBI MWIX      TEST  SP3B       CAD       SP3B ROA3         UBA:=1, JSB IF INTERRUPT;
5603                                                                                      READ NEXT SOURCE WORD, SP3B:=SP3B - 1
5604    0585            OPA  JSB  *-1  DATD  RA   UBA  JSB  MWX  RA           ZERO  WRITE TARGET WORD, LOOP BACK;
5606                                                                                      WORD COUNT:=WORD COUNT + 1, JSB IF = 0
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5608            ****************************************************************
5609            *     MFSU is the entry point for the Move from Data Segment (MFDS)  *
5610            *     and Scan Until (SCU) instructions   The SR preadjust is set    *
5611            *     to 2                                                           *
5612            ****************************************************************
5613            *
5614            $LUT INSTR=MFDS/SCU:0 010 000 001 11,SR=2,ENTRY=MFSU
5615            *
5616  0586  MFSU 0008 CIR  ANDL                          ADD       BKX3          ZERO  UBA:=CIR(12:1); BKX3:=0
5618  0587       0004      ADDL      SP4A                 UBA  JSL  SCU           ZERO  SP4A:=4; JSB IF SCU INSTRUCTION
5620  0588  MFDS SR   UBA  JSZC PSHM      CRRY           JSZ  TRP6                NPRV  Trap if not privileged; Else push if SR > 4
5622  0589       SR   SP4A JSZS PULM      NCRY  SR   SP4A JSZC PSHM              CRRY  PULL IF NOT (SR >= 4); PUSH IF SR > 4
5624  058A       SR   SP4A JSZS PULM      NCRY               INC  LSL  SP1B ROX3       PULL IF NOT (SR >= 4);
5626                                                                                   READ DST POINTER AT LOCATION 2, SP1B:=2
5627  058B       SR   SP4A JSZC PSHM      CRRY           RC   JSZ  DSTV RH        ZERO  PUSH IF SR > 4; DSTV IF DS# = 0, RH:=DS#
5629  058C       RC   RC   ADD  LSL                      OPB  ADD                 ROA3  UBA:= DS# * 4; READ FIRST WORD OF DST
5631  058D       UBA  UBB  ADD             ROX3  2000        ADDL      RG             READ FIRST WORD OF DST ENTRY; RG:=BIT MASK
5633  058E       RC   OPA  JSZC DSTV      CRRY  UBA  SP1B ADD           SP1B ROX3       DSTV IF DS# > DS LENGTH;
5635                                                                                   READ THIRD WORD OF DST ENTRY, SP1B:=ADDR
5636  058F       RG   OPA  IOR             DATA        BNKD ADD           SP2B         SET REFERENCED BIT; SP2B:=BNKD
5638  0590            OPA  JSB  ADSC      NEG         SP1B INC                 ROX3    ABDS IF ABSCENCE BIT; READ FOURTH ENTRY WORD
5640  0591       RD   DB   ADD       RF   WRD   00FF OPB  ANDL          BKX3          RF:=TARGET ADDR, WRITE USING BNKD;
5642                                                                                   BKX3:=SOURCE BANK
5643  0592  FDS1 UBB  SP2B SUB             ZERO  RB   OPB  ADD           SP3B ROA3     Do the blanking test if bank #s are same;
5645                                                                                     SP3B := src adr, read 1st word in OPA
5646  0593                  ADD             UNC         UBB  JSB  MDSR          F2    Skip blanking test if bank #'s are not same;
5648                                                                                     UBB := src adr, JSB if reversed MDS
5649  0594       RF   UBB  JSBC MABF      ZERO          RA   JSB  D07S XR0       ZERO  JSB for overlapped move if tar adr 1 greater
5651                                                                                     than src adr; XRB0 := word count, JSB for
5652                                                                                     stack dec if = 0
5653  0595                  JSBI DSIX      TEST  SP3B        INC           SP3B ROA3     UBA:=1, JSB IF INTERRUPT;
5655                                                                                   READ NEXT SOURCE WORD, SP3B:=SP3B + 1
5656  0596            OPA   JSB  *-1       DATA  RA   UBA  JSBS FDSX RA          ZERO  WRITE TARGET WORD, LOOP BACK;
5658                                                                                   WORD COUNT:=WORD COUNT - 1, JSB IF = 0
5659  0597  MDSR              JSBI DSIX      TEST  SP3B        CAD           SP3B ROA3     UBA:=1, JSB IF INTERRUPT
5661                                                                                   READ NEXT SOURCE WORD,DEC SOURCE POINTER
5662  0598            OPA   JSB  *-1       DATD  RA   UBA  JSB  FDSX RA          ZERO  WRITE TARGET WORD,LOOP BACK ;
5664                                                                                   WORD COUNT=WORD COUNT +1, EXIT IF=0
5665  0599  FDSX 0007 CIR   ANDL                        RD   XR0  JSB  D071 RD        UNC   UBA:=CIR.(13:3) = STACK DECREMENT;
5667                                                                                   RD:=RD + WORD COUNT
5668  059A  DSIX             ADD                       RA   XR0  SUB                 ;  UBB:=WORD COUNT - ORIGINAL WORD COUNT
5670  059B                   ADD                       RB   UBB  SUB          RB     ;  RB:=RB + NUMBER OF WORDS MOVED
5672  059C                   JSZ   IRDN      UNC   RD   SREG SUB          RD        JSB TO INTERRUPT HANDLER;
5674                                                                                   RD:=RD + NUMBER OF WORDS MOVED
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5676            ********************************************************************
5677            *    MTSW is the entry point for the Move to Data Segment (MTDS)     *
5678            *    and Scan While (SCW) instructions.  The SR preadjust is set     *
5679            *    to 2.                                                           *
5680            ********************************************************************
5681            *
5682            $LUT INSTR=MTDS/SCW:0 010 000 001 01,SR=2,ENTRY=MTSW
5683            *
5684    059D  MTSW 0008 CIR   ANDL                          ADD       BKX3          UBA:=CIR(12:1); BKX3:=0
5686    059E       0004       ADDL       SP4A              UBA   JSL   SCW              ZERO  SP4A:=4; JSB IF SCW INSTRUCTION
5688    059F  MTDS SR    UBA  JSZC PSHM       CRRY              JSZ   TRP6             NPRV  Trap if not privileged; Else push if SR > 4
5690    05A0       SR    SP4A JSZS PULM       NCRY  SR    SP4A JSZC PSHM             CRRY  PULL IF NOT (SR >= 4); PUSH IF SR > 4
5692    05A1       SR    SP4A JSZS PULM       NCRY              INC   LSL   SP1B ROX3      PULL IF NOT (SR >= 4);
                                                                                           READ DST POINTER AT LOCATION 2, SP1B:=2
5694    05A2       SR    SP4A JSZC PSHM       CRRY  2000        ADDL      RG
5697    05A3       RD    RD   ADD  LSL        SF3A        OPB   ADD             ROA3     PUSH IF SR > 4; RG:=BIT MASK
                                                                                           UBA:= DS# * 4, SF3A TO INDICATE MFDS;
5699                                                                                       READ FIRST WORD OF DST
5700    05A4       UBA   UBB  ADD        SPOA ROX3        RD    JSZ   DSTV RH          ZERO  SPOA:=ADDR OF FIRST WORD OF DST ENTRY, READ;
5702                                                                                       RH:=DS#, DSTV IF DS# = 0
5703    05A5       RD    OPA  JSZC DSTV       CRRY  UBA   SP1B  ADD             ROX3     DSTV IF DS# > DS LENGTH;
5705                                                                                       READ THIRD WORD OF DST ENTRY
5706    05A6             OPA  JSB  ADSD       NEG         UBB   INC             ROA3     ABDS IF ABSCENCE BIT; READ FOURTH ENTRY WORD
5708    05A7       RG    OPA  IOR             DATA        BNKD  ADD       SP2B           SET REFERENCED BIT IN DST ENTRY; SP2B:=BNKD
5710    05A8       RC    OPA  ADD        RF   WRX3  00FF  OPB   ANDL      BKX3           RF:=TARGET ADDR, WRITE USING BKX3;
5712                                                                                       BKX3:=TARGET ADDR
5713    05A9       UBB   SP2B XOR              NZRO  RB    DB    ADD       SP3B ROAD      SKIP IF BANK ADDRESSES NOT THE SAME;
5715                                                                                       SP3B:=SOURCE ADDR, READ FIRST WORD INTO OPA
5716    05AA       RF    UBB  JSBC MABF       ZERO        RA    JSB   D07S  XR0      ZERO  JSB FOR OVERLAPED MOVE IF TARGET ADDR ONE
5718                                                                                         GREATER THAN SOURCE;
5719                                                                                       XR0:=WORD COUNT, JSB FOR STACK DEC IF = 0
5720    05AB                   JSBI MWIX       TEST  SP3B        INC       SP3B ROAD      UBA:=1, JSB IF INTERRUPT;
5722                                                                                       READ NEXT SOURCE WORD, SP3B:=SP3B + 1
5723    05AC             OPA   JSB  *-1       DATA  RA    UBA   JSBS TDSX  RA         ZERO  WRITE TARGET WORD, LOOP BACK;
5725                                                                                       WORD COUNT:=WORD COUNT - 1, JSB IF = 0
5726    05AD  TDSX 0007 CIR   ANDL                       RC    XR0   JSB   D071  RC         UNC   UBA:=CIR.(13:3) = STACK DECREMENT;
5728                                                                                       RC:=RC + WORD COUNT
5729    05AE  ADSD       RD   ADD        SPOA                      JSZ   ABDS            UNC   SPOA:=PARAMETER:=DS#; JSB FOR ABSENT DS
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5732           ***************************************************************
5733           *     Stack Decrement Routines for Moves                        *
5734           *                                                               *
5735           *     These routines decrement the stack by from 0 to 7 words.  *
5736           ***************************************************************
5737           *
5738     05AF  D07S 0007 CIR  ANDL                          ADD                F5B   UBA:=CIR (13:3) = STACK DECREMENT;
5740                                                                                 SKIP ADJUSTING (S-1) IF F5B
5741     05B0  D071 SR   UBA  JSBS D07L      NCRY  RB   XRO ADD       RB              JSB IF NOT (SR >= STACK DECREMENT);
5743                                                                                 RB :=RB + WORD COUNT
5744     05B1  D07G      CIR  CSR       SP4A POS         UBA ADD                    SKIP IF NOT DEC BY ONE; UBB:=SR - STACK DEC
5746     05B2            UBB  ADD            EPOP        UBA ADD  LSR          EVEN  UBA:=SR - STACK DEC, EPOP IF DEC BY ONE;
5748                                                                                 SKIP IF NOT STACK DEC BY FOUR
5749     05B3            SP4A ADD            ODD             ADD            EPP4      SKIP IF DEC BY TWO; EPP4 IF DEC BY FOUR
5751     05B4                 ADD            NEXT            ADD                      NEXT IF NOT STACK DEC BY TWO;
5753     05B5                 JSZ  NEXT      UNC             ADD            EPP2      JSB FOR NEXT IN ONE CLOCK; EPP2
5755     05B6  D07L UBA  SM   ADD       SP4A                ADD                      UBA=(SR + SM) - STACK DECREMENT;
5757     05B7       UBA  Q    JSZS STUN      NCRY            ADD                      JSB FOR STUN IF NOT (NEW SM >= Q);
5759     05B8                 JSZ  NEXT      UNC         SP4A ADD       SM   CLSR    JSB FOR NEXT IN ONE CLOCK;
5761                                                                                 SM:=S - STACK DECREMENT, CLSR
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5763           *********************************************************************
5764           *    BCY instruction, (Branch on carry)  -   BCY is the entry       *
5765           *    point for the direct case, and BCYI is the entry point for     *
5766           *    the indirect case.                                             *
5767           *********************************************************************
5768           *
5769           $LUT INSTR=BCY:0 001 001 100 0X,DSPL=5,P,ENTRY=BCY
5770           $LUT INSTR=BCY:0 001 101 100 0X,DSPL=5,P,INDR,ENTRY=BCYI
5771           $LUT INSTR=BCY:0 001 001 100 1X,DSPL=5,P,F2,ENTRY=BCY
5772           $LUT INSTR=BCY:0 001 101 100 1X,DSPL=5,P,F2,INDR,ENTRY=BCYI
5773           *
5774    05B9   BCY  0400 STA  ANDL      SPOA            P    ADD       RH           SPOA:=UBA:=CARRY BIT; RH:=P BEFORE INCREMENT
5776    05BA   BCY1      UBA  ADD           ZERO        UBA  ADD                NZRO  SKIP IF CARRY; SKIP IF NO CARRY
5778    05BB        RH   DSPL ADSB          RONP            ADD                NEXT  UBA:=TARGET P, READ INTO NIR; NEXT
5780    05BC        UBA  PB   UBNE                  PL   UBA  UBNE                 BOUNDS CHECK TARGET P
5782    05BD        SPOA STA  XOR       STA             SREG JSZ  NEXT P     UNC   CLEAR CARRY OR OVERFLOW IF SET;
5784                                                                               P:=TARGET P AFTER BOUNDS CHECKING, JSB
5785    05BE   BCYI 0400 STA  ANDL      SPOA            P    ADD                   SPOA:=UBA:=CARRY BIT, UBB:=P
5787    05BF   BCI1 UBB  DSPL ADSB      RH   ROP        UBA  ADD                NZRO  RH:=UBA:=INDIRECT CELL ADDR, READ;
5789                                                                               SKIP NEXT IF CARRY IS SET
5790    05C0        UBA  PB   SUB           CTF1            ADD                NEXT  F1:=0 IF BNDV; NEXT IF BRANCH NOT TAKEN
5792    05C1        RH   OPA  ADD           RONP  PL   RH   UBNE                   UBA:=TARGET P, READ INTO NIR; BOUNDS CHECK
5794    05C2   BCI2 UBA  PB   UBNE                  PL   UBA  UBNE                 BOUNDS CHECK TARGET P
5796    05C3        SPOA STA  XOR       STA             SREG JSZ  NEXT P     UNC   CLEAR CARRY OR OVERFLOW IF SET;
5798                                                                               P:=TARGET P AFTER BOUNDS CHECKING, JSB
5799           *
5800           *********************************************************************
5801           *    BNCY instruction, (Branch on no carry)  -  BNCY is the entry   *
5802           *    point for the direct case, and BNCI is the entry point for     *
5803           *    the indirect case.                                             *
5804           *********************************************************************
5805           *
5806           $LUT INSTR=BNCY:0 001 001 101 0X,DSPL=5,P,ENTRY=BNCY
5807           $LUT INSTR=BNCY:0 001 101 101 0X,DSPL=5,P,INDR,ENTRY=BNCI
5808           $LUT INSTR=BNCY:0 001 001 101 1X,DSPL=5,P,F2,ENTRY=BNCY
5809           $LUT INSTR=BNCY:0 001 101 101 1X,DSPL=5,P,F2,INDR,ENTRY=BNCI
5810           *
5811    05C4   BNCY 0400 STA  ANDL      SPOA            P    ADD       RH           SPOA:=UBA:=CARRY BIT; RH:=P BEFORE INCREMENT
5813    05C5   BNC1     UBA  ADD           NZRO        UBA  ADD                ZERO  SKIP IF NO CARRY; SKIP IF CARRY
5815    05C6        RH   DSPL ADSB          RONP            ADD                NEXT  UBA:=TARGET P, READ INTO NIR; NEXT
5817    05C7        UBA  PB   UBNE                  PL   UBA  UBNE                 BOUNDS CHECK TARGET P
5819    05C8        SPOA STA  XOR       STA             SREG JSZ  NEXT P     UNC   CLEAR CARRY OR OVERFLOW IF SET;
5821                                                                               P:=TARGET P AFTER BOUNDS CHECKING, JSB
5822                                                                               SPOA:=UBA:=CARRY BIT; UBB:=P
5823    05C9   BNCI 0400 STA  ANDL      SPOA            P    ADD                   SPOA:=UBA:=CARRY BIT;
5825                                                                               UBB:=P BEFORE INCREMENT
5826    05CA   BNI1 UBB  DSPL ADSB      RH   ROP        UBA  ADD                ZERO  RH:=UBA:=INDIRCT CELL ADDR, READ;
5828                                                                               SKIP NEXT IF CARRY IS NOT SET
5829    05CB        UBA  PB   SUB           CTF1            ADD                NEXT  F1:=0 IF BNDV; NEXT IF BRANCH NOT TAKEN
5831    05CC        RH   OPA  JSB  BCI2     RONP  PL   RH   UBNE                   UBA:=TARGET P, READ, JSB; BOUNDS CHECK
```

```
RECORD   C S          ********** ALU A **********  ************* ALU B **************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5834           ************************************************************************
5835           *    BOV instruction, (Branch on overflow)  -  BOV is the entry       *
5836           *    point for the direct case, and BOVI is the entry point for       *
5837           *    the indirect case.                                               *
5838           ************************************************************************
5839           *
5840           $LUT INSTR=BOV:0 001 011 000 0X,DSPL=5,P,ENTRY=BOV
5841           $LUT INSTR=BOV:0 001 111 000 0X,DSPL=5,P,INDR,ENTRY=BOVI
5842           $LUT INSTR=BOV:0 001 011 000 1X,DSPL=5,P,F2,ENTRY=BOV
5843           $LUT INSTR=BOV:0 001 111 000 1X,DSPL=5,P,F2,INDR,ENTRY=BOVI
5844           *
5845     05CD  BOV  0800 STA  ANDL      SPOA            P    JSB  BCY1 RH        UNC   SPOA:=UBA:=OVERFLOW BIT; RH:=P, JSB
5847     05CE  BOVI 0800 STA  ANDL      SPOA            P    JSB  BCI1           UNC   SPOA:=UBA:=OVERFLOW BIT; UBB:=P, JSB
5849           *
5850           ************************************************************************
5851           *    BNOV instruction, (Branch on no overflow)  -  BNOV is the        *
5852           *    entry point for the direct case, and BNOI is the entry point     *
5853           *    for the indirect case.                                           *
5854           ************************************************************************
5855           *
5856           $LUT INSTR=BNOV:0 001 011 001 0X,DSPL=5,P,ENTRY=BNOV
5857           $LUT INSTR=BNOV:0 001 111 001 0X,DSPL=5,P,INDR,ENTRY=BNOI
5858           $LUT INSTR=BNOV:0 001 011 001 1X,DSPL=5,P,F2,ENTRY=BNOV
5859           $LUT INSTR=BNOV:0 001 111 001 1X,DSPL=5,P,F2,INDR,ENTRY=BNOI
5860           *
5861     05CF  BNOV 0800 STA  ANDL      SPOA            P    JSB  BNC1 RH        UNC   SPOA:=UBA:=OVERFLOW BIT; RH:=P, JSB
5863     05D0  BNOI 0800 STA  ANDL      SPOA            P    JSB  BNI1           UNC   SPOA:=UBA:=OVERFLOW BIT; UBB:=P, JSB
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5866            ***************************************************************
5867            *    BRO instruction, (Branch on TOS odd)  -  BRO is the entry      *
5868            *    point for the direct case, and BROI is the entry point for     *
5869            *    the indirect case.                                             *
5870            ***************************************************************
5871            *
5872            $LUT INSTR=BRO:0 001 011 110 0X,SR=1,DSPL=5,P,ENTRY=BRO
5873            $LUT INSTR=BRO:0 001 111 110 0X,SR=1,DSPL=5,P,INDR,ENTRY=BROI
5874            $LUT INSTR=BRO:0 001 011 110 1X,SR=1,DSPL=5,P,F2,ENTRY=BRO
5875            $LUT INSTR=BRO:0 001 111 110 1X,SR=1,DSPL=5,P,F2,INDR,ENTRY=BROI
5876            *
5877    05D1    BRO  UBA  RA   XFRS           EVEN       RA   ADD            EPOP ODD   RREG:=P, SKIP IF (S) EVEN; SKIP IF ODD, EPOP
5879    05D2         RREG DSPL ADSB           RONP            JSZ  NEXT            UNC   UBA:=TARGET P, READ; NEXT IF (S) NOT ODD
5881    05D3         UBA  PB   UBNE                     PL   UBA  UBNE                   BOUNDS CHECK TARGET P
5883    05D4                   ADD                          SREG JSZ  NEXT P         UNC   JSB; P:=TARGET P AFTER BOUNDS CHECKING
5885    05D5    BROI UBA  DSPL ADSB      RH   ROP            RA   ADD            EPOP ODD   RH:=UBA:=TARGET P, READ; (S) ODD?, EPOP
5887    05D6         UBA  PB   SUB            CTF1            JSZ  NEXT            UNC   F1:=0 IF BNDV; NEXT IF (S) NOT ODD
5889    05D7         RH   OPA  ADD            RONP  PL   RH   UBNE                       UBA:=TARGET P, READ INTO NIR; BOUNDS CHECK
5891    05D8         UBA  PB   UBNE           F1    PL   UBA  UBNE                 NF1   BOUNDS CHECK, BOUNDS CHECK, SKIP IF BNDV
5893    05D9                   JSZ  BNDV      UNC            SREG JSZ  NEXT P       UNC   JSB IF BNDV; P:=TARGET P, JSB
5895            *
5896            ***************************************************************
5897            *    BRE instruction, (Branch on TOS even)  -  BRE is the entry     *
5898            *    point for the direct case, and BREI is the entry point for     *
5899            *    the indirect case.                                             *
5900            ***************************************************************
5901            *
5902            $LUT INSTR=BRE:0 001 011 111 0X,SR=1,DSPL=5,P,ENTRY=BRE
5903            $LUT INSTR=BRE:0 001 111 111 0X,SR=1,DSPL=5,P,INDR,ENTRY=BREI
5904            $LUT INSTR=BRE:0 001 011 111 1X,SR=1,DSPL=5,P,F2,ENTRY=BRE
5905            $LUT INSTR=BRE:0 001 111 111 1X,SR=1,DSPL=5,P,F2,INDR,ENTRY=BREI
5906            *
5907    05DA    BRE  UBA  RA   XFRS           ODD        RA   ADD            EPOP EVEN  RREG:=P, SKIP IF (S) ODD; SKIP IF EVEN, EPOP
5909    05DB         RREG DSPL ADSB           RONP            JSZ  NEXT            UNC   UBA:=TARGET P, READ; NEXT IF (S) NOT EVEN
5911    05DC         UBA  PB   UBNE                     PL   UBA  UBNE                   BOUNDS CHECK TARGET P
5913    05DD                   ADD                          SREG JSZ  NEXT P         UNC   JSB; P:=TARGET P AFTER BOUNDS CHECKING
5915    05DE    BREI UBA  DSPL ADSB      RH   ROP            RA   ADD            EPOP EVEN  RH:=UBA:=TARGET P, READ; (S) EVEN?, EPOP
5917    05DF         UBA  PB   SUB            CTF1            JSZ  NEXT            UNC   F1:=0 IF BNDV, JSB; JSB NEXT IF (S) NOT EVEN
5919    05E0         RH   OPA  ADD            RONP  PL   RH   UBNE                       UBA :=TARGET P, READ INTO NIR; BOUNDS CHECK
5921    05E1         UBA  PB   UBNE           F1    PL   UBA  UBNE                 NF1   BOUNDS CHECK, BOUNDS CHECK, SKIP IF BNDV
5923    05E2                   JSZ  BNDV      UNC            SREG JSZ  NEXT P       UNC   JSB IF BNDV; P:=TARGET P, JSB
```

```
 RECORD   C S       ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 5926           ********************************************************************
 5927           *    For the CPRB instruction, (Compare range and branch), the     *
 5928           *    integer to be compared is in X, the upper bound is in (S),    *
 5929           *    and the lower bound is in (S-1).                              *
 5930           ********************************************************************
 5931           *
 5932           $LUT INSTR=CPRB:0 001 010 110 0X,DSPL=5,P,SR=2,ENTRY=CPRB
 5933           $LUT INSTR=CPRB:0 001 110 110 0X,DSPL=5,P,SR=2,INDR,ENTRY=CPRI
 5934           $LUT INSTR=CPRB:0 001 010 110 1X,DSPL=5,P,SR=2,F2,ENTRY=CPRB
 5935           $LUT INSTR=CPRB:0 001 110 110 1X,DSPL=5,P,SR=2,INDR,F2,ENTRY=CPRI
 5936           *
 5937    05E3   CPRB UBA   DSPL ADSB       RH          8000      ADDL      SP1B           RH:=TARGET P; SP1B:=UBB:=32K
 5939    05E4        X     UBB  ADD        SP4A        RREG RA   ADD            EPP2       SP4A:=UBA:=X AS A LOGICAL QUANTITY;
 5941                                                                                      UBB:=(S) AS A LOGICAL QUANTITY, EPP2
 5942    05E5        RB    SP1B ADD                          UBB  UBA  SUB            CCZ  NCRY  UBA:=(S-1) AS A LOGICAL QUANTITY;
 5944                                                                                      NCRY IF NOT (S) >= X, SET CCG
 5945    05E6        UBA   SP4A CAD              NCRY                  CAD            CCA  UNC   NCRY IF (S-1) > X; SET CCL
 5947    05E7                   ADD              NEXT                  ADD                 NEXT  NEXT IF (S-1) > X; NEXT IF X>(S)
 5949    05E8                RH   JSB  BCC1      RONP                  ADD            CCA        READ TARGET P INTO NIR; SET CCE
 5951    05E9   CPRI UBA   DSPL ADSB       RH    ROP   8000      ADDL      SP1B           RH:=INDIRECT CELL, READ; SP1B:=UBB:=32K
 5953    05EA        X     UBB  ADD        SP4A        RREG RA   ADD            EPP2       SP4A:=UBA:=X AS A LOGICAL QUANTITY;
 5955                                                                                      UBB:=(S) AS A LOGICAL QUANTITY, EPP2
 5956    05EB        RB    SP1B ADD                          UBB  UBA  SUB            CCZ  NCRY  UBA:=(S-1) AS A LOGICAL QUANTITY;
 5958                                                                                      NCRY IF NOT (S) >= X, SET CCG
 5959    05EC        UBA   SP4A CAD              NCRY                  CAD            CCA  UNC   NCRY IF (S-1) > X; SET CCL
 5961    05ED                   ADD              NEXT                  ADD                 NEXT  NEXT IF (S-1) > X; NEXT IF X > (S)
 5963    05EE        RH    PB   UBNE                         PL   RH   UBNE                     BOUNDS CHECK INDIRECT CELL ADDRESS
 5965    05EF        RH    OPA  JSB  BCC1      RONP                  ADD            CCA        UBA:=TARGET P, READ INTO NIR; SET CCE
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

5968           ****************************************************************
5969           *    The BR instruction, (Branch unconditionally), has the same     *
5970           *    entry points as the BCC instruction for P relative addressing  *
5971           *    In the direct case the entry point is BCC and in the indirect  *
5972           *    case the entry point is BCCI.  DSPL selects bits 8 through 15  *
5973           *    for BR, (P relative), and bits 11 through 15 for BCC.  F2 is   *
5974           *    set to the value of bit 7 for BR and bit 10 for BCC.  For BR,  *
5975           *    bit 4 indicates whether indexing is used  If the branch        *
5976           *    should not be taken the hardware inhibits memory references    *
5977           *    in ALUA and sets F5B.  The first line of the instruction       *
5978           *    jumps to do a NEXT based on F5B                                *
5979           ****************************************************************
5980           *
5981           $LUT INSTR=BR(P+):1 100 000 0XX XX,DSPL=8,P,ENTRY=BCC
5982           $LUT INSTR=BR(P+):1 100 100 0XX XX,DSPL=8,X,P,ENTRY=BCC
5983           $LUT INSTR=BR(P-):1 100 000 1XX XX,DSPL=8,P,F2,ENTRY=BCC
5984           $LUT INSTR=BR(P-):1 100 100 1XX XX,DSPL=8,X,F2,P,ENTRY=BCC
5985           $LUT INSTR=BR(P+):1 100 010 0XX XX,DSPL=8,INDR,P,ENTRY=BCCI
5986           $LUT INSTR=BR(P+):1 100 110 0XX XX,DSPL=8,X,INDR,P,ENTRY=BCCI
5987           $LUT INSTR=BR(P-):1 100 010 1XX XX,DSPL=8,INDR,F2,P,ENTRY=BCCI
5988           $LUT INSTR=BR(P-):1 100 110 1XX XX,DSPL=8,X,INDR,F2,P,ENTRY=BCCI
5989           $LUT INSTR=BCC(P+):1 100 001 XXX 0X,DSPL=5,P,ENTRY=BCC
5990           $LUT INSTR=BCC(P+):1 100 101 XXX 0X,DSPL=5,P,INDR,ENTRY=BCCI
5991           $LUT INSTR=BCC(P-):1 100 001 XXX 1X,DSPL=5,P,F2,ENTRY=BCC
5992           $LUT INSTR=BCC(P-):1 100 101 XXX 1X,DSPL=5,P,F2,INDR,ENTRY=BCCI
5993           *
5994     05F0  BCC  UBA  DSPL ADSB           RONP           JSZ  NEXT          F5B   UBA:=TARGET P, READ INTO NIR, (INCLUDES XC);
5996                                                                                  JSB IF CONDITION CODE NOT MET
5997     05F1  BCC1 UBA  PB   UBNE                    PL    UBA  UBNE                BOUNDS CHECK TARGET P
5999     05F2                 ADD                           SREG JSZ  NEXT P        UNC    P:=TARGET P AFTER BOUNDS CHECKING
6001     05F3  BCCI UBA  DSPL ADSB      RH   ROP            JSZ  NEXT          F5B   RH:=UBA:=INDIRECT CELL ADDR, READ;
6003                                                                                  JSB IF CONDITION CODE NOT MET
6004     05F4       XC   UBA  ADD                 PL    UBA  UBNE                    UBA:=XC + INDIRECT ADDR; BOUNDS CHECK
6006     05F5       UBA  OPA  ADD           RONP  RH    PB   UBNE                    UBA:=TARGET P, READ INTO NIR; BOUNDS CHECK
6008     05F6       UBA  PB   UBNE                PL    UBA  UBNE                BOUNDS CHECK TARGET P
6010     05F7                 ADD                           SREG JSZ  NEXT P        UNC   JSB; P:=TARGET P AFTER BOUNDS CHECKING
6012           *
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6014            ************************************************************************
6015            *    BRIS is the entry point for the BR, (Branch unconditionally),    *
6016            *    instruction for S- indirect addressing.  BRID is the entry       *
6017            *    point for DB+, Q-, and Q+ indirect addressing.  Bit 4            *
6018            *    indicates whether indexing is used.                              *
6019            ************************************************************************
6020            *
6021            $LUT INSTR=BR(DB+):1 100 011 0XX XX,DSPL=8,DB,INDR,ENTRY=BRID
6022            $LUT INSTR=BR(DB+):1 100 111 0XX XX,DSPL=8,DB,X,INDR,ENTRY=BRID
6023            $LUT INSTR=BR(Q+):1 100 011 10X XX,DSPL=7,Q,INDR,ENTRY=BRID
6024            $LUT INSTR=BR(Q+):1 100 111 10X XX,DSPL=7,Q,X,INDR,ENTRY=BRID
6025            $LUT INSTR=BR(Q-) 1 100 011 110 XX,DSPL=6,Q,INDR,F2,ENTRY=BRID
6026            $LUT INSTR=BR(Q-):1 100 111 110 XX,DSPL=6,Q,X,INDR,F2,ENTRY=BRID
6027            $LUT INSTR=BR(S-):1 100 011 111 XX,DSPL=6,SM,INDR,F2,ENTRY=BRIS
6028            $LUT INSTR=BR(S-):1 100 111 111 XX,DSPL=6,SM,X,INDR,F2,ENTRY=BRIS
6029            *
6030    05F8  BRIS SR   SM   ADD                        ADD                          UBA:=SR + SM;
6032    05F9  BRID UBA  DSPL ADSB      RH   ROD   SR   SM   ADD                      RH:=UBA:=INDIRECT CELL ADDR, READ; UBB:=S
6034    05FA       UBA  SM   CAD                     UBB  UBA  BNDE       CTR  TICB    BOUNDS CHECK ADDR<=S, CTR:=IF ADDR>SM AND
6036                                                                                    S>=ADDR AND NOT (FSS AND DB REL ADDR)
6037                                                                                      THEN (S-ADDR) ELSE CODE FOR OPERAND
6038    05FB       XC   PB   ADD                     RH   DL   BNDE            STFF     UBA:=XC + PB;
6040                                                                                    BOUNDS CHECK ADDR >= DL, STFF
6041    05FC       UBA  REGN ADD            RONP                JSB  BCC1           UNC  UBA:=TARGET P, READ INTO NIR; JSB
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6044             ************************************************************************
6045             *                                                                      *
6046             *          Halt Mode Interrupt Module                                  *
6047             *                                                                      *
6048             *          This module will handle the interrupts occurring            *
6049             *          while halted.  These include:                               *
6050             *                                                                      *
6051             *                     CPX1(4)   -  Run/Halt Switch Interrupt           *
6052             *                     CPX1(8)   -  DCU Command Interrupt               *
6053             *                     CPX1(12)  -  Power fail warning                  *
6054             *                                                                      *
6055             *          Entered with SP4A = CPX1 & csl(1)                           *
6056             *                                                                      *
6057             ************************************************************************
6058             *
6059   05FD HMIT 1000 STA  ANDL      SPOA       0800      ADDL                         SPOA := right stackop bit; UBB := CCPX mask
6061   05FE      0010 SP4A ANDL                      UBB   ADD             CCPX       UBA := PFWINT; Clear R/HSWINT
6063   05FF      0100 SP4A ANDL                      UBA   JSZ  PFW             NZRO  UBA := DCU cmd int; Jump if power fail
6065   0600      1000 SP4A ANDL                      UBA   JSL  DCMD            NZRO  UBA := R/HSWINT; Process DCU interrupt
6067   0601      0600      ADDL                      UBA   JSL  STOP            ZERO  UBA := mask to set RUN FF; Jump if not any
6069                                                                                    of the expected interrupts
6070   0602      SPOA      JSZ  RSRT      UNC       UBA   ADD             CCPX       UBA := right stackop bit, restart instr;
6072                                                                                    Set RUN FF bit in CPX2
6073   0603 BKPT           ADD                     001B      ADDL      CTR            ; Set counter to DCUSTOR with value of 3
6075   0604      00B0      ADDL      REGN                    ADD                RSB   Issue DCUSTOR; Return to #STOP thru #RMIT
```

```
RECORD   C.S          ********** ALU A **********  ************* ALU B ***************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6078           ************************************************************************
6079           *    DSPL for all the Loop Control instructions is 8.  F2 is set      *
6080           *    by the LUT to the value of bit 7 in the instruction.             *
6081           ************************************************************************
6082           *
6083           ************************************************************************
6084           *    Test and Branch, limit in A  (TBA)                                *
6085           ************************************************************************
6086           *
6087           *
6088           $LUT INSTR=TBA:0 101 000 0xx xxx xxx, SR=3, DSPL=8, P, ENTRY=TBA
6089           $LUT INSTR=TBA:0 101 000 1xx xxx xxx, SR=3, DSPL=8, P, ENTRY=TBA, F2
6090           *
6091     0605  TBA  UBA  DSPL ADSB      RH   RONP  SR   SM   ADD       SP3B            RH:=P +/- DSPL (TARGET P); SP3B:=SR + SM
6093     0606       RC   DB   ADD       SPOA ROBD  PL   UBA  UBNE                      SPOA:=VARIABLE ADDR, READ;
6095                                                                                   BOUNDS CHECK PL >= TARGET P
6096     0607       UBA  SM   CAD                   SP3B UBA  BNDE      CTR  TICB      ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
6098                                                                                     S>=ADDR AND NOT (FSS AND DB REL ADDR)
6099                                                                                     THEN (S-ADDR) ELSE CODE FOR OPERAND
6100     0608            RB   ADD             HBF2 8000 RA   ADDL      RH              F2:=SIGN OF INCREMENT;
6102                                                                                   RH:=LIMIT AS A LOGICAL QUANTITY
6103     0609       RH   PB   UBNE      RH   NF2  RREG REGN ADD                 F2    BOUNDS CHECK TARGET P >= PB, RH:=P - PB;
6105                                                                                   UBB:=VARIABLE AS A LOGICAL QUANTITY
6106     060A       UBB  RH   SUB             CTF1 RH   UBB  SUB            CTF1      F1:=IF INCR < 0 THEN (VAR >= LIMIT)
6108                                                                                     ELSE (LIMIT >= VAR)
6109     060B       SPOA DL   BNDE            F1   0003      ADDL                    BOUNDS CHECK VARIABLE ADDR >= DL; UBB:=3
6111     060C            P    JSB  MTA1      RONP UBB  CTRS JSZC BNDV           CRRY  REREAD (P) IF LIMIT EXCEEDED, JSB;
6113                                                                                   BNDV IF S-2 <= VARIABLE ADDR <= S,
6114                                                                                     ((3 > CTRS) IMPLYS ADDR IN PARAMETERS)
6115     060D                 ADD                 RH   PB   JSZ  NEXT P         UNC   ; P:=TARGET P, JSB FOR NEXT
```

```
RECORD  C S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6118            ********************************************************************
6119            *     Modify Variable, Test and Branch, limit in A  (MTBA)         *
6120            ********************************************************************
6121            *
6122            $LUT INSTR=MTBA:0 101 010 0xx xxx xxx, SR=3, DSPL=8, P, ENTRY=MTBA
6123            $LUT INSTR=MTBA:0 101 010 1xx xxx xxx,SR=3, DSPL=8, P, ENTRY=MTBA, F2
6124            *
6125    060E    MTBA UBA  DSPL ADSB      RH   RONP  SR   SM   ADD       SP3B            RH:=P+DSPL (TARGET P), READ; SP3B:=SR+SM
6127    060F         RC   DB   ADD       SPOA ROD         RB   ADD            HBF2       SPOA:=VARIABLE ADDR, READ;
6129                                                                                     F2:=SIGN OF INCREMENT
6130    0610         UBA  SM   CAD                  SP3B UBA  BNDE      CTR  TICB       , BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
6132                                                                                       S>=ADDR AND NOT (FSS AND DB REL ADDR)
6133                                                                                       THEN (S-ADDR) ELSE CODE FOR OPERAND
6134    0611         0003      ADDL                 PL   RH   UBNE                       UBA:=4; BOUNDS CHECK PL >= TARGET P
6136    0612         SPOA DL   BNDE                 UBA  CTRS JSZC BNDV           CRRY  BOUNDS CHECK VARIABLE ADDR >= DL;
6138                                                                                     BNDV IF S-2 <= VARIABLE ADDR <= S,
6139                                                                                       ((3 > CTRS) IMPLYS ADDR IN PARAMETERS)
6140    0613         RB   REGN ADD            DATA 8000 RA   ADDL      RH                 UBA:=INCREMENT + VARIABLE, WRITE;
6142                                                                                     RH:=LIMIT AS A LOGICAL QUANTITY
6143    0614         RH   PB   UBNE      SPOA       RREG UBA  ADD       SP3B       F2    BOUNDS CHECK TARGET P >= PB, SPOA:=P - PB;
6145                                                                                     SP3B:=UBB:=VARIABLE AS A LOGICAL QUANTITY
6146    0615         RB   REGN ADD       REGN NOFL  RH   UBB  SUB            CTF1 UNC   SKIP IF NOFL FROM SUM INCREMENT + VARIABLE;
6148                                                                                     IF INCREMENT>=0 THEN F1:=(LIMIT >= SUM)
6149    0616              ADD            CF1  SP3B RH   SUB            CTF1            CF1 IF OVERFLOW FROM SUMMING;
6151                                                                                     IF INCREMENT<0 THEN F1:=(SUM >= LIMIT)
6152    0617         SPOA PB   ADD            F1             ADD                 NF1   UBA:=TARGET P, SKIP IF LIMIT NOT EXCEEDED;
6154                                                                                     SKIP IF LIMIT EXCEEDED
6155    0618              P    ADD            RONP       UBA  JSZ  NEXT P       UNC   REREAD (P); P:=TARGET P, JSB FOR NEXT
6157    0619    MTA1           JSZ  NOP       EPOP            ADD            EPP2      JSB TO WAIT 2 CLOCKS FOR NEXT, EPOP; EPP2
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6160            ***************************************************************
6161            *    Test and Branch, variable in X  (TBX)                      *
6162            ***************************************************************
6163            *
6164            $LUT INSTR=TBX:0 101 100 0XX XX,SR=2,DSPL=8,P,ENTRY=TBX
6165            $LUT INSTR=TBX:0 101 100 1XX XX,SR=2,DSPL=8,P,F2,ENTRY=TBX
6166            *
6167    061A    TBX  UBA  DSPL ADSB      RH   RONP  8000      ADDL      SP1B           RH:=UBA;=TARGET P, READ; SP1B:=32K
6169    061B         RB   ADD            HBF2 PL    UBA  UBNE                        F2:=SIGN OF INCREMENT;
6171                                                                                  BOUNDS CHECK PL >= TARGET P
6172    061C         X    SP1B ADD            NF2  RA    SP1B ADD                F2    UBA:=VARIABLE AS A LOGICAL QUANTITY;
6174                                                                                  UBB:=LIMIT AS A LOGICAL QUANTITY
6175    061D         UBA  UBB  SUB            CTF1 UBB   UBA  SUB          CTF1       F1:=INCR < 0 THEN (VARIABLE >= LIMIT)
6177                                                                                    ELSE (LIMIT >= LIMIT)
6178    061E         RH   PB   UBNE           F1         P    ADD       SP1B  NF1    BOUNDS CHECK TARGET P >= PB;
6180                                                                                  SP1B:=P, SKIP IF LIMIT EXCEEDED
6181    061F              P    ADD           RONP       RH   JSZ  NEXT P     UNC    REREAD (P) INTO NIR IF LIMIT EXCEEDED;
6183                                                                                  P:=TARGET P, JSB FOR NEXT
6184    0620                   JSZ  NOP      UNC        SP1B ADD       P     EPP2   JSB FOR NEXT; RESTORE OLD P, EPP2
6186            $NOWARN
6187            ***************************************************************
6188            *    Modify variable in X, Test and Branch  (MTBX)              *
6189            ***************************************************************
6190            *
6191            $LUT INSTR=MTBX:0 101 110 0xx xxx xxx, SR=2, DSPL=8, P, ENTRY=MTBX
6192            $LUT INSTR=MTBX:0 101 110 1xx xxx xxx, SR=2, DSPL=8, P, F2,ENTRY=MTBX
6193            *
6194    0621    MTBX UBA  DSPL ADSB      SPOA RONP      RB   ADD            HBF2       SPOA:=TARGET P, READ; F2:=SIGN OF INCREMENT
6196    0622         X    RB   ADD        X    NOFL  PL    UBA  UBNE                  X:=X ; INCREMENT, SKIP IF NO OVERFLOW;
6198                                                                                  BOUNDS CHECK PL >= TARGET P
6199    0623              UBA  JSB  MTX1   CF1  8000  RA    ADDL      RH             JSB IF OVFL, CF1 TO INDICATE LIMIT EXCEEDED;
6201                                                                                  RH:=LIMIT AS A LOGICAL QUANTITY
6202    0624         SPOA PB   UBNE      RH   NF2   RREG  UBA  ADD            F2     BOUNDS CHECK TARGET P >= PB;
6204                                                                                  UBB:=X AS A LOGICAL QUANTITY
6205    0625         UBB  RH   SUB            CTF1  RH    UBB  SUB          CTF1       F1:=IF INCR < 0 THEN (VARIABLE >= LIMIT)
6207                                                                                    ELSE (LIMIT >= VARIABLE)
6208    0626    MTX1                 ADD      NF1   RH    PB   ADD       P     F1     SKIP IF LIMIT NOT EXCEEDED;
6210                                                                                  P:=(TARGET P - PB) + PB,
6211                                                                                  SKIP IF LIMIT EXCEEDED
6212    0627                         ADD      NEXT        P    ADD       P     RONP   NEXT; RESTORE P AND READ IF NOT EXCEEDED
6214    0628                         JSZ  NOP      UNC                   ADD       EPP2   JSB to #NOP to wait 2 clocks; E-pop twice
6216            $WARN
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6218           ********************************************************************
6219           *    SCAN (Scan bits) Instruction                                  *
6220           ********************************************************************
6221           *
6222           $LUT INSTR=SCAN:0 001 000 110 00x xxx, SR=1, DSPL=6, ENTRY=SCAN
6223           $LUT INSTR=SCAN:0 001 100 110 00x xxx, SR=1, DSPL=6, X, ENTRY=SCNX
6224           *
6225     0629  SCAN       RA   JSB  SCN0       ZERO       RA   REPC                 NEG   UBA:=(S), JSB IF ZERO;
6227                                                                                       UBB:=(S), REPC IF NOT (S).(0:1)
6228     062A            UBA  ADD  LSL        NEG        UBB  INC                          SHIFT (S) UNTIL BIT 0 = 1;
6230                                                                                       UBB:=(S) + #SHIFTS
6231     062B       UBB  RA   CAD       X                  UBA  JSB  SCN1            UNC   X:=((S) + #SHIFTS + 1) - (S) - 1;
6233                                                                                       UBB:=SHIFTED (S), JSB
6234     062C  SCN0 0010       ADDL      X                       ADD                CCA  NEXT  X:=16, CCA ON 0, NEXT
6236     062D  SCNX X          ADD       SP4A               RA   JSB  SCX0            ZERO  SP4A:=X; JSB IF (S) = 0
6238     062E                  ADD                          RA   REPC                 NEG   UBA:=0; UBB:=(S), SKIP REPEAT IF BIT(0)
6240     062F            UBA   INC                          UBB  ADD  LSL             NEG   UBA:=NUMBER OF SHIFTS UNTIL 1 SHIFTED OUT;
6242                                                                                       UBB:=SHIFTED (S)
6243     0630       X    UBA   ADD       X                  UBB  ADD                        X:=X + #SHIFTS;
6245                                                                                       UBB:=(S) SHIFTED UNTIL 1 SHIFTED OUT, JSB
6246     0631  SCN1           ADD                 NEXT       SREG ADD       RA   CCA       NEXT; (S):=SREG, CCA
6248     0632  SCX0 0010 SP4A ADDL      X                       ADD                CCA  NEXT  X:=X + 16; CCA ON 0, NEXT
6250           *
6251           ********************************************************************
6252           *    TBC (Test Bit and set Condition code) instruction             *
6253           ********************************************************************
6254           *
6255           $LUT INSTR=TBC:0 001 011 010 XX,SR=1,DSPL=6,SR=1,ENTRY=TBC
6256           $LUT INSTR=TBC:0 001 111 010 XX,SR=1,DSPL=6,SR=1,X,ENTRY=TBC
6257           *
6258     0633  TBC  XC   DSPL ADD                            ADD                SF1       UBA:=CIR.(12:4)+XC; SF1 IF TBC
6260     0634  TBC1           ADD                 000F UBA   ANDL                         ; UBB:=UBA.(12:4);
6262     0635                 ADD                 002F UBB   SUBL       CTR               ; CTR:=(47 - BIT POSITION) FOR REGN
6264     0636                 ADD           F1            ADD                          SKIP RSB IF TBC, WAIT ONE CLOCK
6266     0637                 ADD           RSB   RA   REGN AND                CCA  NEXT  RSB IF NOT TBC; TEST BIT, CCA, RSB
6268           *
6269           ********************************************************************
6270           *    TRBC (Test and Reset Bit, set Condition code) instruction     *
6271           ********************************************************************
6272           *
6273           $LUT INSTR=TRBC:0 001 011 011 XX,SR=1,DSPL=6,SR=1,ENTRY=TRBC
6274           $LUT INSTR=TRBC:0 001 111 011 XX, SR=1,DSPL=6,X,ENTRY=TRBC
6275           *
6276     0638  TRBC XC   DSPL ADD                            JSB  TBC1              UNC   UBA:=CIR.(12:4)+IF INDEXED THEN X; JSB
6278     0639                 ADD                      REGN CAD                          ; UBB:=NOT BIT
6280     063A                 ADD                 RA   UBB  AND        RA            NEXT  ; RESET BIT, NEXT
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6283            ************************************************************************
6284            *     TSBC (Test and Set Bit, set Condition code) instruction          *
6285            ************************************************************************
6286            *
6287            $LUT INSTR=TSBC:0 001 011 100 xx, SR=1, DSPL=6, SR=1, ENTRY=TSBC
6288            $LUT INSTR=TSBC:0 001 111 100 xx, SR=1, DSPL=6, SR=1, X, ENTRY=TSBC
6289            *
6290    063B    TSBC XC   DSPL ADD                              JSB  TBC1          UNC   UBA:=CIR.(12:4)+IF INDEXED THEN X; JSB
6292    063C                   ADD                  RA   REGN IOR       RA        NEXT  ; SET BIT
6294            *
6295            ************************************************************************
6296            *     TCBC (Test and Complement Bit, set Condition code) instruction   *
6297            ************************************************************************
6298            *
6299            $LUT INSTR=TCBC:0 001 011 101 XX,SR=1,DSPL=6,SR=1,ENTRY=TCBC
6300            $LUT INSTR=TCBC:0 001 111 101 XX,SR=1,DSPL=6,SR=1,X,ENTRY=TCBC
6301            *
6302    063D    TCBC XC   DSPL ADD                              JSB  TBC1          UNC   UBA:=CIR.(12:4)+IF INDEXED THEN X; JSB
6304    063E                   ADD                  RA   REGN XOR       RA        NEXT  ; COMPLEMENT BIT
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6307            ********************************************************************
6308            *    EXF (Extract Field) Instruction                               *
6309            ********************************************************************
6310            *
6311            $LUT INSTR=EXF:0 010 110 1XX XX,SR=1,DSPL=4,ENTRY=EXF
6312            *
6313    063F    EXF       DSPL ADD                    0070      ADDL      SP1B            UBA:=K; SP1B:=MASK FOR MODULO 8
6315    0640          UBA UBA  ADD  LSL               0020 UBA  ADDL      CTR             UBA:=K & LSL(2); CTR:=K + OFFSET FOR REGN
6317    0641          UBA UBA  ADD  LSL               FFF0      ADDL      SP2B            UBA:=K & LSL(4); SP2B:=-8
6319    0642          UBA CIR  JSB  EXFS      BIT8    FFFF REGN ADDL      RH              UBA.(8:4):=J + K, SKIP IF (J+K) MOD 16 >= 8;
6321                                                                                      RH:=MASK WITH K RIGHTMOST BITS
6322    0643                RA   ADD                   UBA  SP1B RPCA              ZERO  UBA:=(S);
6324                                                                                      UBB.(9:3):=(J+K) MOD 8, REPEAT IF <> 0
6325    0644                UBA  CSL                   UBB  SP2B ADD               ZERO  CSL UNTIL COUNT IN UBB.(9:3) IS DECREMENTED
6327                                                                                        TO 0
6328    0645          RH    SREG AND       RA   CCA                ADD               NEXT  (S):=K BITS OF (S) STARTING AT J, CCA; NEXT
6330    0646    EXFS        RA   ADD  SWAB            RREG      REPC              ZERO  UBA:=(S) & CSL(8), REPEAT IF COUNT <> 0
6332    0647                UBA  CSL                   UBB  SP2B ADD               ZERO  CSL UNTIL COUNT IN UBB.(9:3) IS DECREMENTED
6334                                                                                        TO 0
6335    0648          RH    SREG AND       RA   CCA                ADD               NEXT  (S):=K BITS OF (S) STARTING AT J, CCA; NEXT
6337            *
6338            ********************************************************************
6339            *    DPF (Deposit Field) Instruction                               *
6340            ********************************************************************
6341            *
6342            $LUT INSTR=DPF:0 010 111 0XX XX,SR=2,DSPL=8,ENTRY=DPF
6343            *
6344    0649    DPF       DSPL ADD                    ADD                             UBA:=J,K;
6346    064A          UBA UBA  ADD  LSL               000F UBA  ANDL                  UBA:=J,K & LSL(2); UBB:=K
6348    064B          UBA UBA  ADD  LSL               0020 UBB  ADDL      CTR         UBA:=J,K & LSL(4); UBB:=K + OFFSET FOR REGN
6350    064C          UBA DSPL ADD  RLZ       NEG             ADD                     UBA.(4:4):=K, UBA.(0:4):=(J+K) MOD 16,
6352                                                                                        SKIP IF >= 8
6353    064D          RREG UBA ADD            UNC     FFFF REGN ADDL                  UBA.(4:4):=(J+K), SKIP;
6355                                                                                      UBB:=MASK WITH K RIGHTMOST BITS
6356    064E                UBB  JSB  DPFS      UNC     0700 UBA  ANDL      CTX       JSB IF ((J+K) MOD 16) > 8; CTX:=(J+K) MOD 8
6358    064F                UBA  CAD                    UBA  RA   RPCA          DCTX CTX0 UBA:=K MASK COMPLEMENTED;
6360                                                                                      UBB:=K RIGHT BITS OF (S), REPEAT IF CTX > 0
6361    0650    DPF1        UBA  CSR                         UBB  CSR             DCTX CTX0 CSR BOTH ALU'S UNTIL CTX IS DECREMENTED TO 0
6363    0651          RB    SREG AND                         SREG ADD             EPOP      UBA:=(S-1) WITH K BITS STARTING AT J ZEROED;
6365                                                                                      UBB:=K BITS OF (S) SHIFTED TO J, EPOP
6366    0652                      ADD               NEXT  UBA  UBB  IOR       RB  CCA      NEXT; DEPOSIT K BITS OF (S) INTO (S-1), CCA
6368    0653    DPFS        UBA  ADD  SWAB                   RA   ADD  SWAB              UBA:=K MASK & CSR(8); UBB:=(S) & CSR(8)
6370    0654                UBA  JSBC DPF1      UNC   UBA  UBB  RPCA          DCTX CTX0 UBA:=NOT K MASK, JSB;
6372                                                                                      UBB:=K BITS OF (S), REPEAT IF CTX <> 0
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6374           ************************************************************************
6375           *   The following immediate instructions use the ADSB setting in     *
6376           *   the LUT to overlap the following pairs of instructions: LDI      *
6377           *   and LDN, LDXI and LDXN, CMPI and CMPN, ADDI and SUBI, and ADXI   *
6378           *   and SBXI.                                                        *
6379           ************************************************************************
6380           *
6381           ************************************************************************
6382           *    LLNI is the entry point for LDI and LDNI                          *
6383           ************************************************************************
6384           *
6385           $LUT INSTR=LDI:0 010 001 0xx xxx xxx, ENTRY=LLNI, DSPL=8
6386           $LUT INSTR=LDNI:0 010 101 0xx xxx xxx, ENTRY=LLNI, DSPL=8, F2
6387           *
6388    0655   LLNI      DSPL ADSB      RH   CCA             JSZ  PSHM           SR7   RH:= +/-(IMMEDIATE OPERAND), CCA,
6390                                                                                   PUSH IF SR = 7
6391    0656                 ADD            EPSH            JSZ  NEXT           UNC
6392           *
6393           ************************************************************************
6394           *    LXNI is the entry point for LDXI and LDXN                         *
6395           ************************************************************************
6396           *
6397           $LUT INSTR=LDXI:0 010 001 1XX XX,DSPL=8,ENTRY=LXNI
6398           $LUT INSTR=LDXN:0 010 101 1XX XX,DSPL=8,F2,ENTRY=LXNI
6399           *
6400    0657   LXNI      DSPL ADSB      X    CF2             JSZ  NEXT           UNC   X:= +/-(IMMEDIATE OPERAND); JSB FOR NEXT
6402           *
6403           ************************************************************************
6404           *    ADSI is the entry point for ADDI and SUBI                         *
6405           ************************************************************************
6406           *
6407           $LUT INSTR=ADDI:0 010 010 1XX XX,DSPL=8,ENTRY=ADSI,SR=1
6408           $LUT INSTR=SUBI:0 010 011 0XX XX,DSPL=8,F2,ENTRY=ADSI,SR=1
6409           *
6410    0658   ADSI RA   DSPL ADSB      RA   CCO             JSZ  NEXT           UNC   S:=S + OR - IMMEDIATE OPERAND, CCO; JSB NEXT
6412           *
6413           ************************************************************************
6414           *    MPYI (Multiply immediate) Instruction                             *
6415           ************************************************************************
6416           *
6417           $LUT INSTR=MPYI:0 010 011 1XX XX,SR=1,DSPL=8,ENTRY=MPYI
6418           *
6419    0659   MPYI      CIR  ADD  RRZ  SP4A            RA   ADD            HBF2       SP4A:=U; F2:=SIGN(V)
6421    065A              UBA  ADD  RLZ  RH                 REPN           CLO  7     RH:=U & LSL(8); UBB:=0, REPEAT 8 TIMES, CLO
6423    065B                   MPAD                  RA   UBB  LINK           DCTR CTR0  MPAD UNTIL CTR IS DECREMENTED TO 0
6425    065C                   ADD                        UBB  ADD                      ; UBB:=MSW
6427    065D       SPOA        ADD  LRZ  SPOA        UBB  RH   XSUB                     UBA:=SPOA:=LSW.(8:8);
6429                                                                                   UBB:=IF V<0 THEN MSW - (U*2**16) ELSE MSW
6430    065E              UBB  ADD  RLZ               FF80 UBB  ANDL                NZRO UBA (0:8):=LSW (0:8); SKIP IF MS 9 BITS<>0
6432    065F       SPOA UBA  IOR       RA   CCA       RREG UBB  XOR                 NEXT (S):=RESULT, CCA,
6434                                                                                   NEXT, UBB:=0 IF MS 9 BITS ALL 1'S
6435    0660                   JSZ  NEXT       UNC         UBB  JSZ  IRO            NZRO  NEXT UNLESS...; MS 9 BITS NOT ALL THE SAME
```

```
RECORD  C S.            ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6438            ********************************************************************
6439            *    DIVI (Divide immediate) Instruction                           *
6440            ********************************************************************
6441            *
6442            $LUT INSTR=DIVI 0 010 100 0xx xxx xxx, SR=1, DSPL=8, ENTRY=DIVI
6443            *
6444    0661    DIVI      DSPL JSZ  TRP4      ZERO          ADD                     TRAP IF DIV BY 0;
6446    0662                   ADD                  RA       REPC      SP1B      POS   ; SP1B:=V, REPC IF NEG
6448    0663                   ADD                       RA  SUB       SP1B SF2  UNC   ; SP1B:=-V & F2:=1 IF V<0, DELAY 1 CLOCK
6450    0664                   ADD       SPOA            SP1B REPN                 1O    SPOA:=UBA:=0; UBB:=V, REPEAT 17 TIMES
6452    0665       UBA  DSPL DVSB                UBB       LINK                DCTR CTR0  DVSB UNTIL CTR IS DECREMENTED TO 0
6454    0666       UBA       ADD                 UBB       ADD                       WAIT ONE CLOCK FOR ANSWER TO BE IN SP4A
6456    0667                  ADD                       SP4A ADSB      RA   CCA  NEXT  ; (S):=W, CCA, NEXT
6458            *
6459            ********************************************************************
6460            *    CNMI is the entry point for CMPI and CMPN.  F2 is set to      *
6461            *    perform a SUB for CMPI and cleared to peform an add for CMPN. *
6462            ********************************************************************
6463            *
6464            $LUT INSTR=CMPI:0 010 010 0XX XX,ENTRY=CMNI,DSPL=8,F2,SR=1
6465            $LUT INSTR=CMPN:0 010 110 0XX XX,ENTRY=CMNI,DSPL=8,SR=1
6466            *
6467    0668    CMNI RA   DSPL ADSB      SP4A NOFL          INC             EPOP      SP4A:=UBA:=(S) +/- IMMEDIATE OPERAND,
6469                                                                                   SKIP NEXT IF NOT OVERFLOW;
6470                                                                                  EPOP, UBB:=1
6471    0669                   JSZ  NEXT      UNC   RA   UBB  IOR             CCA       NEXT IF OVERFLOW;
6473                                                                                  SIGN(UBB):=SIGN(S), BUT UBB CANNOT = 0, CCA
6474    066A           SP4A ADD             CCA             ADD                  NEXT  CCA ON (S) - IMMEDIATE OPERAND IF NOFL; NEXT
6476            *
6477            ********************************************************************
6478            *    ASXI is the entry point for ADXI and SBXI                     *
6479            ********************************************************************
6480            *
6481            $LUT INSTR=ADXI:0 011 010 1XX XX,DSPL=8,ENTRY=ASXI
6482            $LUT INSTR=SBXI:0 011 011 0XX XX,DSPL=8,F2,ENTRY=ASXI
6483            *
6484    066B    ASXI X    DSPL ADSB      X     CCA             JSZ  NOP            UNC   X:=X +/- (IMMEDIATE OPERAND), CCA; JSB NOP
6486            *
6487            ********************************************************************
6488            *    OR Immediate instruction  (ORI)                               *
6489            ********************************************************************
6490            *
6491            $LUT INSTR=ORI:0 011 110 1XX XX,DSPL=8,ENTRY=ORI,SR=1
6492            *
6493    066C    ORI  RA   DSPL IOR       RA    CCA             JSZ  NOP            UNC   (S):=(S) IOR (IMMEDIATE OPERAND), CCA;
6495                                                                                   JSB FOR NOP
6496            *
6497            ********************************************************************
6498            *    EXclusive OR Immediate  (XORI)                                *
6499            ********************************************************************
6500            *
6501            $LUT INSTR=XORI:0 011 111 0XX XX,DSPL=8,ENTRY=XORI,SR=1
6502            *
6503    066D    XORI RA   DSPL XOR       RA    CCA             JSZ  NOP            UNC   (S):=(S) XOR (IMMEDIATE OPERAND), CCA;
6505                                                                                     JSB FOR NOP
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6507           ********************************************************************************
6508           *    AND Immediate  (ANDI)                                                     *
6509           ********************************************************************************
6510           *
6511           $LUT INSTR=ANDI:0 011 111 1xx xxx xxx, DSPL=8, ENTRY=ANDI, SR=1
6512           *
6513     066E  ANDI RA   DSPL AND       RA   CCA             JSZ  NOP            UNC   (S):=(S) AND (IMMEDIATE OPERAND), CCA;
6515                                                                                     JSB FOR NEXT
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6517            ********************************************************************
6518            *    ADSB is the entry point for the ADD and SUB instructions.     *
6519            ********************************************************************
6520            *
6521            $LUT INSTR=ADD:0 000 010 000 XX,ENTRY=ADSB,SR=2
6522            $LUT INSTR=SUB:0 000 010 001 XX,F2,ENTRY=ADSB,SR=2
6523            *
6524    066F    ADSB RB    RA    ADSB       RB    CCO             JSL  DEL            UNC   (S-1):=(S-1) +/- (S), CCO;
6526                                                                                        JSB TO DELETE (S) AND FOR NEXT
6527            *
6528            ********************************************************************
6529            *    MPY (Multiply) Instruction                                    *
6530            ********************************************************************
6531            *
6532            $LUT INSTR=MPY:0 000 010 010 XX,ENTRY=MPY,SR=2
6533            *
6534    0670    MPY        RB    ADD        SP4A HBF2            REPN           CLO  F      SP4A:=U, F2:=SIGN(U); REPEAT 16 TIMES, CLO
6536    0671               MPAD                      RA   UBB  LINK           DCTR CTR0   RREG:=0; MPAD UNTIL CTR IS DECREMENTED TO 0
6538    0672       RA      ADD                HBF2       UBB  ADD            POP         F2:=SIGN(V), (RA MUST BE SPECIFIED IN RREG);
6540                                                                                        UBB:=MSW, POP (ALLOWS OVERLAP WITH MPYM)
6541    0673               RB    ADD                      UBB  RA   XSUB                 UBA:=U;
6543                                                                                        UBB:=IF U<0 THEN MSW - (V*2**16) ELSE MSW
6544    0674    MPY1 SP0A        ADD        RA    CCA       UBB  UBA  XSUB                 (S):=LSW, CCA;
6546                                                                                        UBB:=IF V<0 THEN MSW - (U*2**16) ELSE MSW
6547    0675               UBB   JSZ  IRO          NZRO      UBA  UBA  LINK                 UBA:=MSW + LSW(0), JSB TO SET OVERFLOW
6549                                                                                          IF MS 17 BITS ARE NOT THE SAME
6550    0676                     ADD                                    ADD            NEXT  ; NEXT
6552            *
6553            ********************************************************************
6554            *    DIV (Divide) Instruction                                      *
6555            ********************************************************************
6556            *
6557            $LUT INSTR=DIV:0 000 010 011 XX,ENTRY=DIV,SR=2
6558            *
6559    0677    DIV        RB    ADD        XR0   POS           RA   ADD            HBF2 NZRO  XR0:=U, SKIP IF>=0; F2:=SIGN(V), SKIP IF<>0
6561    0678               RB    SUB        RB                RB   JSZ  TRP4             UNC   (S-1):=-(U) IF U < 0;
6563                                                                                        UBB:=U, TRAP IF DIVIDE BY ZERO
6564    0679       RA      UBB   XOR                HBF2       RA   ADSB      RH           F2:=(SIGN(V) XOR SIGN(U)); RH:=ABS(V)
6566    067A                     JSB  DVL2          UNC        RB   REPN           CLO  10    UBA:=0, JSB;
6568                                                                                        UBB:=ABS(U), CLO, REPN JSB TARGET 17 TIMES
6569            *
6570            ********************************************************************
6571            *    NEG (Negate) Instruction                                      *
6572            ********************************************************************
6573            *
6574            $LUT INSTR=NEG:0 000 010 100 XX,ENTRY=NEG,SR=1
6575            *
6576    067B    NEG        RA    SUB        RA    CCO             JSZ  NEXT           UNC   (S):=-(S); JSB TO NEXT
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6579           ************************************************************************
6580           *     CMP [Compare] Instruction                                        *
6581           ************************************************************************
6582           *
6583           $LUT INSTR=CMP:0 000 001 111 xxx xxx, ENTRY=CMP, SR=2
6584           *
6585     067C  CMP   RB    RA    SUB        SP4A NOFL            INC            EPP2       SKIP IF NO OVERFLOW ON (S-1) - (S);
6587                                                                                      UBB:=1, EPP2
6588     067D                      JSZ  NEXT       UNC   RB   UBB  IOR            CCA        NEXT IF OVERFLOW;
6590                                                                                      SET CCL ON (S-1), UBB CANNOT EQUAL ZERO
6591     067E                SP4A ADD             CCA             ADD                 NEXT  SET CCA ON (S-1) - (S); NEXT
```

```
RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6594           *****************************************************************
6595           *    DADS is the entry point for the DADD and DSUB instructions.    *
6596           *****************************************************************
6597           *
6598           $LUT INSTR=DADD:0000001001XX,ENTRY=DADS,SR=4
6599           $LUT INSTR=DSUB:0000001010XX,ENTRY=DADS,SR=4,F2
6600           *
6601    067F   DADS RD    RB   ADSB           CCO   RC   RA   LINK       RC   EPP2       UBA:=MSW((S-3),(S-2) +/- (S-1),(S)), CCO;
6603                                                                                    NEW (S):=LEAST SIGNIFICANT WORD, EPP2
6604    0680               UBA  ADD       RD              UBB  ADD            DCC  NEXT  NEW (S-1):=MOST SIGNIFICANT WORD;
6606                                                                                    DCC ON LEAST SIGNIFICANT WORD, NEXT
6607           *
6608           *****************************************************************
6609           *    MPYL (Multiply long) Instruction - U is multiplied by V.       *
6610           *****************************************************************
6611           *
6612           $LUT INSTR=MPYL:0 000 001 011 XX,ENTRY=MPYL,SR=2
6613           *
6614    0681   MPYL        RA   ADD       SP4A HBF2                REPN           CLO  000F  SP4A:=V, F2:=sign(V); repeat 16 times, CLO
6616    0682                    MPAD                       RB   UBB  LINK           DCTR CTRO  RREG:=0; MPAD UNTIL CTR IS DECREMENTED TO 0
6618    0683         RB         ADD            HBF2             UBB  ADD                     F2:=SIGN(U) (RB MUST BE SPECIFIED IN RREG);
6620                                                                                    UBB:=MSW;
6621    0684         SPOA       ADD       RA           UBB  RB   XSUB                    (S):=LSW;
6623    0685                    ADD                    UBB  RA   XSUB       RB           ; (S-1):=IF U < 0 THEN MSW - (V*2**16)
6625                                                                                                ELSE MSW
6626    0686               UBB  JSZ  DTS1      UNC   RA   RA   LINK           CCRY        UBA := msw + lsw(0); Clear carry bit in STA
6628           *
6629           *****************************************************************
6630           *    DIVL (Divide long) Instruction - U is divided by V.  *         *
6631           *****************************************************************
6632           *
6633           $LUT INSTR=DIVL:0 000 001 100 XX,ENTRY=DIVL,SR=3
6634           *
6635    0687   DIVL        RC   ADD       XR0   HBF2       RA   SUB            POP  NEG   XR0:=MSU (AFTER POP RH=MSU), F2:=SIGN(U);
6637                                                                                    SKIP IF V < 0, POP TO ALLOW OVERLAP WITH
6638                                                                                      DIV AND LDIV INSTRUCTIONS AND CORRECT
6639                                                                                      STACK FOR DIVL AND LDIV IF 0 DIVIDE TRAP
6640    0688         RA   RC   XOR                 HBF2       UBB  JSZ  TRP4 RH         ZERO  F2:=SIGN(W);
6642                                                                                    RH:=-(V) IF <= 0, TRAP IF DIVIDE BY ZERO
6643    0689              RB   ADSB      RB                    RA   LINK       SP1B             RB:=UBA:=ABS(MSU); SP1B:=UBB:=ABS(LSU)
6645    068A         UBA  RH   DVSB            NCRY  UBB        LINK                        FIRST DVSB, SKIP IF NOT (MSU >= V);
6647    068B         UBA       JSB  DVL1      UNC   UBB        REPN           CLO  000F  Overflow if msU >= V, JSB, (stops REPN);
6649                                                                                      Repeat 16 times, CLO
6650    068C   DVL2  UBA  RH   DVSB                      UBB        LINK           DCTR CTRO  DVSB UNTIL CTR IS DECREMENTED TO 0
6652    068D         UBA       ADD  LSR  RA   F1HB  UBB        XFRS       SP3B F2HB       (S):=REM; SP3B(0):=(SIGN(U) XOR SIGN(V))
6654    068E         UBA  XR0  XOR            POS             SP4A ADSB       RB   CCA  ZERO  SKIP IF (SIGN(REM) = SIGN(U));
6656                                                                                    (S-1):=QUOTIENT, CCA, SKIP IF QUOTIENT = 0
6657    068F              RA   SUB       RA          SP3B UBB  JSZ  IRO            NEG   (S):=-REM;
6659                                                                                    JSB IF (SIGN(QUOTIENT) <>
6660                                                                                      (SIGN(U) XOR SIGN(V)))
6661    0690                   ADD                                 ADD                NEXT  ; NEXT
6663    0691   DVL1       SP1B ADD       SP0A                     ADD                      SP0A:=LSU;
6665    0692                   ADD            RSB        RB   REPN           SOV  0020  UBA:=0, RSB; UBB:=MSU, REPEAT 33 TIMES, SOV
```

```
RECORD  C S             ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6668            ********************************************************************
6669            *    DNEG (Double Negate) Instruction                              *
6670            ********************************************************************
6671            *
6672            $LUT INSTR=DNEG:0 000 001 101 xxx xxx, ENTRY=DNEG, SR=2
6673            *
6674    0693    DNEG      RB   SUB            CCO         RA   LINK      RA              CCO ON MSW OF NEGATED DOUBLE WORD; (S):=LSW
6676    0694              UBA  ADD       RB                UBB  ADD           DCC  NEXT  (S-1):=MSW; DCC ON LSW, NEXT
6678            *
6679            ********************************************************************
6680            *    DCMP (Double Compare) Instruction                             *
6681            ********************************************************************
6682            *
6683            $LUT INSTR=DCMP,0 000 001 000 XX,ENTRY=DCMP,SR=4
6684            *
6685    0695    DCMP RD   RB   SUB            NOFL  RC   RA   LINK      SP1B EPP4       UBA,SP1B:=(OPERAND 1) - (OPERAND 2), SKIP
6687                                                                                    IF NO OVERFLOW, EPP4
6688    0696               ADD              UNC         UBA  ADD           CCA         SKIP IF OVFL; SET CCA ON MSW OF DIFFERENCE
6690    0697               INC              NEXT        SP1B ADD           DCC         UBA:=1, NEXT IF NOFL; SET DCC ON LSW
6692    0698         RD    UBA  IOR          CCA              ADD                 NEXT  SET CCA ON SIGN OF MSW OF OPERAND 1; NEXT
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6695           ************************************************************************
6696           *    The following code handles the SR preadjust for DMUL and          *
6697           *    DDIV.  Entry for both is through the optional instructions        *
6698           *    entry point (#GRP7).  SR is adjusted to 4.                        *
6699           ************************************************************************
6700           *
6701    0699   DMDV           JSZ  PULM      SRL2           JSZ  PUL2           SRZ   Pull 2 if SR = 0 else pull 1 if SR = 1
6702    069A         SR  SP4A  JSZ  PULM      ZERO  SR  SP4A JSZI PUL2           ZERO  Pull 2 if SR = 2 else pull 1 if SR = 3
6705                                                                                    (UBB = SR - 3 + 1) (UBA = SR - 3)
6706    069B              CIR  JSB  DDIV      ODD            ADD                       Jump if Double Divide instruction
6708           *
6709           ************************************************************************
6710           *    DMUL (Double Multiply) Instruction - Entry is through             *
6711           *    optional instructions entry point                                 *
6712           ************************************************************************
6713           *
6714    069C   DMUL       RD   ADD                HBF2           RB   ADD                 POS   F2:=SIGN(U); SKIP IF V<0
6716    069D              RB   ADD                POS            RA   SUB       RA   CTF1      SKIP IF V<0; (S):=-(S) IF V<0, F1:=CARRY
6718    069E              RB   SUB          RB   F1TC  RB       RD   XOR            HBF2      (S-1):=-(S-1) IF V<0; F2:=SIGN OF ANSWER
6720    069F              RD   ADSB                               RC   LINK                    UBA,UBB:=ABS(U)
6722    06A0              UBB  ADD          SP4A                  UBA  ADD       SP3B          SP4A:=LSU, SP3B:=MSU
6724    06A1                   ADD                               REPN                CLO  1F   UBA:=0, UBB:=0, CLO, REPEAT 32 TIMES
6726    06A2         RB   UBA  MPAD                    RA   UBB  LINK                DCTR CTR0 MPAD UNTIL CTR IS DECREMENTED TO 0
6728    06A3              UBA  ADD                     UBB  ADD                       UBA:=ABS(PRODUCT.(0:16));
6730                                                                                    UBB:=ABS(PRODUCT.(16:16))
6731    06A4         SP0A      ADD                POS   UBA  UBB  IOR                 EPP2 ZERO SKIP IF ABS(PRODUCT.(32:1)) = 0
6733                                                                                    SKIP IF ABS(PRODUCT.(0:32)) = 0, EPP2
6734    06A5   DMSO  SP0A      ADD                SF1   SP3B       ADD                 SOV  UNC  UBA:=ABS(PRODUCT.(32:16)), F1:=BIT 32
6736                                                                                    UBB:=ABS(PRODUCT.(48:16)), SOV AND SKIP IF
6737                                                                                      MS 32 BITS <> 0
6738    06A6              UBA  ADSB       RD   CCA            UBB  LINK                F1    NEGATE ABS(PRODUCT) IF PRODUCT < 0,
6740                                                                                      (S-1) AFTER EPP2 := PRODUCT.(32:16), CCA;
6741                                                                                    UBB:=PRODUCT.(48:16), SKIP IF BIT 32 = 1
6742    06A7   DMNX       UBA  ADD  LSL                         UBB  ADD            RC   DCC  NEXT  UBA:=PRODUCT.(33:15);
6744                                                                                    (S) AFTER EPP2 := PRODUCT.(48:16), DCC
6745                                                                                      NEXT IF MS 32 BITS OF ABS(PRODUCT) <> 0
6746                                                                                      OR MS 32 BITS = 0 AND BIT 32 = 0
6747    06A8                   JSB  DMSO       NF2   UBA  UBB  IOR                 ZERO  JSB IF EXPECTED SIGN IS POSITIVE;
6749                                                                                    SKIP IF PRODUCT.(33:31) = 0  (+/-(2**31))
6750    06A9                   JSB  DMNX       UNC                JSB  DMSO        UNC   JSB TO FINISH IF PRODUCT = -2**31
6752                                                                                    SOV TO INDICATE MS BITS ARE NOT THE SAME
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6754          ************************************************************************
6755          *    DDIV (Double Divide) Instruction - Entry is through               *
6756          *    optional instructions entry point                                 *
6757          ************************************************************************
6758          *
6759    06AA  DDIV RA   RB   IOR             NZRO       RB   ADD            HBF2       SKIP IF V=0; F2:=SIGN(V)
6761    06AB               JSZ  TRP4        UNC        RD   ADD            HBF2       JSB IF DIVIDE BY ZERO; F2:=SIGN(U)
6763    06AC          RB   ADSB                         RA   LINK       RA             UBA,RA:=ABS(V)
6765    06AD          UBA  ADD        RB         RB    RD   XOR            HBF2       RB:=ABS(MSV); F2:=(SIGN(U) XOR SIGN(V))
6767    06AE          RD   ADSB       SPOA               RC   LINK       SP3B           SPOA,SP3B:=ABS(U)
6769    06AF               ADD                           REPN           CLO  20    UBA,UBB:=0, REPEAT 33 TIMES, CLO
6771    06B0     UBA  RB   DVSB                  UBB  RA   LINK           DCTR CTR0  DVSB, DECREMENT CTR UNTIL EQUAL TO 0
6773    06B1     UBA       ADD  LSR  SP4A F1HB  UBB       LINK       RA             SP4A,(S):=ABS(REM)
6775    06B2          SP4A ADSB       RC   CTF1  SP3B       ADD                    (S-2):=LSW, F1:=CARRY; UBB:=ABS(MSW)
6777    06B3          UBB  ADSB       RD   F1TC  UBA  UBB  IOR            F2HB ZERO  (S-3):=MSW;
6779                                                                       UBB.(0:1):=(SIGN(U) XOR SIGN(V)),
6780                                                                         SKIP IF QUOTIENT ON TBUS EQUALS 0
6781    06B4     RD   SP4A XOR             HBF2  UBA  UBB  XOR                NEG    F2:=(SIGN(U) <> SIGN(ABS(REM)));
6783                                                                       SKIP IF SIGN(QUOTIENT) <>
6784                                                                         (SIGN(U) XOR SIGN(V))
6785    06B5          RD   ADD             CCA             ADD                UNC    SET CCA ON MSW;
6787                                                                       SKIP IF QUOTIENT=0 OR
6788                                                                         SIGN(QUOTIENT)=(SIGN(U) XOR SIGN(V))
6789    06B6          RA   ADSB       RA   CTF1              ADD            SOV        (S):=LS(REM); SOV (QUOTIENT IS +(2**32))
6791    06B7          SP4A ADSB       RB   F1TC        RC   ADD            DCC  NEXT  (S-1):=MS(REM); DCC ON LSW, NEXT
```

```
RECORD  C S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6794            ************************************************************************
6795            *    INCX (Increment X) Instruction                                    *
6796            ************************************************************************
6797            *
6798            $LUT INSTR=INCX:0 000 000 100 XX,ENTRY=INCX,SR=0
6799            *
6800    06B8    INCX X          INC       X    CCO             JSZ  NEXT          UNC   X:=X + 1, CCO; JSB FOR NEXT
6802            *
6803            ************************************************************************
6804            *    DECX (Decrement X) instruction                                    *
6805            ************************************************************************
6806            *
6807            $LUT INSTR=DECX:0 000 000 101 XX,ENTRY=DECX,SR=0
6808            *
6809    06B9    DECX X          CAD       X    CCO             JSZ  NEXT          UNC   X:=X - 1, CCO; JSB FOR NEXT
6811            *
6812            ************************************************************************
6813            *    INCA (Increment A) Instruction                                    *
6814            ************************************************************************
6815            *
6816            $LUT INSTR=INCA:0 000 011 011 XX,ENTRY=INCA,SR=1
6817            *
6818    06BA    INCA RA         INC       RA   CCO             JSZ  NEXT          UNC   (S):=(S) + 1; JSB FOR NEXT
6820            *
6821            ************************************************************************
6822            *    DECA (Decrement A) Instruction                                    *
6823            ************************************************************************
6824            *
6825            $LUT INSTR=DECA:0 000 011 100 XX,ENTRY=DECA,SR=1
6826            *
6827    06BB    DECA RA         CAD       RA   CCO             JSZ  NEXT          UNC   (S):=(S) - 1; JSB FOR NEXT
6829            *
6830            ************************************************************************
6831            *    INCB (Increment B) Instruction                                    *
6832            ************************************************************************
6833            *
6834            $LUT INSTR=INCB:0 000 111 011 XX,ENTRY=INCB,SR=2
6835            *
6836    06BC    INCB RB         INC       RB   CCO             JSZ  NEXT          UNC   (S-1):=(S-1) + 1; JSB FOR NEXT
6838            *
6839            ************************************************************************
6840            *    DECB (Decement B) Instruction                                     *
6841            ************************************************************************
6842            *
6843            $LUT INSTR=DECB:0 000 111 100 XX,ENTRY=DECB,SR=2
6844            *
6845    06BD    DECB RB         CAD       RB   CCO             JSZ  NEXT          UNC   (S-1):=(S-1) - 1; JSB FOR NEXT
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6848            ************************************************************************
6849            *    LCMP (Logical Compare) Instruction                                *
6850            ************************************************************************
6851            *
6852            $LUT INSTR=LCMP:0 000 101 111 XX,ENTRY=LCMP,SR=2
6853            *
6854    06BE  LCMP            ADD             EPOP  RB   RA   SUB            CCZ  CRRY  EPOP; SET CCZ ON (S-1) - (S), SKIP IF >=
6856    06BF                  JSZ  NEXT       EPOP             CAD            CCA        JSB FOR NEXT, EPOP; SET CCL IF (S-1) < (S)
6858            *
6859            ************************************************************************
6860            *    LADS is the entry point for the LADD and LSUB instructions.       *
6861            ************************************************************************
6862            *
6863            $LUT INSTR=LADD:0 000 110 000 XX,ENTRY=LADS,SR=2
6864            $LUT INSTR=LSUB:0 000 110 001 XX,ENTRY=LADS,SR=2,F2
6865            *
6866    06C0  LADS RB   RA    ADSB      RB    CCA   RB   RA   ADSB                 CRRY  (S-1):=(S-1) +/- (S), CCA; SKIP IF CARRY
6868    06C1                  ADD             EPOP             ADD            CCRY UNC   EPOP; CLEAR CARRY, SKIP IF NO CARRY
6870    06C2                  ADD             NEXT             ADD            SCRY       NEXT; SET CARRY
6872            *
6873            ************************************************************************
6874            *    LMPY (Logical Multiply) Instruction                               *
6875            ************************************************************************
6876            *
6877            $LUT INSTR=LMPY:0 000 110 010 XX,ENTRY=LMPY,SR=2
6878            *
6879    06C3  LMPY       RA   ADD       SP4A                   REPN           CCRY 000F  SP4A := V; UBB := 0, repeat 16 times, CCRY
6881    06C4                  MPAD                    RB   UBB  LINK           DCTR CTR0  RREG:=0; MPAD UNTIL CTR IS DECREMENTED TO 0
6883    06C5                  ADD                     UBB  ADD       RB        CCA  ZERO  ; (S-1):=MSW, CCA, SKIP IF = 0
6885    06C6       SPOA       ADD       RA                   ADD            SCRY       (S):=LSW; SCRY IF MSW <> 0
6887    06C7                  ADD                 UBA        ADD            DCC  NEXT  ; DCC ON LSW, NEXT
6889            *
6890            ************************************************************************
6891            *    LDIV (Logical Divide) Instruction                                 *
6892            ************************************************************************
6893            *
6894            $LUT INSTR=LDIV:0 000 110 011 XX,ENTRY=LDIV,SR=3
6895            *
6896    06C8  LDIV       RA   JSB  DIVL      ZERO                   ADD                JSB (#DIVL DOES POP AND TRAPS);
6898    06C9             RB   ADD       SPOA SF1   RC   RA   JSBS LDV2          NCRY  SPOA:=LEAST SIGNIFICANT DIVIDEND, SF1 TO
6900                                                                                    INHIBIT EFFECT OF DVSB ON RREG;
6901                                                                                    JSB IF MS DIVIDEND < DIVISOR
6902    06CA                  ADD             CF1   RC        REPN           SOV  0020  CF1, UBA:=0; UBB:=MSU, REPEAT 33 TIMES, SOV
6904    06CB  LDV1 UBA   RA   DVSB                   UBB       LINK           DCTR CTR0  DIVIDE BY V ANOTHER 17 TIMES
6906    06CC       UBA        ADD  LSR   RB   F1HB             ADD            EPOP       AFTER EPOP (S):=REMAINDER; EPOP
6908    06CD             SP4A ADD       RC   CCA                ADD                 NEXT  (S-1):=QUOTIENT MOD 2**16, CCA, NEXT
6910    06CE  LDV2       RC   JSB  LDV1      CF1        RB   REPN           CLO  0010  CF1, UBA:=MSU, JSB;
6912                                                                                    UBB:=LSU, REPEAT 17 TIMES, CLO
6913                                                                                    JSB FOR NEXT; CCB ON (S)
```

```
 RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 6915           *************************************************************************
 6916           *     Performs system clock, process clock simulation                   *
 6917           *                                                                       *
 6918           *     begin                                <<#TCKI: TCLK interrupt >>   *
 6919           * L1:   TCLK:=TCLK+K;                      << reset TCLK for 1ms int >> *
 6920           *       if TCLK < 0 then                   << check if TCLK went too>>  *
 6921           *                                          << negative               >>  *
 6922           *        if not ICSFLAG PC := PC+1;        << inc PCLK               >>  *
 6923           *        CR := CR+1;                       <<update CR               >>  *
 6924           *        go to L1;                                                       *
 6925           *       else                                                            *
 6926           *       if not ICSFLAG then PC:=PC+1;     << update process clock >>    *
 6927           *       update CIR display;               << do front panel display>>   *
 6928           *       CR:=CR+1;                         << inc COUNT REGISTER >>      *
 6929           *       IF CR=0 THEN                                                    *
 6930           *         begin                       <<#TICK: system clock interrupt>>*
 6931           *           if not (STAT.(1:1) and TON) then                            *
 6932           *             begin                         <<#TICO: cant do s/w int. >> *
 6933           *               SINCBIT:=1;                 << flag missed timer int. >> *
 6934           *               DINTFF:=1;                  << set deferred int F/F   >> *
 6935           *               (TR):=(TR)+(LR);            << update TSLI (TR)       >> *
 6936           *             end                                                       *
 6937           *           else                                                        *
 6938           *             begin                         <<#TIC1 set up s/w int >>   *
 6939           *               SINCBIT:=0;                                             *
 6940           *               TON:=0;                     << disable sys clock ints >> *
 6941           *               TR:=0;                      << TSLI is now 0 >>         *
 6942           *               PLABEL:=!8C01;                                          *
 6943           *               INT9;                       << ICS setup,xfer to PLABEL>>*
 6944           *             end;                                                      *
 6945           *           end;                                                        *
 6946           *         RESTART instruction processing;                               *
 6947           *       end;        << TCLK int handler >>                            *
 6948           *                                                                       *
 6949           *   K = clock constant ( ( 1ms/clock period ) - 2 )                     *
 6950           *************************************************************************
 6951           *
 6952  06CF TCLK          JSZ  TIME      UNC   00A4       ADDL      SP2B                Add delta t to normal execution count
 6954  06D0       RH      ADD       SP4A                 XR31 ADD       BKX7                Store mask to clear TCLKINT in SP4A4  (2635)
 6956                                                                                      Store CR value into BKX7              (2635)
 6957  06D1       0014    ADDL      RH           0030      ADDL      CTR                 RH := %20 for read of (LR) & (TR);
 6959                                                                                        CTR := REGN code for TCLK
 6960  06D2 TC1   3413    ADDL                             SP4A ADD            CCPX      UBA := (1 ms / 75.0 ns) - 2;          (2635)
 6962  06D3               ADD                       UBA   REGN ADD       REGN          Clear TCLKINT (UBB = !0A00)
 6963  06D4       1000 CPX2 ANDL                     UBB        JSB  TC2           POS    TCLK := TCLK + clock constant
 6965                                                                                      UBA := ICSFLAG                        (2635)
 6967  06D5       UBA     ADD                       UBA        ADD                NZRO   If TCLK >= 0, go to TC2                (2635)
 6968                                                                                      UBA := ICSFLAG                        (2635)
 6970  06D6       UBA     ADD                             XR20 INC       XR20           Skip next line if ICSFLAG             (2635)
 6971                                                                                      UBA := ICSFLAG                        (2635)
 6973  06D7       UBA     ADD                             BKX7 JSBI TC1  BKX7      UNC  Increment PCLK if not ICS             (2635)
 6974                                                                                      UBA := ICSFLAG                        (2635)
 6976                                                                                      Inc CR and repeat the loop            (2635)
 6977  06D8 TC2       RH  INC              ROB3         UBA  ADD                 NZRO   UBA := %21, read (LR); Skip if ICSFLAG
 6979  06D9               JSB  CIRD        UNC          XR20 INC       XR20           Update CIR display; Inc PCLK if not on ICS
 6981  06DA               JSZ  TIME        UNC   000F       ADDL      SP2B           JSB to #TIME to set countsave = TCLK;
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

6983                                                                        SP2B := STOR NOP to throw away delta t
6984    06DB        RH         CAD             ROX3       BKX7 JSBI TICR XR31      NEG  UBA := %17,read (TR);(CR) := (CR) + 1,(2635)
6986                                                                          restart instruction if < 0
6987          **** tick:  CR has been incremented to 0, may do s/w interrupt
6988          *
6989    06DC  TICK      STA  ADD  LSL       POS   UBB  OPB  ADD       RH            Skip if interrupts are off; RH := 0 + (LR)
6991    06DD            XR15 JSB  TIC1      BIT8       OPB  SUB       XR31 SF1      Service interrupt if interrupts on and TON;
6993                                                                          (CR) := -(LR), set F1
6994          *      TIC0:  (interrupts LAND TON) = 0 so don't do timer interrupt
6995          *
6996    06DE  TIC0      XR15 ADD       XR15 F1HB  0007       ADDL                  Set SINC bit; UBB := mask to set DINTFF
6998    06DF        RH  OPA  ADD            DATA       UBB  ADD            CCPX     (TR) := (LR) + (TR); Set DINTFF
7000    06E0  TICR 1000 STA  ANDL                          JSZ  RSRT           UNC  UBA := right stackop bit; Restart instr
7002          *
7003          *      TIC1:  (interrupts LAND TON) = 1 so do timer interrupt
7004          *
7005    06E1  TIC1      UBA  ADD       SPOA SF2   7F7F       ADDL                  SPOA := param [(TR)+(LR)], set F2 for #INT9;
7007                                                                          UBB := mask to clear SINC bit and TON
7008    06E2        UBB XR15 AND       XR15 CF1   8C01       ADDL      SP1B        Clear SINC bit and TON, clear F1 for #IR1;
7010                                                                          SP1B := external label (%106001)
7011    06E3                JSZ  INT9      DATA              ADD                   (TR) := 0, service interrupt;
```

```
RECORD    C.S.          ********** ALU A **********  ********** ALU B **********
  NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7014            ***************************************************************
7015            *     CIR Update Routine:  for LP and PP systems just return    *
7016            ***************************************************************
7017            *
7018      06E4  CIRD          ADD             RSB             ADD                       Just return for now
7020      06E5                ADD                             ADD                       NOP to keep system happy          (2555)
7022      06E6                ADD                             ADD                       NOP to keep system happy          (2555)
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7025           ************************************************************************
7026           *    PLSA is the entry point for PLDA and PSTA instructions.           *
7027           *    Q relative addressing is specified in the LUT entry to            *
7028           *    negate the effect of FSS in the calculation of TICB.              *
7029           ************************************************************************
7030           *
7031           $LUT INSTR=PLDA/PSTA:0 010 000 011 01x xxx, Q, ENTRY=PLSA
7032           *
7035           %0700
7036           *
7037    0700   PLSA       CIR  JSB  PSTA       ODD   SR    SM    JSZ  TRP6 SP3B       NPRV  JSB IF PSTA INSTRUCTION;
7039                                                                                         SP3B:=SR + SM, TRP6 IF NPRV
7040           *
7041           ************************************************************************
7042           *    PLDA (Privileged load from absolute address) Instruction          *
7043           ************************************************************************
7044           *
7045    0701   PLDA             JSZ  PSHM       SR7             ADD        BKX3              JSB TO EMPTY A TOS REG IF SR = 7; BKX3:=0
7047    0702          X          ADD             ROB3      BNKS ADD                    NZRO  UBA:=ADDR, READ; SKIP IF BNKS <> 0
7049    0703          UBA  SM    CAD             EPSH  SP3B UBA  SUB        CTR       TICB  EPSH; CTR:=(S - ADDR), SKIP IF ADDR > SM AND
7051                                                                                           AND S >= ADDR AND BNKS = 0
7052    0704                     ADD                       OPB  ADD        RH   CCA  NEXT  ; RH:=OPERAND, CCA, NEXT
7054    0705                     ADD                       REGN ADD        RH   CCA  NEXT  ; RH:=WORD IN TOS, CCA, NEXT
7056           *
7057           ************************************************************************
7058           *    PSTA (Privileged store into absolute address) Instruction         *
7059           ************************************************************************
7060           *
7061    0706   PSTA             JSZ  PULM       SRZ             ADD        BKX3 TICB         JSB TO LOAD ONE TOS IF SR = 0;
7063                                                                                         BKX3:=0, TICB LOADS CTR WITH CODE FOR NOP
7064    0707          X          ADD             WRX3      BNKS ADD                    NZRO  UBA:=ADDR, WRITE USING BKX3;
7066                                                                                         SKIP IF BNKS <> 0
7067    0708          UBA  SM    CAD             EPOP  SP3B UBA  SUB        CTR       TICB  EPOP; CTR:=(S - ADDR), SKIP IF ADDR > SM AND
7069                                                                                           S >= ADDR AND BNKS = 0
7070    0709               RA   ADD        REGN DATA            ADD                    NEXT  WRITE (S); NEXT
7072           *
7073           ************************************************************************
7074           *    LSSD is the entry point for LSEA, SSEA, LDEA, and SDEA.           *
7075           *    The SR preadjust is set to the minimum value, 2, required         *
7076           *    for any of the four instructions.  Q is specified in the LUT      *
7077           *    to make the TICB option work properly.                            *
7078           ************************************************************************
7079           *
7080           $LUT INSTR=LSEA/SSEA/LDEA/SDEA:0 010 000 011 10, ENTRY=LSSD, SR=2, Q
7081           *
7082    070A   LSSD       CIR  ADD  LSR        EVEN  0003      ADDL       SP1B              SKIP IF NOT CIR(14); SP3B:=3
7084    070B                   JSB  DEA        UNC             JSZ  TRP6             NPRV  JSB IF LDEA OR SDEA INSTRUCTIONS;
7086                                                                                         TRAP IF NOT PRIVLEGED
7087    070C               CIR  JSB  SSEA       ODD   SR    SP1B SUB            CTF1      JSB if SSEA instruction; Set F1 if SR >= 3
7089           ************************************************************************
7090           *    LSEA (Load single word from extended address) instruction         *
7091           ************************************************************************
7092           *
7093    070D   LSEA             ADD                  00FF RB   ANDL       BKX3              ; BKX3 := bank adr.(8:8)
7095    070E          SR   SM    ADD        RH                UBB  JSZ  PSHM SP3B       SR7   RH := SR + SM, SP3B := bank adr (for rtn
7097                                                                                           from #PSHM), Push one TOS if SR = 7
```

```
RECORD   C.S.                 ********** ALU A **********  ********** ALU B **********
  NO     ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 7098    070F          RA   ADD             ROB3  UBB  BNKS SUB                  NZRO  UBA:=ADDR, READ; SKIP IF BANK ADDR <> BNKS
 7100    0710     UBA  SM   CAD             EPSH  RH   UBA  SUB        CTR       TICB  EPSH; CTR:=(S - ADDR), SKIP IF ADDR > SM AND
 7102                                                                                    AND S >= ADDR AND BANK ADDR = BNKS
 7103    0711          ---  ADD                                OPB  ADD        RH   CCA  NEXT  ; RH:=OPERAND, CCA, NEXT
 7105    0712          ---  ADD                                REGN ADD        RH   CCA  NEXT  ; RH:=WORD IN TOS, CCA, NEXT
 7107            *
 7108            ********************************************************************
 7109            *    SSEA (Store single word into extended address) instruction    *
 7110            ********************************************************************
 7111            *
 7112    0713  SSEA          ADD                       OOFF RC   ANDL       BKX3              ; BKX3 := bank adr.(8:8)
 7114    0714          RB   JSB  SSE1        WRX3            UBB  JSZ  PULM SP3B       NF1   Write using BKX3, jump if SR >= 3; Else
 7116                                                                                    SP3B := UBB (for rtn from #PSHM), push one
 7117                                                                                    TOS
 7118    0715          RB   ADD              WRX3      OOFF UBA  ANDL       BKX3              Write using BKX3; BKX3 := bank adr.(8:8)
 7120    0716  SSE1 SR  SM   ADD                       UBB  BNKS SUB                  NZRO  UBA:=SR + SM; SKIP IF BANK ADDR <> BNKS
 7122    0717       RB  SM   CAD              EPOP      UBA  RB   SUB        CTR       TICB  EPOP; CTR:=(S - ADDR), SKIP IF ADDR > SM AND
 7124                                                                                    S >= ADDR AND BANK ADDR <> BNKS
 7125    0718          RA   ADD              DATA                ADD                 NEXT  WRITE (S); NEXT
 7127    0719          RA   ADD        REGN                      ADD                 NEXT  STORE (S) IN TOS; NEXT
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7130            ********************************************************************
7131            *    DEA splits apart LDEA and SDEA instructions.                  *
7132            ********************************************************************
7133            *
7134    071A    DEA       CIR  JSB  SDEA      ODD   0002      ADDL      SP2B            JSB IF SDEA INSTRUCTIONS; SP2B:=2
7136            *
7137            ********************************************************************
7138            *    LDEA (Load double from extended address) Instruction          *
7139            ********************************************************************
7140            *
7141    071B    LDEA            JSZ  PSM2      SRG5  00FF RB   ANDL      BKX3            JSB IF SR > 5; BKX3:=BANK ADDR
7143    071C            RA      ADD            ROX3        RA   INC             ROX3      READ FIRST WORD; READ SECOND WORD
7145    071D                    ADD                  00FF RB   ANDL
7146    071E                    ADD            EPSH  UBB  BNKS JSBS LDE1            ZERO  EPSH; JSB IF BANK ADDR = BNKS
7148    071F            OPA     ADD       RH   CCA   SR   SM   ADD             EPSH      NEW (S-1):=FIRST WORD, CCA;
7150                                                                                     UBB:=SR + SM, EPSH
7151    0720                    ADD            NEXT  UBB  OPB  XFRS      RG    DCC       NEXT; RREG:=S, NEW (S):=SECOND WORD, DCC
7153    0721    LDE1 RA  SM     CAD            EPSH  RREG RA   SUB       CTR   TICB      EPSH; CTR:=IF ADDR > SM AND S >= ADDR THEN
7155                                                                                       (S - ADDR) ELSE CODE FOR OPERAND
7156    0722         RA  SM     SUB                  RREG RA   CAD       CTR   TICB      ; CTR:=IF (ADDR+1) > SM AND S >= (ADDR+1)
7158                                                                                       THEN (S -(ADDR+1)) ELSE CODE FOR OPERAND
7159    0723            REGN ADD       RH   CCA                ADD                     RH:=FIRST WORD, CCA;
7161    0724                    ADD            NEXT        REGN ADD       RG    DCC       NEXT; RG:=SECOND WORD, DCC
7163            *
7164            ********************************************************************
7165            *    SDEA (Store double word into extended address) Instruction    *
7166            ********************************************************************
7167            *
7168    0725    SDEA SR   SP1B JSZC PULM      NCRY  SR   SP2B JSZC PUL2            NCRY  PULL WORD INTO TOS IF NOT (SR > 3);
7170                                                                                     PULL 2 WORDS IF NOT (SR > 2)
7171    0726            RC     ADD            WRX3        BNKS ADD             EPP2      READ ADDR (S-2) (WILL BE VALID EITHER WAY);
7173    0727       00FF RD     ANDL                       UBB  ADD
7174    0728       UBA  UBB    JSBS SDE1      ZERO        UBA  ADD       BKX3            JSB IF BNKD = BANK ADDR; BKX3:=BANK ADDR
7176    0729            RB     ADD            DATA              ADD                     WRITE FIRST WORD;
7178    072A            RA     ADD            DATA  SR   SM   ADD                 NEXT  WRITE SECOND ADDR; UBB:=SR + SM, NEXT
7180    072B    SDE1 RC  SM     CAD                  UBB  RC   SUB       CTR   TICB      ; CTR:=IF ADDR > SM AND S >= ADDR THEN
7182                                                                                       (S - ADDR) ELSE CODE FOR NOP
7183    072C            RB     ADD       REGN DATA  RREG       ADD                     WRITE FIRST WORD; RREG:=S
7185    072D         RC SM     SUB                  RREG RC   CAD       CTR   TICB      ; CTR:=IF (ADDR+1) > SM AND S >= (ADDR+1)
7187                                                                                       THEN (S - (ADDR+1)) ELSE CODE FOR NOP
7188    072E            RA     ADD       REGN DATA             ADD                 NEXT  WRITE SECOND WORD; NEXT
```

```
RECORD   C.S.          ************ ALU A ************  ************* ALU B **************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7191           ********************************************************************
7192           *    INCM; Increment memory, direct cases                          *
7193           ********************************************************************
7194           *
7195           $LUT INSTR=INCM(DB+):1 010 000 0XX XX,DSPL=8,DB,ENTRY=IMD
7196           $LUT INSTR=INCM(DB+):1 010 100 0XX XX,DSPL=8,DB,X,ENTRY=IMD
7197           $LUT INSTR=INCM(Q+):1 010 000 10X XX,DSPL=7,Q,ENTRY=IMD
7198           $LUT INSTR=INCM(Q+):1 010 100 10X XX,DSPL=7,Q,X,ENTRY=IMD
7199           $LUT INSTR=INCM(Q-):1 010 000 110 XX,DSPL=6,Q,F2,ENTRY=IMD
7200           $LUT INSTR=INCM(Q-):1 010 100 110 XX,DSPL=6,Q,F2,X,ENTRY=IMD
7201           $LUT INSTR=INCM(S-):1 010 000 111 XX,DSPL=6,SM,F2,ENTRY=IMSM
7202           $LUT INSTR=INCM(S-):1 010 100 111 XX,DSPL=6,SM,F2,X,ENTRY=IMSM
7203           *
7204    072F   IMSM SR   UBA  ADD                              ADD                      UBA:=SR + (SM + XC);
7206    0730   IMD  UBA  DSPL ADSB       RH   ROD    SR   SM   ADD                      RH:=UBA:=(BASE + XC) +/- DSPL, READ; UBB:=S
7208    0731   IMD1 UBA  SM   CAD                    UBB  UBA  BNDE      CTR  TICB     ; BOUNDS CHECK S>=ADDR;
7210                                                                                    CTR:=IF ADDR>S AND S>=ADDR AND
7211                                                                                    NOT (FSS AND (DB REL ADDR OR TFF)) THEN
7212                                                                                    (S-ADDR) ELSE CODE FOR OPERAND AND NOP
7213    0732                 INC                    RH   DL   BNDE                     UBA:=1; BOUNDS CHECK ADDR >= DL
7215    0733   IMD2 UBA  REGN ADD            CCO                ADD                      UBA:=UBA + MEMORY LOCATION, CCO;
7217    0734              UBA  ADD       REGN DATA              ADD                NEXT  WRITE MEMORY LOCATION OR STORE IN TOS; NEXT
7219           *
7220           ********************************************************************
7221           *    INCM; Increment memory, indirect cases                        *
7222           ********************************************************************
7223           *
7224           $LUT INSTR=INCM(DB+):1 010 010 0XX XX,DSPL=8,DB,INDR,ENTRY=IMID
7225           $LUT INSTR=INCM(DB+):1 010 110 0XX XX,DSPL=8,DB,X,INDR,ENTRY=IMID
7226           $LUT INSTR=INCM(Q+):1 010 010 10X XX,DSPL=7,Q,INDR,ENTRY=IMID
7227           $LUT INSTR=INCM(Q+):1 010 110 10X XX,DSPL=7,Q,X,INDR,ENTRY=IMID
7228           $LUT INSTR=INCM(Q-):1 010 010 110 XX,DSPL=6,Q,INDR,F2,ENTRY=IMID
7229           $LUT INSTR=INCM(Q-):1 010 110 110 XX,DSPL=6,Q,X,INDR,F2,ENTRY=IMID
7230           $LUT INSTR=INCM(S-):1 010 010 111 XX,DSPL=6,SM,INDR,F2,ENTRY=IMIS
7231           $LUT INSTR=INCM(S-):1 010 110 111 XX,DSPL=6,SM,X,INDR,F2,ENTRY=IMIS
7232           *
7234    0735   IMIS SR   SM   ADD                              ADD                      UBA:=SR + SM;
7235    0736   IMID UBA  DSPL ADSB       RH   ROD    SR   SM   ADD       SP3B           RH:=UBA:=BASE +/- DSPL, READ; SP3B:=UBB:=S
7237    0737        UBA  SM   CAD                    UBB  UBA  BNDE      CTR  TICB     ; BOUNDS CHECK S>=ADDR; CTR:=IF ADDR>SM AND
7239                                                                                    S>=ADDR AND NOT (FSS AND DB REL ADDR)
7240                                                                                    THEN (S-ADDR) ELSE CODE FOR OPA AND NOP
7241    0738        XC   DB   ADD                    RH   DL   BNDE           STFF     UBA:=XC + DB; BOUNDS CHECK ADDR >= DL, STFF
7243    0739        UBA  REGN ADD       RH   ROD    SP3B      JSB  IMD1          UNC   RH:=UBA:=TARGET ADDR, READ; UBB:=S, JSB
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7246            ************************************************************************
7247            *    DECM; Decrement memory, direct cases                              *
7248            ************************************************************************
7249            *
7250            $LUT INSTR=DECM(DB+):1 010 001 0XX XX,DSPL=8,DB,ENTRY=DMD
7251            $LUT INSTR=DECM(DB+):1 010 101 0XX XX,DSPL=8,DB,X,ENTRY=DMD
7252            $LUT INSTR=DECM(Q+):1 010 001 10X XX,DSPL=7,Q,ENTRY=DMD
7253            $LUT INSTR=DECM(Q+):1 010 101 10X XX,DSPL=7,Q,X,ENTRY=DMD
7254            $LUT INSTR=DECM(Q-):1 010 001 110 XX,DSPL=6,Q,F2,ENTRY=DMD
7255            $LUT INSTR=DECM(Q-):1 010 101 110 XX,DSPL=6,Q,F2,X,ENTRY=DMD
7256            $LUT INSTR=DECM(S-):1 010 001 111 XX,DSPL=6,SM,F2,ENTRY=DMS
7257            $LUT INSTR=DECM(S-):1 010 101 111 XX,DSPL=6,SM,F2,X,ENTRY=DMS
7258            *
7259    073A    DMS  SR   UBA  ADD                          ADD                        UBA:=SR + (SM + XC);
7261    073B    DMD  UBA  DSPL ADSB       RH   ROD   SR   SM   ADD                   RH:=UBA:=(BASE + XC) +/- DSPL, READ; UBB:=S
7263    073C    DMD1 UBA  SM   CAD                    UBB  UBA  BNDE       CTR  TICB  ; BOUNDS CHECK S>=ADDR,
7265                                                                                    CTR:=IF ADDR > S AND S >= ADDR AND
7266                                                                                    NOT (FSS AND (DB REL ADDR OR TFF)) THEN
7267                                                                                    (S-ADDR) ELSE CODE FOR OPERAND AND NOP
7268    073D                    JSBC IMD2      UNC   RH   DL   BNDE                   UBA:=-1, JSB; BOUNDS CHECK ADDR >= DL
7270            *
7271            ************************************************************************
7272            *    DECM; Decrement memory, indirect cases                            *
7273            ************************************************************************
7274            *
7275            $LUT INSTR=DECM(DB+):1 010 011 0XX XX,DSPL=8,DB,INDR,ENTRY=DMID
7276            $LUT INSTR=DECM(DB+):1 010 111 0XX XX,DSPL=8,DB,X,INDR,ENTRY=DMID
7277            $LUT INSTR=DECM(Q+):1 010 011 10X XX,DSPL=7,Q,INDR,ENTRY=DMID
7278            $LUT INSTR=DECM(Q+):1 010 111 10X XX,DSPL=7,Q,X,INDR,ENTRY=DMID
7279            $LUT INSTR=DECM(Q-):1 010 011 110 XX,DSPL=6,Q,INDR,F2,ENTRY=DMID
7280            $LUT INSTR=DECM(Q-):1 010 111 110 XX,DSPL=6,Q,X,INDR,F2,ENTRY=DMID
7281            $LUT INSTR=DECM(S-):1 010 011 111 XX,DSPL=6,SM,INDR,F2,ENTRY=DMIS
7282            $LUT INSTR=DECM(S-):1 010 111 111 XX,DSPL=6,SM,X,INDR,F2,ENTRY=DMIS
7283            *
7284    073E    DMIS SR   SM   ADD                          ADD                        UBA:=SR + SM;
7286    073F    DMID UBA  DSPL ADSB       RH   ROD   SR   SM   ADD        SP3B       RH:=UBA:=BASE +/- DSPL, READ; SP3B:=UBB:=S
7288    0740         UBA  SM   CAD                    UBB  UBA  BNDE       CTR  TICB  ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
7290                                                                                    S>=ADDR AND NOT (FSS AND DB REL ADDR)
7291                                                                                    THEN (S-ADDR) ELSE CODE FOR OPA AND NOP
7292    0741         XC   DB   ADD                    RH   DL   BNDE            STFF   UBA:=XC + DB; BOUNDS CHECK ADDR >= DL, STFF
7294    0742         UBA  REGN ADD       RH   ROD   SP3B      JSB  DMD1             UNC  RH:=UBA:=TARGET ADDR, READ; UBB:=S, JSB
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7297            ********************************************************************
7298            *    ADDM; Add memory to TOS, DB, Q, and S relative                *
7299            ********************************************************************
7300            *
7301            $LUT INSTR=ADDM(DB+):0 111 001 0XX XX,SR=1,DSPL=8,DB,ENTRY=AMD
7302            $LUT INSTR=ADDM(DB+):0 111 101 0XX XX,SR=1,DSPL=8,DB,X,ENTRY=AMD
7303            $LUT INSTR=ADDM(Q+):0 111 001 10X XX,SR=1,DSPL=7,Q,ENTRY=AMD
7304            $LUT INSTR=ADDM(Q+):0 111 101 10X XX,SR=1,DSPL=7,Q,X,ENTRY=AMD
7305            $LUT INSTR=ADDM(Q-):0 111 001 110 XX,SR=1,DSPL=6,Q,F2,ENTRY=AMD
7306            $LUT INSTR=ADDM(Q-):0 111 101 110 XX,SR=1,DSPL=6,Q,F2,X,ENTRY=AMD
7307            $LUT INSTR=ADDM(S-):0 111 001 111 XX,SR=1,DSPL=6,SM,F2,ENTRY=AMS
7308            $LUT INSTR=ADDM(S-):0 111 101 111 XX,SR=1,DSPL=6,SM,F2,X,ENTRY=AMS
7309            *
7310    0743    AMS  SR   UBA  ADD                          ADD                      UBA:=SR + (SM + XC);
7312    0744    AMD  UBA  DSPL ADSB      RH   ROD   SR   SM   ADD           CF2    RH:=UBA:=(BASE + XC) +/- DSPL, READ;
7314                                                                                UBB:=S, CF2 FOR ADDITION
7315    0745    AMD1 UBA  SM   CAD                    UBB  UBA  BNDE      CTR  TICB   ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM
7317                                                                                  AND S>=ADDR AND NOT (FSS AND DB REL ADDR)
7318                                                                                  THEN (S-ADDR) ELSE CODE FOR OPA AND NOP
7319    0746    ASMN RH   DL   BNDE                         ADD                      BOUNDS CHECK ADDR >= DL;
7321    0747         RA   REGN ADSB      RA   CCO                JSZ  NEXT          UNC  (S):=(S) +/- (ADDR), CCO; NEXT
7323            *
7324            ********************************************************************
7325            *    ADDM; Add memory to TOS, Indirect DB, Q, and S relative       *
7326            ********************************************************************
7327            *
7328            $LUT INSTR=ADDM(DB+):0111 0110 XXXX,SR=1,DSPL=8,DB,INDR,ENTRY=AMID
7329            $LUT INSTR=ADDM(DB+):0111 1110 XXXX,SR=1,DSPL=8,DB,X,INDR,ENTRY=AMID
7330            $LUT INSTR=ADDM(Q+):0111 0111 0XXX,SR=1,DSPL=7,Q,INDR,ENTRY=AMID
7331            $LUT INSTR=ADDM(Q+):0111 1111 0XXX,SR=1,DSPL=7,Q,X,INDR,ENTRY=AMID
7332            $LUT INSTR=ADDM(Q-):0111 0111 10XX,SR=1,DSPL=6,Q,INDR,F2,ENTRY=AMID
7333            $LUT INSTR=ADDM(Q-):0111 1111 10XX,SR=1,DSPL=6,Q,X,INDR,F2,ENTRY=AMID
7334            $LUT INSTR=ADDM(S-):0111 0111 11XX,SR=1,DSPL=6,SM,INDR,F2,ENTRY=AMIS
7335            $LUT INSTR=ADDM(S-):011111111XX,SR=1,DSPL=6,SM,X,INDR,F2,ENTRY=AMIS
7336            *
7337    0748    AMIS SR   SM   ADD                          ADD                      UBA:=SR + SM;
7339    0749    AMID UBA  DSPL ADSB      RH   ROD   SR   SM   ADD           CF2    RH:=UBA:=BASE +/- DSPL, READ;
7341                                                                                SP3B:=UBB:=S, CF2 FOR ADDITION
7342    074A         UBA  SM   CAD                    UBB  UBA  BNDE      CTR  TICB   ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
7344                                                                                  S>=ADDR AND NOT (FSS AND DB REL ADDR)
7345                                                                                  THEN (S-ADDR) ELSE CODE FOR OPA AND NOP
7346    074B         XC   DB   ADD                        RH   DL   BNDE           STFF  UBA:=XC + DB; BOUNDS CHECK ADDR >= DL, STFF
7348    074C         UBA  REGN ADD       RH   ROD   SP3B       JSB  AMD1          UNC  RH:=UBA:=TARGET ADDR, READ; UBB:=S, JSB
```

```
RECORD  C.S.          ********** ALU A **********  ************* ALU B **************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7351          ***************************************************************
7352          *    ADDM; Add memory to TOS, P relative                        *
7353          ***************************************************************
7354          *
7355          $LUT INSTR=ADDM(P+):0 111 000 0XX XX,SR=1,DSPL=8,P,ENTRY=AMP
7356          $LUT INSTR=ADDM(P+):0 111 100 0XX XX,SR=1,DSPL=8,X,P,ENTRY=AMP
7357          $LUT INSTR=ADDM(P-):0 111 000 1XX XX,SR=1,DSPL=8,P,F2,ENTRY=AMP
7358          $LUT INSTR=ADDM(P-):0 111 100 1XX XX,SR=1,DSPL=8,X,F2,P,ENTRY=AMP
7359          *
7360    074D  AMP   UBA  DSPL ADSB      RH   ROP             ADD                    RH:=UBA:=(P + XC) +/- DSPL, READ;
7362    074E  AMP1  UBA  PB   UBNE                    PL   UBA  UBNE                 BOUNDS CHECK ADDR >= PB, CHECK PL >= ADDR
7364    074F        RA   OPA  ADD       RA   CCO             ADD                NEXT  (S):=(S) + MEMORY LOCATION, CCO, NEXT
7366          *
7367          ***************************************************************
7368          *    ADDM; Add memory to TOS, Indirect P relative               *
7369          ***************************************************************
7370          *
7371          $LUT INSTR=ADDM(P+):0111 0100 XXXX,SR=1,DSPL=8,INDR,P,ENTRY=AMIP
7372          $LUT INSTR=ADDM(P+):0111 1100 XXXX,SR=1,DSPL=8,X,INDR,P,ENTRY=AMIP
7373          $LUT INSTR=ADDM(P-):0111 0101 XXXX,SR=1,DSPL=8,INDR,F2,P,ENTRY=AMIP
7374          $LUT INSTR=ADDM(P-):0111 1101 XXXX,SR=1,DSPL=8,X,INDR,F2,P,ENTRY=AMIP
7375          *
7376    0750  AMIP UBA  DSPL ADSB      RH   ROP             ADD                    RH:=UBA:=P +/- DSPL, READ;
7378    0751        XC   UBA  ADD                   UBA  PB   UBNE                 UBA:=XC + INDIRECT CELL ADDR;
7380                                                                               BOUNDS CHECK INDIRECT CELL ADDR >= PB
7381    0752        UBA  OPA  JSB  AMP1      ROP   PL   RH   UBNE                 READ XC + CELL ADDR + CELL CONTENTS, JSB
7383                                                                               BOUNDS CHECK PL>= INDIRECT CELL ADDR
7384          *
7385          ***************************************************************
7386          *    SUBM; Subtract memory from TOS, DB, Q, and S relative      *
7387          ***************************************************************
7388          *
7389          $LUT INSTR=SUBM(DB+):1 000 001 0XX XX,SR=1,DSPL=8,DB,ENTRY=SMD
7390          $LUT INSTR=SUBM(DB+):1 000 101 0XX XX,SR=1,DSPL=8,DB,X,ENTRY=SMD
7391          $LUT INSTR=SUBM(Q+):1 000 001 10X XX,SR=1,DSPL=7,Q,ENTRY=SMD
7392          $LUT INSTR=SUBM(Q+):1 000 101 10X XX,SR=1,DSPL=7,Q,X,ENTRY=SMD
7393          $LUT INSTR=SUBM(Q-):1 000 001 110 XX,SR=1,DSPL=6,Q,F2,ENTRY=SMD
7394          $LUT INSTR=SUBM(Q-):1 000 101 110 XX,SR=1,DSPL=6,Q,F2,X,ENTRY=SMD
7395          $LUT INSTR=SUBM(S-):1 000 001 111 XX,SR=1,DSPL=6,SM,F2,ENTRY=SMS
7396          $LUT INSTR=SUBM(S-):1 000 101 111 XX,SR=1,DSPL=6,SM,F2,X,ENTRY=SMS
7397          *
7398    0753  SMS  SR   UBA  ADD                             ADD                    UBA:=SR + (SM + XC);
7400    0754  SMD  UBA  DSPL ADSB      RH   ROD   SR   SM   ADD            SF2     RH:=UBA:=(BASE + XC) +/- DSPL, READ;
7402                                                                               UBB:=S, SF2 FOR SUBTRACTION
7403    0755  SMD1 UBA  SM   JSBC ASMN      UNC   UBB  UBA  BNDE      CTR  TICB    JSB, BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM
7405                                                                                 AND S>=ADDR AND NOT (FSS AND DB REL ADDR)
7406                                                                                 THEN (S-ADDR) ELSE CODE FOR OPA AND NOP
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7408           ***************************************************************
7409           *    SUBM; Subtract memory from TOS, Indirect DB, Q, and S relative *
7410           ***************************************************************
7411           *
7412           $LUT INSTR=SUBM(DB+):1000 0110 XXXX,SR=1,DSPL=8,DB,INDR,ENTRY=SMID
7413           $LUT INSTR=SUBM(DB+):1000 1110 XXXX,SR=1,DSPL=8,DB,X,INDR,ENTRY=SMID
7414           $LUT INSTR=SUBM(Q+):1000 0111 0XXX,SR=1,DSPL=7,Q,INDR,ENTRY=SMID
7415           $LUT INSTR=SUBM(Q+):1000 1111 0XXX,SR=1,DSPL=7,Q,X,INDR,ENTRY=SMID
7416           $LUT INSTR=SUBM(Q-):1000 0111 10XX,SR=1,DSPL=6,Q,INDR,F2,ENTRY=SMID
7417           $LUT INSTR=SUBM(Q-):1000 1111 10XX,SR=1,DSPL=6,Q,X,INDR,F2,ENTRY=SMID
7418           $LUT INSTR=SUBM(S-):1000 0111 11XX,SR=1,DSPL=6,SM,INDR,F2,ENTRY=SMIS
7419           $LUT INSTR=SUBM(S-):1000111111XX,SR=1,DSPL=6,SM,X,INDR,F2,ENTRY=SMIS
7420           *
7421    0756   SMIS SR   SM   ADD                          ADD                        UBA:=SR ; SM;
7423    0757   SMID UBA  DSPL ADSB      RH   ROD    SR   SM   ADD       SP3B SF2     RH:=UBA:=BASE +/- DSPL, READ;
7425                                                                                  SP3A:=UBB:=S, SF2 FOR SUBTRACTION
7426    0758        UBA  SM   CAD                    UBB  UBA  BNDE      CTR  TICB    ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
7428                                                                                    S>=ADDR AND NOT (FSS AND DB REL ADDR)
7429                                                                                    THEN (S-ADDR) ELSE CODE FOR OPA AND NOP
7430    0759        XC   DB   ADD                    RH   DL   BNDE           STFF    UBA:-XC + DB; BOUNDS CHECK ADDR >= DL, STFF
7432    075A        UBA  REGN ADD      RH   ROD    SP3B      JSB  SMD1           UNC  RH:=UBA:=TARGET ADDR, READ; UBB:=S, JSB
7434           *
7435           ***************************************************************
7436           *    SUBM; Subtract memory from TOS, P relative                 *
7437           ***************************************************************
7438           *
7439           $LUT INSTR=SUBM(P+):1 000 000 0XX XX,SR=1,DSPL=8,P,ENTRY=SMP
7440           $LUT INSTR=SUBM(P+):1 000 100 0XX XX,SR=1,DSPL=8,X,P,ENTRY=SMP
7441           $LUT INSTR=SUBM(P-):1 000 000 1XX XX,SR=1,DSPL=8,P,F2,ENTRY=SMP
7442           $LUT INSTR=SUBM(P-):1 000 100 1XX XX,SR=1,DSPL=8,X,F2,P,ENTRY=SMP
7443           *
7444    075B   SMP  UBA  DSPL ADSB      RH   ROP          ADD                        RH:=UBA:=(P + XC) +/- DSPL, READ;
7446    075C   SMP1 UBA  PB   UBNE                  PL   UBA  UBNE                  BOUNDS CHECK ADDR >= PB; CHECK PL >= ADDR
7448    075D        RA   OPA  SUB      RA   CCO          ADD                  NEXT   (S):=(S) - MEMORY LOCATION, CCO; NEXT
7450           *
7451           ***************************************************************
7452           *    SUBM; Subtract memory from TOS, Indirect P relative        *
7453           ***************************************************************
7454           *
7455           $LUT INSTR=SUBM(P+):1000 0100 XXXX,SR=1,DSPL=8,INDR,P,ENTRY=SMIP
7456           $LUT INSTR=SUBM(P+):1000 1100 XXXX,SR=1,DSPL=8,X,INDR,P,ENTRY=SMIP
7457           $LUT INSTR=SUBM(P-):1000 0101 XXXX,SR=1,DSPL=8,INDR,F2,P,ENTRY=SMIP
7458           $LUT INSTR=SUBM(P-):1000 1101 XXXX,SR=1,DSPL=8,X,INDR,F2,P,ENTRY=SMIP
7459           *
7460    075E   SMIP UBA  DSPL ADSB      RH   ROP          ADD                        RH:=UBA:=P +/- DSPL, READ;
7462    075F        XC   UBA  ADD                    UBA  PB   UBNE                  UBA:=XC + INDIRECT CELL ADDR;
7464                                                                                  BOUNDS CHECK INDIRECT CELL ADDR >= PB
7465    0760        UBA  OPA  JSB  SMP1      ROP    PL   RH   UBNE                  READ XC + CELL ADDR + CELL CONTENTS, JSB
7467                                                                                  BOUNDS CHECK PL>= INDIRECT CELL ADDR
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7469            ************************************************************************
7470            *     MPYM; Multiply TOS by memory, DB, Q, and S relative              *
7471            ************************************************************************
7472            *
7473            $LUT INSTR=MPYM(DB+):1 001 001 0XX XX,SR=1,DSPL=8,DB,ENTRY=MMDQ
7474            $LUT INSTR=MPYM(DB+):1 001 101 0XX XX,SR=1,DSPL=8,DB,X,ENTRY=MMDQ
7475            $LUT INSTR=MPYM(Q+):1 001 001 10X XX,SR=1,DSPL=7,Q,ENTRY=MMDQ
7476            $LUT INSTR=MPYM(Q+):1 001 101 10X XX,SR=1,DSPL=7,Q,X,ENTRY=MMDQ
7477            $LUT INSTR=MPYM(Q-):1 001 001 110 XX,SR=1,DSPL=6,Q,F2,ENTRY=MMDQ
7478            $LUT INSTR=MPYM(Q-):1 001 101 110 XX,SR=1,DSPL=6,Q,F2,X,ENTRY=MMDQ
7479            $LUT INSTR=MPYM(S-):1 001 001 111 XX,SR=1,DSPL=6,SM,F2,ENTRY=MPMS
7480            $LUT INSTR=MPYM(S-):1 001 101 111 XX,SR=1,DSPL=6,SM,F2,X,ENTRY=MPMS
7481            *
7482    0761    MPMS SR   SM   ADD                            ADD                      UBA:=SR + SM
7484    0762    MMDQ UBA  DSPL ADSB      RH   ROD   SR   SM   ADD       SP3B           RH:=UBA:=BASE+DSPL; SP3B:=UBB:=SR+SM
7486    0763    MMD1 UBA  SM   CAD                 UBB  UBA  BNDE      CTR  TICB      ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
7488                                                                                    S>=ADDR AND NOT (FSS AND DB REL ADDR)
7489                                                                                    THEN (S-ADDR) ELSE CODE FOR OPERAND
7490    0764         RH   DL   BNDE                           JSB  MMP2          UNC   BOUNDS CHECK ADDR >= DL; JSB
7492            *
7493            ************************************************************************
7494            *     MPYM; Multiply TOS by memory, indirect DB, Q, and S relative     *
7495            ************************************************************************
7496            *
7497            $LUT INSTR=MPYM(DB+):1001 0110 XXXX,SR=1,DSPL=8,DB,INDR,ENTRY=MMID
7498            $LUT INSTR=MPYM(DB+):1001 1110 XXXX,SR=1,DSPL=8,DB,INDR,X,ENTRY=MMID
7499            $LUT INSTR=MPYM(Q+):1001 0111 0XXX,SR=1,DSPL=7,Q,INDR,ENTRY=MMID
7500            $LUT INSTR=MPYM(Q+):1001 1111 0XXX,SR=1,DSPL=7,Q,INDR,X,ENTRY=MMID
7501            $LUT INSTR=MPYM(Q-):1001 0111 10XX,SR=1,DSPL=6,Q,INDR,F2,ENTRY=MMID
7502            $LUT INSTR=MPYM(Q-):1001 1111 10XX,SR=1,DSPL=6,Q,INDR,F2,X,ENTRY=MMID
7503            $LUT INSTR=MPYM(S-):1001 0111 11XX,SR=1,DSPL=6,SM,INDR,F2,ENTRY=MMIS
7504            $LUT INSTR=MPYM(S-):1001111111XX,SR=1,DSPL=6,SM,INDR,F2,X,ENTRY=MMIS
7505            *
7506    0765    MMIS SR   UBA  ADD                            ADD                      UBA:=SM + SR;
7508    0766    MMID UBA  DSPL ADSB      RH   ROD   SR   SM   ADD       SP3B           RH:=UBA:=BASE + DSPL, READ;
7510                                                                                    SP3B:=UBB:=SR + SM
7511    0767         UBA  SM   CAD                 UBB  UBA  BNDE      CTR  TICB      ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
7513                                                                                    S>=ADDR AND NOT (FSS AND DB REL ADDR)
7514                                                                                    THEN (S-ADDR) ELSE CODE FOR OPERAND
7515    0768         XC   DB   ADD                            RH   DL   BNDE           STFF  UBA:=XC+DB; BOUNDS CHECK ADDR >= DL, STFF
7517    0769         UBA  REGN ADD       RH   ROD   SP3B      JSB  MMD1          UNC   RH:=UBA:=TARGET ADDR, READ; UBB:=S, JSB
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7520            ;*****************************************************************
7521            ;    MPYM, Multiply TOS by memory, P relative                     *
7522            ;*****************************************************************
7523            ;
7524            $LUT INSTR=MPYM(P+) 1 001 000 0XX XX,SR=1,DSPL=8,P,ENTRY=MMP
7525            $LUT INSTR=MPYM(P+) 1 001 100 0XX XX,SR=1,DSPL=8,P,X,ENTRY=MMP
7526            $LUT INSTR=MPYM(P-) 1 001 000 1XX XX,SR=1,DSPL=8,P,F2,ENTRY=MMP
7527            $LUT INSTR=MPYM(P-) 1 001 100 1XX XX,SR=1,DSPL=8,P,F2,X,ENTRY=MMP
7528            ;
7529    076A    MMP   UBA   DSPL ADSB       RH    ROP    000F       ADDL       CTR                RH:=UBA:=(P + XC) +/- DSPL, READ,
7531                                                                                              CTR :=REGN CODE FOR OPERAND
7532    076B    MMP1  UBA   PB   UBNE                    PL    UBA   UBNE                       BOUNDS CHECK ADDR >= PB, CHECK PL >= ADDR
7534    076C    MMP2        REGN ADD        RH                      ADD                        RH :=U
7536    076D                UBA  ADD        SP4A HBF2               REPN              CLO  000F  SP4A :=U, F2:=SIGN(U); REPN 16 TIMES, CLO
7538    076E                     MPAD                    RA    UBB  LINK              DCTR CTRQ  , MPAD UNTIL CTR IS DECREMENTED TO 0
7540    076F          RA         XFRR            HBF2           UBB  JSL   MPY1             NF1   F2 :=SIGN(V) (SREG IS OVERWRITTEN BY MPAD),
7542                                                                                              UBB:=MSW, MEDIUM JSB TO USE FOLLOWING UBB
7543    0770                RH   ADD                     UBB   RA   XSUB                         UBA:=U
7545                                                                                              UBB:=IF U < 0 THEN MSW - (V*2**16) ELSE MSW
7546            ;
7547            ;*****************************************************************
7548            ;    MPYM, Multiply TOS by memory, indirect, P relative           *
7549            ;*****************************************************************
7550            ;
7551            $LUT INSTR=MPYM(P+),1001 0100 XXXX,SR=1,DSPL=8,P,INDR,ENTRY=MMIP
7552            $LUT INSTR=MPYM(P+),1001 1100 XXXX,SR=1,DSPL=8,P,X,INDR,ENTRY=MMIP
7553            $LUT INSTR=MPYM(P-),1001 0101 XXXX,SR=1,DSPL=8,P,F2,INDR,ENTRY=MMIP
7554            $LUT INSTR=MPYM(P-),1001 1101 XXXX,SR=1,DSPL=8,P,F2,X,INDR,ENTRY=MMIP
7555            ;
7556    0771    MMIP  UBA   DSPL ADSB       RH    ROP                ADD                        RH :=UBA :=P +/- DSPL, READ,
7558    0772          XC    UBA  ADD                       UBA   PB   UBNE                       UBA :=XC + INDIRECT CELL ADDR,
7560                                                                                              BOUNDS CHECK INDIRECT CELL ADDR >= PB
7561    0773          UBA   OPA  JSB  MMP1      ROP    PL    RH   UBNE                       READ XC + CELL ADDR + CELL CONTENTS, JSB
7563                                                                                              BOUNDS CHECK PL >= INDIRECT CELL ADDR
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7565            ************************************************************************
7566            *    CMPM; Compare TOS with memory, DB, Q, and S relative              *
7567            ************************************************************************
7568            *
7569            $LUT INSTR=CMPM(DB+):0 110 001 0XX XX,SR=1,DSPL=8,DB,ENTRY=CMDM
7570            $LUT INSTR=CMPM(DB+):0 110 101 0XX XX,SR=1,DSPL=8,DB,X,ENTRY=CMDM
7571            $LUT INSTR=CMPM(Q+):0 110 001 10X XX,SR=1,DSPL=7,Q,ENTRY=CMDM
7572            $LUT INSTR=CMPM(Q+):0 110 101 10X XX,SR=1,DSPL=7,Q,X,ENTRY=CMDM
7573            $LUT INSTR=CMPM(Q-):0 110 001 110 XX,SR=1,DSPL=6,Q,F2,ENTRY=CMDM
7574            $LUT INSTR=CMPM(Q-):0 110 101 110 XX,SR=1,DSPL=6,Q,F2,X,ENTRY=CMDM
7575            $LUT INSTR=CMPM(S-):0 110 001 111 XX,SR=1,DSPL=6,SM,F2,ENTRY=CMS
7576            $LUT INSTR=CMPM(S-):0 110 101 111 XX,SR=1,DSPL=6,SM,F2,X,ENTRY=CMS
7577            *
7578    0774    CMS   SR   UBA  ADD                        ADD                        UBA:=SR + (SM + XC);
7580    0775    CMDM UBA   DSPL ADSB      RH   ROD   SR   SM   ADD                  RH:=UBA:=(BASE + XC) +/- DSPL, READ; UBB:=S
7582    0776    CMD2 UBA   SM   CAD                  UBB  UBA  BNDE       CTR  TICB  ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
7584                                                                                    S>=ADDR AND NOT (FSS AND (DB REL ADDR OR
7585                                                                                    TFF)) THEN (S-ADDR) ELSE CODE FOR OPERAND
7586    0777                 ADD                      RH   DL   BNDE            EPOP  ; BOUNDS CHECK ADDR >= DL,POP TOS
7588    0778         RA    REGN SUB            NOFL              INC                  SKIP IF NO OVERFLOW ON (S) - MEMORY; UBB:=1
7590    0779                 ADD                 NEXT  RA   UBB  IOR            CCA   NEXT IF OVERFLOW;
7592                                                                                    SET CCL ON (S), UBB CANNOT EQUAL ZERO
7593    077A         RA    REGN SUB            CCA               ADD                NEXT SET CCA ON ((S) - MEMORY) IF NOFL; NEXT
7595            *
7596            ************************************************************************
7597            *    CMPM; Compare TOS with memory, Indirect DB, Q, and S relative     *
7598            ************************************************************************
7599            *
7600            $LUT INSTR=CMPM(DB+):0110 0110 XXXX,SR=1,DSPL=8,DB,INDR,ENTRY=CMID
7601            $LUT INSTR=CMPM(DB+):0110 1110 XXXX,SR=1,DSPL=8,DB,X,INDR,ENTRY=CMID
7602            $LUT INSTR=CMPM(Q+):0110 0111 0XXX,SR=1,DSPL=7,Q,INDR,ENTRY=CMID
7603            $LUT INSTR=CMPM(Q+):0110 1111 0XXX,SR=1,DSPL=7,Q,X,INDR,ENTRY=CMID
7604            $LUT INSTR=CMPM(Q-):0110 0111 10XX,SR=1,DSPL=6,Q,INDR,F2,ENTRY=CMID
7605            $LUT INSTR=CMPM(Q-):0110 1111 10XX,SR=1,DSPL=6,Q,X,INDR,F2,ENTRY=CMID
7606            $LUT INSTR=CMPM(S-):0110 0111 11XX,SR=1,DSPL=6,SM,INDR,F2,ENTRY=CMIS
7607            $LUT INSTR=CMPM(S-):0110111111XX,SR=1,DSPL=6,SM,X,INDR,F2,ENTRY=CMIS
7608            *
7609    077B    CMIS SR   SM   ADD                        ADD                        UBA:=SR + SM;
7611    077C    CMID UBA  DSPL ADSB       RH   ROD   SR   SM   ADD        SP3B      RH:=UBA:=BASE +/- DSPL, READ; SP3B:=UBB:=S
7613    077D         UBA  SM   CAD                  UBB  UBA  BNDE       CTR  TICB  ; BOUNDS CHECK S>=ADDR, CTR:=IF ADDR>SM AND
7615                                                                                    S>=ADDR AND NOT (FSS AND DB REL ADDR)
7616                                                                                    THEN (S-ADDR) ELSE CODE FOR OPA AND NOP
7617    077E         XC   DB   ADD                  RH   DL   BNDE            STFF  UBA:=XC + DB; BOUNDS CHECK ADDR >= DL, STFF
7619    C77F         UBA  REGN ADD        RH   ROD   SP3B      JSB  CMD2          UNC  RH:=UBA:=TARGET ADDR, READ; UBB:=S, JSB
```

```
RECORD  C S          ********** ALU A **********  ************** ALU B **************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7622            ***********************************************************************
7623            *     CMPM; Compare TOS with memory, P relative                       *
7624            ***********************************************************************
7625            *
7626            $LUT INSTR=CMPM(P+):0 110 000 0XX XX,SR=1,DSPL=8,P,ENTRY=CMPM
7627            $LUT INSTR=CMPM(P+):0 110 100 0XX XX,SR=1,DSPL=8,X,P,ENTRY=CMPM
7628            $LUT INSTR=CMPM(P-):0 110 000 1XX XX,SR=1,DSPL=8,P,F2,ENTRY=CMPM
7629            $LUT INSTR=CMPM(P-):0 110 100 1XX XX,SR=1,DSPL=8,X,F2,P,ENTRY=CMPM
7630            *
7631    0780    CMPM UBA  DSPL ADSB      RH   ROP            ADD                      RH:=UBA:=(P + XC) +/- DSPL, READ;
7633    0781    CMP1 UBA  PB   UBNE                  PL   UBA  UBNE          EPOP      BOUNDS CHECK ADDR >= PB; CHECK PL >= ADDR
7635                                                                                  POP OFF TOP OF STACK
7636    0782         RA   OPA  SUB             NOFL           INC                      SKIP IF NO OVERFLOW ON (S) - MEMORY; UBB:=1
7638    0783                   ADD             NEXT RA   UBB  IOR          CCA       NEXT IF OVERFLOW;
7640                                                                                  SET CCL ON (S), UBB CANNOT EQUAL ZERO
7641    0784         RA   OPA  SUB             CCA            ADD              NEXT  SET CCA ON ((S) - MEMORY) IF NOFL; NEXT
7643            *
7644            ***********************************************************************
7645            *     CMPM; Compare TOS with memory, Indirect P relative              *
7646            ***********************************************************************
7647            *
7648            $LUT INSTR=CMPM(P+):0110 0100 XXXX,SR=1,DSPL=8,INDR,P,ENTRY=CMIP
7649            $LUT INSTR=CMPM(P+):0110 1100 XXXX,SR=1,DSPL=8,X,INDR,P,ENTRY=CMIP
7650            $LUT INSTR=CMPM(P-):0110 0101 XXXX,SR=1,DSPL=8,INDR,F2,P,ENTRY=CMIP
7651            $LUT INSTR=CMPM(P-):0110 1101 XXXX,SR=1,DSPL=8,X,INDR,F2,P,ENTRY=CMIP
7652            *
7653    0785    CMIP UBA  DSPL ADSB      RH   ROP            ADD                      RH:=UBA:=P +/- DSPL, READ;
7655    0786         XC   UBA  ADD                  UBA  PB   UBNE                     UBA:=XC + INDIRECT CELL ADDR;
7657                                                                                  BOUNDS CHECK INDIRECT CELL ADDR >= PB
7658    0787         UBA  OPA  JSB  CMP1      ROP  PL   RH   UBNE                     READ XC + CELL ADDR + CELL CONTENTS, JSB
7660                                                                                  BOUNDS CHECK PL>= INDIRECT CELL ADDR
```

```
RECORD   C.S.        ********** ALU A **********  ************** ALU B **************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7662           ********************************************************************************
7663           *                                                                              *
7664           *   MLOG -- MICROCODE LOGGING instruction  (020104 000006)                     *
7665           *                                                                              *
7666           *           This instruction pushes four 32-bit counters onto                  *
7667           *           the stack and then zeros those counters.                           *
7668           *                                                                              *
7669           *           TOS-6   # of clocks spent in channel microcode (XR34)              *
7670           *           TOS-4   # of clocks spent in PAUSE instruction (XR35)              *
7671           *           TOS-2   # of clocks of 'normal execution'      (XR36)              *
7672           *           TOS     # of CSRQ's                            (XR37)              *
7673           *                                                                              *
7674           ********************************************************************************
7675           *
7676    0788   MLOG            JSZ  PSHA       SRNZ  00A2       ADDL      CTR               Empty TOS if SR <> 0; CTR := XR34 REGN code
7678    0789                SM  INC             WRS   00A6       ADDL      RA                Start writing data at SM + 1; RA := stopper
7680    078A                REGN ADD            DATA        REGN ADD             ICTR       Write msw of data; inc CTR
7682    078B                UBB  ADD            DATA  0002  SM   ADDL      SM                Write lsw of data; SM := SM + 2
7684    078C                     ADD                      RA   CTRS JSBS *-2          NZRO  ; Loop back if CTR was <> A5
7686    078D         00A5        ADDL       RA           00A2       ADDL      CTR           ; CTR := XR34 REGN code
7688    078E                     ADD        REGN                    ADD       REGN ICTR     Clear extended register pair
7690    078F                     ADD                     RA   CTRS JSBS *-1          NZRO  ; Loop back to clear another XR if CTR was
7692                                                                                          <> A5
7693    0790                     ADD                                 JSL  NEX1           UNC   ; Take a slow path to the next instruction
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7696           ********************************************************************
7697           *                                                                  *
7698           *    POWER ON:                                                     *
7699           *                                                                  *
7700           *              1)  Read (ZI+1)                                     *
7701           *              2)  Set RUNFF, clear PONINT*                        *
7702           *              3)  Ext. Plabel, Parameter = !A301 (%121401)        *
7703           *              4)  Initialize TCLK to 1                            *
7704           *              5)  Go to HALT if not running at Power Failure      *
7705           *              6)  Set up ICS                                      *
7706           *              7)  Trap to segment 1                               *
7707           *                                                                  *
7708           *    Entered with RA = !C606                                       *
7709           *                                                                  *
7710           ********************************************************************
7711           *
7712    0791   PON  2000       ADDL      XR12                JSL  CKSM BNKD      UNC   XR12=4000/2 ;BNKD=0                    (2324)
7714                                                                                     Init module map, do WCS & LUT checksum
7715    0792        RA         ADD  RRZ       ROD   0030      ADDL      CTR             Read abs(6); CTR := TCLK optn
7717    0793        SPOA       CSL       SPOA                INC       REGN            SPOA := %121401 (seg 1, STT %43); TCLK := 1
7719    0794             OPA   INC             ROBD      UBA  ADD       SP1B            Read (ZI+1); SP1B := %121401 (seg 1,STT %43)
7721    0795             XR12  ADD  LSL  XR12        RA        ADD  LLZ       CCPX     XR12=4000 ; CLEAR PONINT*, SET RUNFF
7723    0796                   JSZ  ICS             CF1       OPB  JSL  STOP        POS   Clear F1 for #IR1,
7725                                                                                       halt if not running, else set up ICS
7726    0797                   ADD       XR15                JSL  IR1  BNKS      UNC   Clear XR15; BNKS := 0, trap to segment 1
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7729           ***************************************************************************
7730           *                                                                         *
7731           *    POWER FAIL:                                                          *
7732           *                                                                         *
7733           *            1)  Go BUSY                                                  *
7734           *            2)  Send IOCL (I/O Clear) to all IMB's that are              *
7735           *                present                                                  *
7736           *            3)  Set PWFINH to disable further interrupts                 *
7737           *            4)  Set MSBit of ZI+1 to 1 if running                        *
7738           *                else set it to 0                                         *
7739           *            5)  Set parameter to 0                                       *
7740           *            6)  Set up ICS type interrupt trap to segment 1,             *
7741           *                STT %11                                                  *
7742           *                                                                         *
7743           *    Entered with UBA = !0C02, BKX3 = 0 and F4 set                        *
7744           *                                                                         *
7745           ***************************************************************************
7746           *
7747    0798   PFW0 0006      ADDL                    UBA       ADD           BUSC      UBA := @ZI; Go BUSY
7749    0799        UBA       ADD             ROX3    B000      ADDL    SP3B          Read ZI; SP3B := IOCL command
7751    079A        0200      ADDL      RH            0098      ANDL    SP2B          RH := RUNFF mask; SP2B := 0
7753    079B             OPA  INC             ROX3    RREG      ADD           CCPX    Read ZI+1; Clear MSGINT & PFWINT
7755    079C             SP2B ADD             CF1               JSL   L2PF BKX5  UNC   UBA := IMB#; BKX5 := 0; go see if it's there
7757    079D             UBA  JSB  *+2        NEG     0080 SP2B ADDL    SP2B          If this one not there jump over IOCL code,
7759                                                                                      SP2B := next IMB#
7760    079E                  ADD                                 JSL   IOMS    UNC   ; Send the IOCL message
7762    079F        0006      ADDL                    RH    SP2B JSBS  *-3     NZRO  UBA := CCPX mask; Loop back if not done
7764    07A0        RH   CPX2 AND             ZERO    UBA       ADD           CCPX    Skip if halted; set PWRFINH (no more ints)
7766    07A1             ADD  ADD             SF1     8901      ADDL    SP1B          Set F1 if running; SP1B := %104401 (4 #INT9)
7768    07A2             OPA  INC             F1HB              ADD                   If running set msb of ZI+1 else clear it,
7770                                                                                      set lsb of ZI+1 to indicate a powerfail
7771                                                                                      has occurred
7772    07A3        UBA       ADD             DATA              ADD           CF4B    Write ZI+1 back; Clear F4 for #INT9
7774    07A4             ADD  ADD       SPOA  SF2               JSZ   INT9    UNC   SPOA := 0; Do PFW trap to
7776                                                                                      segment 1, set F2 for #INT9
```

```
RECORD    C S                ********** ALU A **********  ********** ALU B **********
  NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7778            ***************************************************************
7779            *                                                             *
7780            *      FLUSH is executed as the last instruction of a         *
7781            *            software power fail routine. (%020104 000005)    *
7782            *                                                             *
7783            *               1)  Push TOS registers into memory.           *
7784            *               2)  FLUSH_OK := false, $FFFF. (so that the DCU will  *
7785            *                   not attempt to auto-restart MPE after a bad flush) *
7786            *               3)  Read cache status, CACSTAT, to determine cache  *
7787            *                   type - old, or new.                       *
7788            *               4)  Flush CPU cache. (use NEWC to flush new cache)  *
7789            *               5)  Clear FLUSH bit in CPX2.                  *
7790            *               6)  Initiate flush of IOA cache(s).           *
7791            *               7)  FLUSH_OK := true, $AAAA. (so DCU can enable an  *
7792            *                   auto-restart)                             *
7793            *               8)  Flush FLUSH_OK to main memory by reading two other *
7794            *                   memory banks at the same address.         *
7795            *               9)  Send SHUTDOWN message to main memory.     *
7796            *              10)  Wait for Certain Death, (PON goes to 0.)  *
7797            *                                                             *
7798            *      Entered with SP4A = !1000, BKX5 = 0                    *
7799            *                                                             *
7800            *      Re-written for new 128K cache, 8/6/85, Chris Shaker (2527)  *
7801            *                                                             *
7802            ***************************************************************
7803            *
7804    07A5    FLSH          JSZ  PSHA      SRNZ            ADD      BKX3 SF4B      Empty TOS registers; BKX3:=0, Set F4B (2527)
7806                                                                                    for IOMS and L2PF routines.          (2527)
7807    07A6          0353    ADDL                  0009      ADDL     CTR            Adr of FLUSH_OK; CTR for CACSTAT REGN (2527)
7809    07A7              UBA ADD       RA   WRX3  1C00      ADDL     SP3B           Set up to write FALSE to FLUSH_OK;   (2527)
7811                                                                                    BUSC opcode to read CACSTAT.         (2527)
7812    07A8                  CAD            DATA            ADD           WRX3       FLUSH_OK := FALSE ($FFFF) ; Set addr (2527)
7814                                                                                    for BUSC (addr does not matter).     (2527)
7815    07A9              RA  ADD            ROX3  SP3B      ADD           BUSC       Read FLUSH_OK (to wait for write);   (2527)
7817                                                                                    Request cache status.                (2527)
7818    07AA                  ADD                      RA    ADD      SP1B           Wait one cycle before reading OPA;   (2527)
7820                                                                                    SP1B := &FLUSH_OK;                   (2527)
7821    07AB              OPA ADD                0070      ADDL                     Mask to set FLUSH in CPX2;           (2527)
7823    07AC              REGN JSB  NEWC       ODD       UBB  ADD           CCPX       Go flush new cache or fall into code (2527)
7825                                                                                    that flushes the old cache; Set the  (2527)
7826                                                                                    FLUSH bit in CPX2.                   (2527)
7827            *
7828            *                  Flush the old CPU cache
7829            *
7830    07AD    OLDC          ADD                    0800      ADDL     SP2B           SP2B:= last block address            (2527)
7832    07AE    OLD1    UBB   JSB  IOAF       ZERO            REPN               0003  If block adr<0 exit;                 (2527)
7834                                                                                    Repeat next line 4 times.            (2527)
7835    07AF        UBB SP2B  IOR            ROX3  UBB  SP4A  ADD           DCTR CTR0  IOR blk adr into adr, do read;
7837                                                                                    inc adr by !1000.
7838    07B0                  JSB  OLD1       UNC   FFF8 SP2B  ADDL     SP2B           Loop back; SP2B := next lowest block (2527)
7840                                                                                    address.                             (2527)
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSR  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7842            *
7843            *                Flush IOA caches
7844            *
7845    07B1  IOAF       OPA  ADD                  00F0      ADDL                      Wait for flush, UBB = mask to clear   (2527)
7847                                                                                    the FLUSH bit in CPX2                 (2527)
7848    07B2                 ADD       RH                 UBB  ADD           CCPX      RH =0, Clear the FLUSH bit in CPX2    (2527)
7850    07B3       FFEB      ADDL      RA         6000         ADDL      SP3B         RA  = -21 (loop count for for IOA2),  (2527)
7852                                                                                    SP3B = IOA flush message              (2527)
7853    07B4  IOA1       RH   ADD                                    JSL  L2PF            UNC  UBA = logical IMB#, Is IOA there? (2527)
7855    07B5             UBA  JSB  IOA3      NEG   0080 RH   ADDL      RH                 If this one not there jump over       (2527)
7857                                                                                    the flush code,  RH = next IMB#       (2527)
7858    07B6       0040 XR9  IORL      XR9                        JSL  IOMS            UNC  XRA9 = send adr to IOA, Send the
7860                                                                                    message to the IOA
7861    07B7  IOA2           ADD                  RA           JSBI IOA2 RA          NZRO  Wait 64 clocks for IOA to flush.  (2527)
7863    07B8  IOA3 0200      ADDL      RG         AAAA         ADDL      SP2B             RG  = IOA flush message,          (2527)
7865                                                                                    SP2B =constant for FLUSH flag         (2527)
7866    07B9       FFEB      ADDL      RA         UBA  RH   JSBS IOA1            NZRO  RA  = -21 (loop cnt), Loop back      (2527)
7868                                                                                    if not done                           (2527)
7869            *
7870            ***    Set FLUSH OK to TRUE ($AAAA) for the DCU
7871            *
7872    07BA                 ADD                            SP1B ADD           WRX3      Write FLUSH success flag to 0000:0353 (2527)
7874    07BB                 ADD                            SP2B ADD           DATA      $AAAA Flags successful flush for DCU. (2527)
7876            *
7877            ***    Now, flush this word out to main memory
7878            *
7879    07BC                 ADD                            BKX3 INC      BKX3          BKX3 := 1 (next bank)                 (2527)
7881    07BD                 ADD                            SP1B ADD           ROA3      Read 1:0353 into the other cache set  (2527)
7883    07BE                 ADD                            BKX3 INC      BKX3          BKX3 := 2 (next bank)                 (2527)
7885    07BF            OPA  ADD                            SP1B ADD           ROA3      Read 2:0353 into the set that 0000    (2527)
7887                                                                                    0353 was in                           (2527)
7888    07C0                 INC       SP4A                       ADD                  SP4A :=1 ('Shutdown Main Memory'      (2527)
7890                                                                                    message for ECM), Wait for Read       (2527)
7891                                                                                    to complete)                          (2527)
7892    07C1  DIE        OPA  ADD                                    JSL  ECM             UNC  , Send memory shutdown msg.       (2527)
7894    07C2  WDIE            JSB  WDIE      UNC                     ADD                       Wait here until the bitter end.  (2527)
7896                                                                                    ( PON <- 0 )                          (2527)
7897            *
7898            ************************************************************************
7899            *                                                                      *
7900            *    NEWC - flush the new (128Kb) cache to main memory                 *
7901            *                                                                      *
7902            ************************************************************************
7903            *
7904    07C3  NEWC            JSB  IOAF      ZERO                    ADD           ROA3      Flush the IOA caches (slow jump for
7906                                                                                    Read); Send START to cache to
7907                                                                                    initiate the new cache FLUSH
```

```
RECORD  C S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7909            ********************************************************************
7910            *                                                                  *
7911            *    DCU Interrupt Command Processing Modules                      *
7912            *                                                                  *
7913            *    These modules will process a DCU command interrupt for the    *
7914            *    following commands:                                           *
7915            *                                                                  *
7916            *    DCU code     Description                                      *
7917            *                                                                  *
7918            *        1        Single Step command                              *
7919            *        2        Update ms byte of switch register                *
7920            *        3          "    ls   "  "   "     "                      *
7921            *        4        Replace contents of switch register              *
7922            *        5        DCU log to MPE (left byte)                       *
7923            *        6          "   "  "   "   (right byte)                    *
7924            *        7        Overtemp warning and condition                   *
7925            *        8        DCU log termination                              *
7926            *        9        Start DCU log to MPE processs                    *
7927            *                                                                  *
7928            ********************************************************************
7929            *
7930    07C4    DCMD      DBUS ADD  LRZ  SP4A       DCU0      ADDL LBL                    UBA := DCU bus value; UBB := @ DCU cmd table
7932                                                                                     SP4A <- COMMAND CODE ERROR              (2635)
7933    07C5        UBA   UBB  ADD                  WRX3  0009 UBA  SUBL              CRRY  UBA := table addr; Check for legal command
7935    07C6            STA   ADD  LSL            RAR       SREG JSZ  BDCC          UNC    set up for RSB to
7937                                                                                       jump table; SYSHALT if bad DCU code
7938    07C7        UBA       ADD       SP4A                   ADD                       SP4A <- STA & LSL(1)                    (2635)
7940    07C8        0080      ADDL                     00B2      ADDL      CTR             UBA := mask to clear DCUINT; CTR := sw reg
7942    07C9                  ADD                 RSB       UBA ADD             CCPX      Jump to table; Clear DCU interrupt
7944            *
7945            ********************************************************************
7946            *    DCU Command Table                                             *
7947            ********************************************************************
7948            *
7949    07CA    DCU0              JSB  CBE1       UNC   0200      ADDL                    Restart instr if DCU cmd = 0; RUNFF mask
7951    07CB                      JSB  DCU1       UNC   0600      ADDL                    Jump; UBB := mask to clear RUNFF
7953    07CC                      JSB  DCU2       UNC             ADD                     Jump
7955    07CD                      JSB  DCU3       UNC             ADD                     Jump
7957    07CE                      JSB  DCU4       UNC             ADD                     Jump
7959    07CF                      JSB  DCU6       CF2   00E5      ADDL      CTR           Jump, clear F2; CTR := @ control words
7961    07D0                      JSB  DCU6       SF2   00E5      ADDL      CTR           Jump, set F2; CTR := @ control words
7963    07D1                      JSB  DCU7       UNC             INC  LSL  SP1B CRF      Jump; SP1B := 2, clear RFLAG
7965    07D2                      JSB  DCU8       CF1   0018      ADDL      CTR           Jump, clear F1; CTR := DCUSTOR code 0
7967    07D3                      JSB  DCU9       CF1   00E5      ADDL      CTR           Jump, clear F1; CTR := @ control words
```

```
RECORD  C S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

7970            ********************************************************************
7971            *    Single Step command                                           *
7972            ********************************************************************
7973            *
7974    07D4  DCU1          ADD                          UBB  ADD          CCPX       ; Set RUN FF for DCU
7976    07D5        SPOA    JSZ  RSRT      UNC               ADD       DBUG          Restart instr; Inform DCU that system is
7978                                                                                    ready to start instruction
7979            ********************************************************************
7980            *    Update ms byte of switch register                             *
7981            ********************************************************************
7982            *
7983    07D6  DCU2 00FF DBUS ANDL                              ADD       CTX           Mask off the data byte from DCU; CTX := 0
7985    07D7            UBA  ADD  RLZ  SP4A                    ADD                     SP4A.(0:8) := DBUS.(8:8), shifted data byte
7987    07D8        00FF REGN ANDL                 0018        ADDL                    Mask off ms byte of Switch Update Register,
7989                                                                                    (SUR); UBB := DCUSTORE code
7990    07D9        UBA  SP4A IOR       REGN           UBB  JSB  CBE  CTR       UNC   Merge new ms byte with old SUR and store SUR
7992                                                                                    ; Set counter to DCUSTORE and jump to exit
7993            ********************************************************************
7994            *    Update ls byte of switch register                             *
7995            ********************************************************************
7996            *
7997    07DA  DCU3 00FF DBUS ANDL      SP4A                   ADD       CTX           Mask off data byte from DCU; CTX := 0
7999    07DB        FF00 REGN ANDL                 0018        ADDL                    Mask off ls byte of switch update register,
8001                                                                                    (SUR); UBB := DCUSTORE code
8002    07DC        UBA  SP4A IOR       REGN           UBB  JSB  CBE  CTR       UNC   Merge new ls byte with old SUR and store SUR
8004                                                                                    ; Set counter to DCUSTORE and jump to exit
8005            ********************************************************************
8006            *    Replace Contents of Switch Register command                   *
8007            ********************************************************************
8008            *
8009    07DD  DCU4        REGN ADD                             ADD       CTX           Read the Switch Update Register; CTX := 0
8011    07DE        0018       ADDL                    UBA  ADD       REGN          UBA := DCUSTORE code;  Replace Switch
8013                                                                                    Register with SUR contents
8014    07DF                   ADD                     UBA  JSB  CBE  CTR       UNC   ; Set counter to DCUSTORE and exit
8016            *
8017            ********************************************************************
8018            *    DCU log to MPE command                                        *
8019            ********************************************************************
8020            *
8021    07E0  DCU6        DBUS ADD  RLZ  RH              00E4       ADDL      CTR           1st data byte; CTR := buffer address
8023    07E1             REGN ADD  LSR                             ADD                     UBA := word count
8025    07E2        UBA  REGN ADD            NF2          REGN    ADD       BKX3 ICTR     Current buffer addr, skip if for 1st byte;
8027                                                                                    Set up bank; CTR := @ control words
8028    07E3             DBUS ADD  RRZ  RH                 UBA  ADD                WRX3 NF2  2nd data byte; Set up addr, skip if 1st byte
8030    07E4                  ADD                          UBB  ADD                ROX3 UNC  ; Read the 1st byte location
8032    07E5             REGN INC       REGN              RH   ADD                DATA UNC  Inc the byte count; Store 1st data byte
8034    07E6                  ADD                     RH   OPB  IOR                DATA      ; Store two data bytes
8036    07E7                  ADD                     0018      ADDL      CTR             ; CTR := DCUSTORE code
8038    07E8        SPOA      ADD       REGN                    JSZ  RSRT          UNC   Acknowledge to DCU, restart instruction
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8041            ********************************************************************
8042            *    Overtemp Warning and Condition                                *
8043            ********************************************************************
8044            *
8045    07E9  DCU7      SP4A JSB  CBE       POS   0018      ADDL      CTR             Ignore DCU interrupt if interrupts are
8047                                                                                    disabled; Set DCU acknowledge
8048    07EA            DBUS ADD  RRZ  SPOA SF2   8A01      ADDL      SP1B            SPOA := param (data byte from DCU), set F2;
8050                                                                                    SP1B := external label (%105001)
8051    07EB                 ADD       REGN CF1             JSZ  INT8           UNC   Acknowledge to DCU; Trap to segment# 1
8053            *
8054            ********************************************************************
8055            *    DCU Log Termination                                           *
8056            ********************************************************************
8057            *
8058    07EC  DCU8           ADD       REGN      00E5      ADDL      CTR             Acknowledge DCU; CTR := @ control words
8060    07ED                 ADD                                   ADD             Wait for CTR
8062    07EE            REGN ADD  LSL  SPOA      8F01      ADDL      SP1B            Parm.(0:15) := byte count;
8064                                                                                    STT# := %17, segment# 1
8065    07EF                 ADD       REGN                ADD       REGN            Zero byte count; Clear start flag
8067    07F0       SPOA      INC       SPOA CF1            JSZ  INT8           UNC   Parm.(15:1) := 1, clear F1;
8069                                                                                    Jump to terminate the process
8070            *
8071            ********************************************************************
8072            *    Start DCU to MPE log process                                  *
8073            ********************************************************************
8074            *
8075    07F1  DCU9           ADD       SPOA      8F01      ADDL      SP1B            Parm := 0; STT# 17, segment# 1
8077    07F2       SPOA      JSZ  RSRT      UNC           REGN JSZ  INT8           ZERO  Jump to start DCU log process if start flag
8079                                                                                    not set else ignore the whole thing
8080            ********************************************************************
8081            *    DCU Commands Exit Routine                                     *
8082            ********************************************************************
8083            *
8084    07F3  CBE            ADD       REGN      0200      ADDL                      Inform DCU that command has been processed;
8086                                                                                    Mask for RUN FF
8087    07F4  CBE1 UBB  CPX2 AND                                 ADD                  Isolate RUN FF
8089    07F5                 ADD                           UBA  JSL  STOP           ZERO  If not running then stop processing
8091    07F6       SPOA      JSZ  RSRT      UNC               ADD                  Restart instruction
```

```
RECORD  C S      ********** ALU A **********  ************** ALU B **************
  NO    ADDR LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8094           ************************************************************************
8095           *                                                                      *
8096           *    Read DCU Logging Facility -- RDCU                                 *
8097           *                                                                      *
8098           *    Entered with RA.(8:8), RB = bank, address of buffer               *
8099           *    RA (0:8) has a control parameter, (0 to transmit data,            *
8100           *    1 to enable DCU logging to MPE).  Normally, the first time        *
8101           *    thru is with a 1, then all remaining calls are with 0.            *
8102           *                                                                      *
8103           *    XRA100, XRB100 are used for address, bank                         *
8104           *    XRA101, XRB101 are used for byte count, start flag                *
8105           *                                                                      *
8106           ************************************************************************
8107           *
8108    07F7 RDCU          JSZ  PULM      SRL2          JSZ  PUL2          SRZ   Pull 2 TOS if SR = 0 else pull 1 if SR = 1
8110    07F8          RA   ADD  LRZ       EPOP  00E5    ADDL      CTR           Isolate control parm, pop 1 TOS;
8112                                                                               CTR := control words
8113    07F9          UBA  JSB  XDCU      ZERO      P   INC       P     RONP    If parm = 0 then jump to enable transfer;
8115                                                                               Read next instruction
8116    07FA               ADD       REGN EPOP          ADD       REGN          Byte count := logging started flag := 0, pop
8118                                                                               2nd TOS
8119    07FB               JSB  JDCU      UNC   001E    ADDL      CTR           Jump to exit routine;
8121                                                                               CTR := !1E (for DCUSTOR 6)
8122    07FC XDCU          ADD            EPOP          INC       REGN DCTR     Pop 2nd TOS; Set logging started flag,
8124                                                                               CTR := !E4 (buffer address)
8125    07FD          RB   ADD       REGN CF2       RA  ADD  RRZ  REGN          XRA100 := mem addr, clear byte position flg;
8127                                                                               XRB100 := mem bank
8128    07FE               ADD                 001C    ADDL      CTR           ; Set DCUSTOR 4 param
8130    07FF JDCU          ADD       REGN          JSZ  NEXT          UNC   Notify DCU to start (!1C) or enable (!1E)
8132                                                                               Jump to next instruction
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8134            *
8135            %0800
8136            *
8137            ********************************************************************
8138            *    DFLT (Double float) Instruction                               *
8139            ********************************************************************
8140            *
8141            $LUT INSTR=DFLT:0 000 011 000 xxx xxx, ENTRY=DFLT, SR=2
8142            *
8143    0800    DFLT      RB   ADD            HBF2  RA   RB   IOR                NZRO  F2:=SIGN(U); SKIP IF U <>0
8145    0801              ADD  ADD      SP4A                 JSB  DFLZ           UNC   SP4A:=0; JSB IF U = 0
8147    0802              RB   ADSB     SPOA                 RA   LINK     SP1B        SPOA:=ABS(MSU); SP1B:=ABS(LSU)
8149    0803              RB   ADD                  CCA   4780      ADDL     SP2B        CCA ON MSU; SP2B:=EXPONENT OFFSET
8151    0804              SP1B ADD  RLZ  SPOA                      ADD      SP3B        SPOA.(0:8):=LSU; SP3B:=0
8153    0805        SPOA       ADD  RLZ                      SP1B ADD  LRZ
8155    0806        RREG       ADD  LRZ            BIT8  UBA  UBB  REPC     SP1B        UBA:=MSU.(0:8), REPEAT IF NOT BIT8;
8157                                                                                    SP1B:=UBB:=LSU
8158    0807              UBA  QASL           BIT8        UBB  LINK     SP1B DCTX       NORMALIZE TO BIT8;
8160                                                                                    SP1B:=LSW, DECREMENT CTX FOR EACH SHIFT LEFT
8161    0808              SREG ADD            NF2         CTRS ASR                      SREG:=MSW, SKIP TO SET F1 SAME AS F2;
8163                                                                                    UBB.(0:9):=EXP ADJUSTMENT - 2
8164    0809        UBB   SREG IASR           SF1         SP1B LINK     RA            ROUND BY INCREMENTING BIT32, ASR, F1:=F2,
8166                                                                                    UBA.(0:10):=EXP ADJUSTMENT - 2; (S):=LSW
8167    080A        UBA   SP2B ADD        RB  F1HB        RA   ADD           DCC  NEXT  (S-1):=MSW+(EXP(U)+EXP(V)-256+2), SET SIGN,
8169                                                                                    DCC ON LSU, NEXT
8170    080B    DFLZ            ADD           CCA             ADD                 NEXT  SET CCA ON 0; NEXT
8172            *
8173            ********************************************************************
8174            *    FLT (Float) Instruction                                       *
8175            ********************************************************************
8176            *
8177            $LUT INSTR=FLT:0 000 100 111 XX,ENTRY=FLT,SR=1
8178            *
8179    080C    FLT       RA   ADD            HBF2              JSZ  PSHM           SR7   F2:=SIGN (S); JSZ IF SR = 6
8181    080D              ADD  ADD      SPOA  NF2         RA   JSB  FLTZ           ZERO  SPOA:=0, SKIP IF (S) >= 0; JSB IF (S) = 0
8183    080E              JSB  FLTN           UNC   4380      ADDL     SP1B              JSB IF (S) < 0; SP1B:=EXPONENT OFFSET
8185    080F              RA   ADD                          RA   ADD  RLZ               UBA:=(S), JSB IF (S)<0; UBB.(0:8):=(S)
8187    0810    FLT1      UBA  ADD  LRZ            BIT8        UBB  REPC     SP2B SF1  UBA:=MSW.(0:8), SKIP REPEAT IF BIT8;
8189                                                                                    SP2B:=UBB:=LSW, SF1
8190    0811              UBA  QASL           BIT8        UBB  LINK     SP2B DCTX       NORMALIZE TO BIT8;
8192                                                                                    SP2B:=LSW, DECREMENT CTX FOR EACH SHIFT LEFT
8193    0812              SREG ADD            F2          CTRS ASR            EPSH      SREG:=MSW, SKIP TO SET F1 SAME AS F2;
8195                                                                                    UBB.(0:9):=EXP ADJUSTMENT - 2
8196    0813        UBB   SREG IASR           CF1         SP2B LINK     RH            ROUND BY INCREMENTING BIT32, ASR, F1:=F2,
8198                                                                                    UBA.(0:10):=EXP ADJUSTMENT - 2; (S):=LSW
8199    0814        UBA   SP1B ADD        RA  F1HB        RA   ADD           CCA  NEXT  (S-1):=MSW+(EXP(U)+EXP(V)-256+2), SET SIGN,
8201                                                                                    CCA ON U, NEXT
8202    0815    FLTN      RA   JSBS FLT1      UNC         RA   SUB  RLZ               UBA:=-(S), JSB; UBB (0:8):=-(S)
8204    0816    FLTZ           JSZ  NEXT      CCA             ADD       RH    EPSH      JSB TO NEXT, CCA ON 0; NEW (S):=0, EPSH
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8207            ********************************************************************
8208            *     FCMP (Floating compare) Instruction                          *
8209            ********************************************************************
8210            *
8211            $LUT INSTR=FCMP:0 000 101 000 xxx xxx, ENTRY=FCMP, SR=4
8212            *
8213    0817    FCMP RD   CIR  IOR            CCA   RD   RB   XOR            EPP4 POS   SET CCG OR CCL ON MSU;
8215                                                                                    SKIP NEXT IF (SIGN(U) = SIGN(V)), EPP4
8216    0818         RD   RB   SUB                  RC   RA   LINK                 NEXT  COMPARE U - V;
8218                                                                                    NEXT IF (SIGN(U) <> SIGN(V))
8219    0819    FCM0      UBA  ADD              CCA   UBB  RD   XFRS                 NEG   SET CCA ON MS WORD;
8221                                                                                    RREG:=UBB, SKIP IF MSU < 0
8222    081A                   ADD              RREG       ADD                  DCC  NEXT  ; DCC ON LS WORD, NEXT IF MSU >= 0
8224    081B         RB   RD   JSBS FCM0        UNC   RA   RC   LINK                 UNC   COMPARE V - U; SKIP TEST AT TARGET
8226            *
8227            ********************************************************************
8228            *     FADD (Floating add) Instruction                              *
8229            ********************************************************************
8230            *
8231            $LUT INSTR=FADD:0 000 101 001 XX,ENTRY=FADD,SR=4
8232            *
8233    081C    FADD      RD   ADD        SP4A  F1HB       RB   ADD        SP2B F2HB        UBA:=SP4A:=ABS(U); UBB:=SP2B:=ABS(V)
8235    081D         UBA  UBB  SUB              CTF1  RC   RA   LINK           CLO         SF1 IF U>=V; CLO
8237    081E         FFC0 SP2B ANDL                   FFC0 SP4A ANDL     SP1B             UBA:=EXP(V); SP1B:=EXP(U)=(EXP(W) IF U>=V)
8239    081F         RD   RB   XOR              HBF2  UBB  UBA  SUB  LSL  XR0         F1  F2:=SIGN(W):=(SIGN(U) <> SIGN(V));
8241                                                                                    XR0:=(EXP(U) - EXP(V)) & LSL(1),
                                                                                          SKIP IF U>=V
8243    0820         003F SP4A ANDL       RH                 SREG JSB  EXUV SP1B       UNC   RH:=MANTISSA OF U;
8245                                                                                    SP1B:=EXP(V) AND JSB TO EXCHANGE U AND V
8246                                                                                      IF U<V
8247    0821         0040 UBA  IORL       XR1                XR0  ADD  LSL  CTX              XR1:=MAN(U) WITH LEADING 1;
8249                                                                                    CTX:=EXPONENT DIFFERENCE
8250    0822         003F SP2B ANDL       RH           0C00  XR0  SUBL                 NCRY  RH:=MANTISSA OF V;
8252                                                                                    SKIP IF EXPONENT DIFFERENCE > 24
8253    0823              RC   ADD        RA           RA   SP2B IOR            DCTR NZRO  RA:=LSU;
8255                                                                                    ANSWER IS U IF V=0, SKIP IF NOT, DCTR TO %FF
8256    0824    FADJ           ADD        SP0A         RREG      JSB  UANS            UNC   SP0A:=0; UBB:=LSV, JSB IF ANSWER IS ZERO
8258    0825              UBB  ADSB             CTF1  0040 RH   IORL      SP3B            UBA:=IF F2 THEN -LSV ELSE LSV;
8260                                                                                    UBB:=MAN(V) WITH LEADING 1, SP3B.(0:9):=0
8261    0826              UBB  ADSB             F1TC  UBA        REPC            DCTX CTX0  UBA:=IF F2 THEN -MSV ELSE MSV; UBB:=LSV
8263                                                                                      NORMALIZE V BY DIFFERENCE IN EXPONENTS
8264    0827         UBA       QASR                   UBB        LINK            DCTX CTX0  SHIFT RIGHT TO NORMALIZE UNTIL CTX IS
8266                                                                                      DECREMENTED TO 0
8267    0828         RREG XR1  ADD        SP4A  SF1    RREG RA   LINK      SP2B CCTX F5B   SP4A:=MSV + MSU, SF1; SP2B:=LSV + LSU,
8269                                                                                      CTX:=0; SKIP IF U AND V WERE EXCHANGED
8270    0829         UBA  UBB  IOR                  NZRO RD        ADD            HBF2 UNC   SKIP IF W <> 0; F2:=SIGN OF ORIGINAL U
8272    082A              SP4A JSB  ZROW           UNC  RB   SP2B XFRR           HBF2       SREG:=MSW, JSB IF W = 0;
8274                                                                                    SREG:=LSW, F2:=SIGN OF V IF U & V SWITCHED
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8276            ***************************************************************
8277            *   Normalize result of floating point operation              *
8278            ***************************************************************
8279            *
8280    082B  NORM       SREG ADD             BIT8       SREG REPC      SP2B ICTR      UBA:=MSW, NO REPEAT IF BIT8;
8282                                                                                   SP1B:=LSW, ICTR TO 0
8283    082C       UBA       QASL            BIT8       UBB  LINK      SP2B DCTX      RREG:=NORMALIZED MSW;
8285                                                                                   SP2B:=LSW, DECREMENT CTX FOR EACH SHIFT LEFT
8286    082D       RREG CIR  XFRS       SP4A F2         CTRS ASR            POP       RREG:=MSW, SP4A:=CIR TO SET CC LATER,
8288                                                                                     SKIP ON F2 TO SET F1 THE SAME;
8289                                                                                   UBB.(0:9):=EXP ADJUSTMENT - 1, POP
8290                                                                                     (MINUS 1 DUE TO NORMALIZATION TO BIT8)
8291    082E       RREG UBB  IASR           CF1        SP2B LINK      RB    POP       ROUND BY INCREMENTING BIT32, ASR, F1:=F2,
8293                                                                                   UBA.(0:10):=(EXP ADJUSTMENT - 1) + (1 FROM
8294                                                                                     BIT8 OR 2 IF ROUNDING CAUSED CARRY NTO
8295                                                                                     BIT9);
8296                                                                                   NEW (S) AFTER POPS := LSW, POP
8297    082F       UBA  SP1B ADD        RB   F1HB  UBA  SP1B CSL            SP2B LBF5      (S-1):=MSW = SIGN AND EXPONENT PLUS
8299                                                                                     MANTISSA AND EXPONENT ADJUSTMENT;
8300                                                                                   IF UNDERFLOW THEN SP2B.(0:1):=1,
8301                                                                                     IF UNDERFLOW OR OVERFLOW THEN F5B:=1
8302    0830       RA   UBB  IOR                NZRO  UBA  SP4A IOR            CCA  F5B   SKIP IF ABS(W) <> 0; SET CCL OR CCG ON MSW
8304                                                                                     SKIP NEXT IN UNDER/OVERFLOW
8305    0831                 JSZ  TRP3           UNC             ADD                 NEXT  UNDERFLOW TRAP IF ABS(W) = 0; NEXT OTHERWISE
8307    0832  FOV            JSZ  TRP3           UNC        SP2B JSZ  TRP2           POS   JSB FOR UNDERFLOW TRAP UNLESS OVERFLOW;
8309                                                                                   JSB FOR OVERLFOW TRAP IF NOT SP2B.(0:1)
8310    0833  EXUV 003F SP2B ANDL                             XR0  SUB  LSL  CTX  SF5B      UBA:=MANTISSA OF OLD U = NEW V;
8312                                                                                   CTX:= -(EXP(U) - EXP(V)) =
8313                                                                                     (EXP(NEW(U) - EXP(NEW(V)), SF5B TO
8314                                                                                     INDICATE U AND V WERE EXCHANGED
8315    0834       0040 UBA  IORL       XR1        0C00 XR0  ADDL            NEG   SP4A:=MAN(U) WITH LEADING 1;
8317                                                                                   SKIP IF EXPONENT DIFFERENCE > 24
8318    0835                 JSB  FADJ           UNC  RC    SP4A IOR            DCTR NZRO  JUMP BACK;
8320                                                                                   ANSWER IS NEW U IF NEW V=0, SKIP IF NOT,
8321                                                                                     DCTR TO %FF
8322    0836  ZROW           JSB  UAN1           CCA             ADD       RD   EPP2      JSB, CCA; (S-1) AFTER EPP2 := UBB:=0, EPP2
8324    0837  UANS           ADD                 F1         RD   ADD             CCA       SKIP IF U AND V WERE NOT EXCHANGED;
8326                                                                                   CCA ON MSU
8327    0838            RB   ADD          RD   CCA        RA   ADD             EPP2      (S-1) AFTER EPP2 := MSU; UBB:=LSU, EPP2
8329    0839  UAN1           ADD                 NEXT       UBB  ADD       RC    DCC       NEXT; (S) AFTER EPP2 := UBB, DCC
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8332            ********************************************************************
8333            *    FSUB (Floating subtract) Instruction                          *
8334            ********************************************************************
8335            *
8336            $LUT INSTR=FSUB:0 000 101 010 XX,ENTRY=FSUB,SR=4
8337            *
8338    083A    FSUB            JSB  FADD       UNC   8000 RB   ADDL       RB                ; TOGGLE SIGN OF V
8340            *
8341            ********************************************************************
8342            *    FMPY (Floating multiply) Instruction                          *
8343            ********************************************************************
8344            *
8345            $LUT INSTR=FMPY: 0 000 101 011 XX,ENTRY=FMPY,SR=4
8346            *
8347    083B    FMPY       RD   ADD            F1HB        RB   ADD        SP2B F2HB      UBA:=ABS(U), SP2B:=ABS(V)
8349    083C         FFC0 UBA   ANDL       SP4A        UBA  RC   IOR             CLO  ZERO  SP4A:=EXP(U), CLO, SKIP IF ABS(U) = 0
8351    083D         FFC0 SP2B  ANDL                   RA   SP2B IOR                  NZRO  UBA:=EXP(V), SKIP IF ABS(V) <> 0
8353    083E         RD   RB    XOR            HBF2  UBA  SP4A JSB  ZROW              UNC   F2:=SIGN(W), UBB:=EXP(V) + EXP(U)
8355                                                                                        JSB IF W = 0 (ABS(U)=0 OR ABS(W)=0)
8356    083F         003F RB    ANDL           C040  UBB  ADDL       SP1B                    UBA:=MANTISSA OF V,
8358                                                                                        SP1B:=EXP(V) + EXP(U) - 256 + 1
8359                                                                                          (PLUS 1 TO COMPENSATE FACT MANTISSA
8360                                                                                           IS SHIFTED RIGHT ONE BIT BY MPAD)
8361    0840         RREG RD   AND                0040 UBA  IORL       SP3B                    UBA:=MANTISSA OF U; SP3B:=MSV WITH LEADING 1
8363    0841              RA   ADD        SP4A        RREG UBA  IOR        RH                 SP4A:=LSV, RH:=MSU WITH LEADING 1
8365    0842                   ADD                      REPN                  16             UBA:=0, UBB:=0, REPEAT 23 TIMES
8367    0843         RH   UBA  MPAD                 RC   UBB  LINK             DCTR CTR0      MPAD U UNTIL CTR IS DECREMENTED TO 0
8369    0844              UBA  JSB  NORM       SF1         UBB  ADD              ICTR           SREG:=MSW, JSB TO NORMALIZE, SF1;
8371                                                                                        SREG:=LSW, ICTR TO %FF
8372            *
8373            ********************************************************************
8374            *   FDIV (Floating divide) Instruction                             *
8375            ********************************************************************
8376            *
8377            $LUT INSTR=FDIV: 0 000 101 100 XX,ENTRY=FDIV,SR=4
8378            *
8379    0845    FDIV       RD   ADD        RH   F1HB        RB   ADD                 F2HB NZRO  RH:=ABS(MSU); UBB:=ABS(MSV)
8381    0846         FFC0 UBB   ANDL       SP4A             RA   JSB  FDZR              ZERO  SP4A:=EXP(V);
8383                                                                                        JSB FOR DIVIDE BY 0 TRAP IF ABS(MSV)=LSV=0
8384    0847         RH   RC    IOR             NZRO  003F RB   ANDL       SP3B            SKIP IF ABS(U) <> 0; SP3B:=MANTISSA OF V
8386    0848                   JSB  ZROW       UNC   0040 UBB  IORL       SP2B            JSB IF W = 0;
8388                                                                                        SP2B:=MANTISSA OF V WITH LEADING 1
8389    0849                   ADD        SPOA CCO   FFC0 RH   ANDL                       SPOA:=0, CLO & CARRY, UBB:=EXP(U)
8391    084A         RD   RB    XOR            HBF2  UBB  SP4A SUB                      F2:=SIGN(W), UBB:=EXP(U) - EXP(V)
8393    084B         003F RD    ANDL                  3FC0 UBB  ADDL       SP1B            UBA:=MANTISSA OF U,
8395                                                                                        SP1B:=(EXP(U) - EXP(V)) + 256 - 1
8396    084C         0040 UBA   IORL                  RC   RFPN                 17         UBA:=MANTISSA OF U WITH LEADING 1, UBB:=LSU
8398    084D         UBA  SP2B  DVSB                  UBB  RA   LINK             DCTR CTR0      DVSB 24 TIMES UNTIL CTR IS DECREMENTED TO 0
8400    084E         UBA  SP2B  SUB             CTF1  UBB  RA   LINK             ICTR           F1:=CARRY (24TH ANSWER BIT), ICTR TO %FF
8402    084F                   ADD        SPOA F1HB  SP3B      ADD  RRZ                    SET 24TH ANSWER BIT IN SPOA(0),
8404                                                                                        UBB (8_8) :=MSW, ICTR TO %FF
8405    0850              UBB  ADD                  SF1         SP4A JSB  NORM            UNC   SREG:=MSW, SF1, SREG:=LSW, JSB TO NORMALIZE
8407    0851 FDZR              ADD                                  ADD            POP         , POP
8409    0852              RD   ADD                  CCA              ADD            POP         SET CCA ON MSU, POP
8411    0853                   JSZ  TRP5       UNC         RB   ADD                 DCC         FLOATING DIVIDE BY ZERO TRAP; SET DCC ON LSU
```

```
RECORD    C S          ********** ALU A **********  ********** ALU B **********
  NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8414            ********************************************************************
8415            *    FNEG (Floating negate) Instruction                            *
8416            ********************************************************************
8417            *
8418            $LUT INSTR=FNEG:0 000 101 101 xxx xxx, ENTRY=FNEG, SR=2
8419            *
8420      0854  FNEG RA   RB   IOR             NZRO  8000      ADDL                  SKIP IF FLOATING POINT WORD <> 0; UBB:=MASK
8422      0855                 JSB  DFLZ        UNC   UBB  RB   XOR           CCA     JSB IF = 0;
8424                                                                                 UBB:=(S-1) AND TOGGLE SIGN BIT, CCA
8425      0856           UBB  ADD        RB    NEXT        RA   ADD           DCC     (S-1):=UBB, NEXT; SET DCC ON (S)
8427            *
8428            ********************************************************************
8429            *    FIXR (Fix and round) Instruction                              *
8430            ********************************************************************
8431            *
8432            $LUT INSTR=FIXR:0 000 111 000 XX,ENTRY=FIXR,SR=2
8433            *
8434      0857  FIXR       RB   ADD  LSL  SP4A      8080      ADDL                  UBA:=SP4A:=MSU & LSL(1), UBB:=%8080
8436      0858       UBA  UBB  JSB  FUNZ      NCRY  F700      ADDL      SP3B        JSB IF NOT (EXPONENT >= -1);
8438                                                                                 SP3B:=CONSTANT TO TEST FOR OVERFLOW LATER
8439      0859           RB   ADD             HBF2  F500 SP4A ADDL              POS   F2:=SIGN(U);
8441                                                                                 UBB:=EXPONENT - 22,
8442                                                                                   SKIP IF (EXPONENT - 256) <= 0
8443      085A       003F RB   ANDL      SP4A       UBB  UBB  JSB  FXSL CTX       UNC   SP4A:=UBA:=MS MANTISSA;
8445                                                                                 CTX:=LS 8 BITS OF EXPONENT, JSB IF
8446                                                                                   ADJUSTED EXPONENT < 0
8447      085B       0040 UBA  IORL                      UBB  CAD       CTX             UBA:=MS MANTISSA WITH IMPLIED LEADING 1;
8449                                                                                 CTX:=-ADJUSTED EXPONENT - 1
8450      085C           UBA  ADD                       RA   REPC           DCTX CTX0  UBA:=MS MANTISSA;
8452                                                                                 UBB:=LS MANTISSA, REPEAT IF CTX <> 0, DCTX
8453      085D           UBA  QASR                      UBB  LINK           DCTX CTX0  UBA & UBB:=MANTISSA SHIFTED RIGHT,
8455                                                                                 END REPEAT WHEN CTX IS DECREMENTED TO 0
8456      085E           SREG IASR                      SREG LINK                      UBA, UBB:=ROUNDED ABSOLUTE ANSWER
8458      085F           UBA  ADSB      SP4A            UBB  LINK      RA   CLO       SP4A, RA:=-(ABSOLUTE ANSWER) IF < 0, CLO
8460      0860           UBA  JSB  FTST      UNC   UBB  UBB  LINK           CCRY    UBA:=MSW + LSW(0), JSB; CCRY
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8463          ************************************************************************
8464          *    FIXT (Fix and truncate) Instruction                                *
8465          ************************************************************************
8466          *
8467          $LUT INSTR=FIXT:0 000 111 001 xxx xxx, ENTRY=FIXT, SR=2
8468          *
8469    0861  FIXT RB   RB   JSB  FUNZ      POS   F700       ADDL       SP3B             UBA:=MSU & LSL(1), JSB IF EXPONENT < 0;
8471                                                                                     SP3B:=CONSTANT TO TEST FOR OVERFLOW LATER
8472    0862           RB   ADD            HBF2  F500 UBA    ADDL                  POS    F2:=SIGN(U);
8474                                                                                     UBB:=EXPONENT - 22,
8475                                                                                       SKIP IF (EXPONENT - 256) <= 0
8476    0863       003F RB   ANDL     SP4A       UBB  UBB    JSB   FXSL CTX        UNC    SP4A:=MS MANTISSA; CTX:=LS 8 BITS OF
8478                                                                                       ADJUSTED EXPONENT, JSB IF >= 0
8479    0864       0040 UBA  IORL                      UBB    CAD        CTX              UBA:=MS MANTISSA WITH IMPLIED LEADING 1;
8481                                                                                     CTX:=-ADJUSTED EXPONENT - 1 (1'S COMPLEMENT)
8482    0865            UBA  ADD                       RA     REPC              DCTX CTX0 UBA:=MS MANTISSA;
8484                                                                                     UBB:=LS MANTISSA, REPEAT IF CTX <> 0, DCTX
8485    0866            UBA  QASR                      UBB    LINK              DCTX CTX0 UBA & UBB:=SHIFTED MANTISSA, END REPEAT WHEN
8487                                                                                       CTX IS DECREMENTED TO 0
8488    0867            UBA  ADSB     SP4A             UBB    LINK       RA     CLO      SP4A & RA:=-(ABSOLUTE ANSWER) IF < 0, CLO
8490                                                                                     (CALLING FOR UBA & UBB SHIFTS ONCE MORE)
8491    0868            UBA  JSB  FTST      UNC   UBB  UBB    LINK              CCRY     UBA:=MSW + LSW(0), JSB; CCRY
8493    0869  FXSL           ADD        SP0A       SP3B CTRS JSB   FOVZ SP1B        CRRY   SP0A:=0;
8495                                                                                     JSB IF SHIFT COUNT >= 9, SP1B:=0 IF = 9
8496    086A       0040 SP4A IORL                      RA     REPC              DCTX CTX0 UBA:=MS MANTISSA WITH IMPLIED LEADING 1;
8498                                                                                     UBB:=LS MANTISSA, REPEAT IF CTX <> 0, DCTX
8499    086B            UBA  ADD  LSL                  UBB    LINK              DCTX CTX0 UBA & UBB:=MANTISSA SHIFTED LEFT,
8501                                                                                     END REPEAT WHEN CTX IS DECREMENTED TO 0
8502    086C            SREG ADSB     SP4A             SREG LINK       RA     CLO      SP4A, RA:=-(ABSOLUTE ANSWER) IF < 0, CLO
8504    086D            UBA  ADD                  UBB  UBB  LINK              CCRY     UBA:=MSW + LSW(0); CCRY
8506    086E  FTST      UBA  ADD            NZRO       SP4A ADD        RB     CCA      SKIP IF MS 17 BITS ARE NOT THE SAME;
8508                                                                                     (S-1):=MSW, CCA
8509    086F                 ADD            NEXT       RA   ADD              DCC      NEXT IF MS 17 BITS ARE THE SAME; DCC ON LSW
8511    0870                 ADD            NEXT            ADD              SCRY     NEXT IF MS 17 BITS ARE NOT THE SAME; SCRY
8513    0871  FOVZ RA   SP4A IOR            F2         SP1B JSZ   IRO             NZRO   UBA:=ZERO IF MANTISSA = 0, SKIP IF W < 0;
8515                                                                                     OVERFLOW TRAP IF EXPONENT <> 9
8516    0872                 JSZI IRO       UNC        UBA  JSZ   IRO             NZRO   UBA:=1, OVERFLOW TRAP IF W > 0;
8518                                                                                     OVERFLOW TRAP IF MANTISSA <> 0
8519    0873                 ADD            NEXT       UBA  CSR        RB     CCRY     NEXT; (S-1):=%8000, W IS -2**31, CCRY
8521    0874  FUNZ           ADD            CCO             ADD        RA              CLEAR OVERFLOW AND CARRY; (S):=0
8523    0875                 ADD            NEXT            ADD        RB              NEXT; (S-1):=0
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8526            ***************************************************************
8527            *    ASL (Arithmetic shift left) Instruction                  *
8528            ***************************************************************
8529            *
8530            $LUT INSTR=ASL:0 001 000 000 XX,ENTRY=ASL,SR=1,DSPL=6
8531            $LUT INSTR=ASL X:0 001 100 000 XX,ENTRY=ASL,SR=1,DSPL=6,X
8532            *
8533  0876  ASL  XC   DSPL JSB  *+3        ZERO  003F      ADDL                      UBA:=SHIFT COUNT, JSB IF 0; UBB:=%003F
8535  0877       UBA  UBB  AND             ZERO        RA   REPC                    UBA:=SHIFT COUNT MOD 64; UBB:=(S)
8537  0878       UBA       CAD             ZERO        UBB  ASL                     REPEAT UNTIL UBA IS DECREMENTED TO 0;
8539                                                                                ARITHMETIC SHIFT LEFT
8540  0879                 ADD             NEXT        SREG ADD       RA   CCA      NEXT; (S):=SHIFTED (S), CCA
8542            *
8543            ***************************************************************
8544            *    ASR (Arithmetic shift right) Instruction                 *
8545            ***************************************************************
8546            *
8547            $LUT INSTR=ASR:0 001 000 001 XX,ENTRY=ASR,SR=1,DSPL=6
8548            $LUT INSTR=ASR X:0 001 100 001 XX,ENTRY=ASR,SR=1,DSPL=6,X
8549            *
8550  087A  ASR  XC   DSPL JSB  *+3        ZERO  003F      ADDL                      UBA:=SHIFT COUNT, JSB IF 0; UBB:=%003F
8552  087B       UBA  UBB  AND             ZERO        RA   REPC                    UBA:=SHIFT COUNT MOD 64; UBB:=(S)
8554  087C       UBA       CAD             ZERO        UBB  ASR                     REPEAT UNTIL UBA IS DECREMENTED TO 0;
8556                                                                                ARITHMETIC SHIFT RIGHT
8557  087D                 ADD             NEXT        SREG ADD       RA   CCA      NEXT; (S):=SHIFTED (S), CCA
8559            *
8560            ***************************************************************
8561            *    LSL (Logical shift left) Instruction                     *
8562            ***************************************************************
8563            *
8564            $LUT INSTR=LSL:0 001 000 010 XX,ENTRY=LSL,SR=1,DSPL=6
8565            $LUT INSTR=LSL X:0 001 100 010 XX,ENTRY=LSL,SR=1,DSPL=6,X
8566            *
8567  087E  LSL  XC   DSPL JSB  *+3        ZERO  003F      ADDL                      UBA:=SHIFT COUNT, JSB IF 0; UBB:=%003F
8569  087F       UBA  UBB  AND             ZERO        RA   REPC                    UBA:=SHIFT COUNT MOD 64; UBB:=(S)
8571  0880       UBA       CAD             ZERO        UBB  ADD  LSL                REPEAT UNTIL UBA IS DECREMENTED TO 0;
8573                                                                                LOGICAL SHIFT LEFT
8574  0881                 ADD             NEXT        SREG ADD       RA   CCA      NEXT; (S):=SHIFTED (S), CCA
8576            *
8577            ***************************************************************
8578            *    LSR (Logical shift right) Instruction                    *
8579            ***************************************************************
8580            *
8581            $LUT INSTR=LSR:0 001 000 011 XX,ENTRY=LSR,SR=1,DSPL=6
8582            $LUT INSTR=LSR X:0 001 100 011 XX,ENTRY=LSR,SR=1,DSPL=6,X
8583            *
8584  0882  LSR  XC   DSPL JSB  *+3        ZERO  003F      ADDL                      UBA:=SHIFT COUNT, JSB IF 0; UBB:=%003F
8586  0883       UBA  UBB  AND             ZERO        RA   REPC                    UBA:=SHIFT COUNT MOD 64; UBB:=(S)
8588  0884       UBA       CAD             ZERO        UBB  ADD  LSR                REPEAT UNTIL UBA IS DECREMENTED TO 0;
8590                                                                                LOGICAL SHIFT RIGHT
8591  0885                 ADD             NEXT        SREG ADD       RA   CCA      NEXT; (S):=SHIFTED (S), CCA
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8594            ************************************************************************
8595            *     CSL (Circular shift left) Instruction                            *
8596            ************************************************************************
8597            *
8598            $LUT INSTR=CSL:0 001 000 100 xxx xxx, ENTRY=CSL, SR=1, DSPL=6
8599            $LUT INSTR=CSL X:0 001 100 100 xxx xxx, ENTRY=CSL, SR=1, DSPL=6, X
8600            *
8601    0886    CSL   XC    DSPL JSB  *+3        ZERO  003F       ADDL                      UBA:=SHIFT COUNT, JSB IF 0; UBB:=%003F
8603    0887          UBA   UBB  AND              ZERO        RA   REPC                      UBA:=SHIFT COUNT MOD 64; UBB:=(S)
8605    0888          UBA        CAD              ZERO        UBB  CSL                       REPEAT UNTIL UBA IS DECREMENTED TO 0;
8607                                                                                        CIRCULAR SHIFT LEFT
8608    0889                     ADD              NEXT        SREG ADD        RA   CCA       NEXT; (S):=SHIFTED (S), CCA
8610            *
8611            ************************************************************************
8612            *     CSR (Circular shift right) Instruction                           *
8613            ************************************************************************
8614            *
8615            $LUT INSTR=CSR:0 001 000 101 XX,ENTRY=CSR,SR=1,DSPL=6
8616            $LUT INSTR=CSR X:0 001 100 101 XX,ENTRY=CSR,SR=1,DSPL=6,X
8617            *
8618    088A    CSR   XC    DSPL JSB  *+3        ZERO  003F       ADDL                      UBA:=SHIFT COUNT, JSB IF 0; UBB:=%003F
8620    088B          UBA   UBB  AND              ZERO        RA   REPC                      UBA:=SHIFT COUNT MOD 64; UBB:=(S)
8622    088C          UBA        CAD              ZERO        UBB  CSR                       REPEAT UNTIL UBA IS DECREMENTED TO 0;
8624                                                                                        CIRCULAR SHIFT RIGHT
8625    088D                     ADD              NEXT        SREG ADD        RA   CCA       NEXT; (S):=SHIFTED (S), CCA
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8628            ****************************************************************
8629            *    DASL (Double arithmetic shift left) Instruction           *
8630            ****************************************************************
8631            *
8632            $LUT INSTR=DASL:0 001 010 000 XX,ENTRY=DASL,SR=2,DSPL=6
8633            $LUT INSTR=DASL X:0 001 110 000 XX,ENTRY=DASL,SR=2,DSPL=6,X
8634            *
8635    088E    DASL XC   DSPL JSB  *+4       ZERO            ADD                     UBA:=SHIFT COUNT, JSB IF = 0;
8637    088F              ADD                       003F UBA  ANDL     CTR            , CTR:=SHIFT COUNT MOD 64
8639    0890              RB   ADD                        RA   REPC           DCTR CTRO  UBA:=(S-1); UBB:=(S),
8641                                                                                   REPC CTR <> 0, DCTR
8642    0891              UBA  ASL                        UBB  LINK           DCTR CTRO  ARITHMETIC SHIFT LEFT
8644                                                                                     UNTIL CTR IS DECREMENTED TO 0
8645    0892              SREG ADD       RB   CCA        SREG JSB  DSFT           UNC   (S-1):=SHIFTED (S-1), CCA; UBB:=SHIFTED (S)
8647            *
8648            ****************************************************************
8649            *    DASR (Double arithmetic shift right) Instruction          *
8650            ****************************************************************
8651            *
8652            $LUT INSTR=DASR:0 001 010 001 XX,ENTRY=DASR,SR=2,DSPL=6
8653            $LUT INSTR=DASR X:0 001 110 001 XX,ENTRY=DASR,SR=2,DSPL=6,X
8654            *
8655    0893    DASR XC   DSPL JSB  *+4       ZERO            ADD                     UBA:=SHIFT COUNT
8657    0894              ADD                       003F UBA  ANDL     CTR            , CTR:=SHIFT COUNT MOD 64
8659    0895              RB   ADD                        RA   REPC           DCTR CTRO  UBA:=(S-1); UBB:=(S),
8661                                                                                   REPC CTR <> 0, DCTR
8662    0896              UBA  ASR                        UBB  LINK           DCTR CTRO  ARITHMETIC SHIFT RIGHT
8664                                                                                     UNTIL CTR IS DECREMENTED TO 0
8665    0897              SREG ADD       RB   CCA        SREG JSB  DSFT           UNC   (S-1):=SHIFTED (S-1), CCA; UBB:=SHIFTED (S)
8667            *
8668            ****************************************************************
8669            *    DLSL (Double logical shift left) Instruction              *
8670            ****************************************************************
8671            *
8672            $LUT INSTR=DLSL:0 001 010 010 XX,ENTRY=DLSL,SR=2,DSPL=6
8673            $LUT INSTR=DLSL X:0 001 110 010 XX,ENTRY=DLSL,SR=2,DSPL=6,X
8674            *
8675    0898    DLSL XC   DSPL JSB  *+4       ZERO            ADD                     UBA:=SHIFT COUNT
8677    0899              ADD                       003F UBA  ANDL     CTR            , CTR:=SHIFT COUNT MOD 64
8679    089A              RB   ADD                        RA   REPC           DCTR CTRO  UBA:=(S-1); UBB:=(S),
8681                                                                                   REPC CTR <> 0, DCTR
8682    089B              UBA  ADD  LSL                   UBB  LINK           DCTR CTRO  LOGICAL SHIFT LEFT
8684                                                                                     UNTIL CTR IS DECREMENTED TO 0
8685    089C              SREG ADD       RB   CCA        SREG JSB  DSFT           UNC   (S-1):=SHIFTED (S-1), CCA; UBB:=SHIFTED (S)
```

```
RECORD   C S                ********** ALU A **********  ************ ALU B **************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8688           ***************************************************************
8689           *    DLSR (Double logical shift right) Instruction             *
8690           ***************************************************************
8691           *
8692           $LUT INSTR=DLSR:0 001 010 011 xxx xxx, ENTRY=DLSR, SR=2, DSPL=6
8693           $LUT INSTR=DLSR X:0 001 110 011 xxx xxx, ENTRY=DLSR, SR=2, DSPL=6, X
8694           *
8695    089D   DLSR XC   DSPL JSB  *+4        ZERO           ADD                      UBA:=SHIFT COUNT
8697    089E             ADD                        003F UBA  ANDL      CTR           ; CTR:=SHIFT COUNT MOD 64
8699    089F             RB   ADD                        RA   REPC           DCTR CTRO  UBA:=(S-1); UBB:=(S),
8701                                                                                   REPC CTR <> 0, DCTR
8702    08A0             UBA  ADD  LSR                   UBB  LINK           DCTR CTRO  LOGICAL SHIFT RIGHT
8704                                                                                     UNTIL CTR IS DECREMENTED TO 0
8705    08A1             SREG ADD       RB   CCA        SREG JSB  DSFT           UNC   (S-1):=SHIFTED (S-1), CCA; UBB:=SHIFTED (S)
8707           *
8708           ***************************************************************
8709           *    DCSL (Double circular shift left) Instruction             *
8710           ***************************************************************
8711           *
8712           $LUT INSTR=DCSL:0 001 010 100 XX,ENTRY=DCSL,SR=2,DSPL=6
8713           $LUT INSTR=DCSL X:0 001 110 100 XX,ENTRY=DCSL,SR=2,DSPL=6,X
8714           *
8715    08A2   DCSL XC   DSPL JSB  *+4        ZERO           ADD                      UBA:=SHIFT COUNT
8717    08A3             ADD                        003F UBA  ANDL      CTR           ; CTR:=SHIFT COUNT MOD 64
8719    08A4             RB   ADD                        RA   REPC           DCTR CTRO  UBA:=(S-1); UBB:=(S),
8721                                                                                   REPC CTR <> 0, DCTR
8722    08A5             UBA  CSL                        UBB  LINK           DCTR CTRO  CIRCULAR SHIFT LEFT
8724                                                                                     UNTIL CTR IS DECREMENTED TO 0
8725    08A6             SREG ADD       RB   CCA        SREG JSB  DSFT           UNC   (S-1):=SHIFTED (S-1), CCA; UBB:=SHIFTED (S)
8727           *
8728           ***************************************************************
8729           *    DCSR (Double circular shift right) Instruction            *
8730           ***************************************************************
8731           *
8732           $LUT INSTR=DCSR:0 001 010 101 XX,ENTRY=DCSR,SR=2,DSPL=6
8733           $LUT INSTR=DCSR X:0 001 110 101 XX,ENTRY=DCSR,SR=2,DSPL=6,X
8734           *
8735    08A7   DCSR XC   DSPL JSB  *+4        ZERO           ADD                      UBA:=SHIFT COUNT
8737    08A8             ADD                        003F UBA  ANDL      CTR           ; CTR:=SHIFT COUNT MOD 64
8739    08A9             RB   ADD                        RA   REPC           DCTR CTRO  UBA:=(S-1); UBB:=(S),
8741                                                                                   REPC CTR <> 0, DCTR
8742    08AA             UBA  CSR                        UBB  LINK           DCTR CTRO  CIRCULAR SHIFT RIGHT
8744                                                                                     UNTIL CTR IS DECREMENTED TO 0
8745    08AB             SREG ADD       RB   CCA        SREG JSB  DSFT           UNC   (S-1):=SHIFTED (S-1), CCA; UBB:=SHIFTED (S)
8747    08AC   DSFT       ADD            NEXT            UBB  ADD       RA   DCC       NEXT, (S):=SHIFTED (S), DCC
8749                                                                                   ; STORE S, SET CC ON DOUBLE WORD
```

```
RECORD  C.S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8751            ***************************************************************
8752            *     QASL (Quadruple arithmetic shift left) Instruction        *
8753            ***************************************************************
8754            *
8755            $LUT INSTR=QASL:0 001 001 111 XX,ENTRY=QASL,SR=4,DSPL=6
8756            *
8757    08AD    QASL              ADD       SP4A                ADD                 SP4A:=0;
8759    08AE          X     DSPL JSB  QAS2       UNC          RA   ADD       SP3B       UBA:=SHIFT COUNT, JSB; SP3B:=(S)
8761    08AF                UBA  QASL                         UBB  LINK            DCTR CTR0  QASL UNTIL CTR IS DECREMENTED TO 0
8763    08B0  QAS1           SREG ADD       RD   CCA          SREG ADD                       STORE (S-3), CCA; UBB:=(S-2)
8765    08B1          SP0A      ADD                          UBB  ADD       RC   DCC       UBA:=(S-1); STORE (S-2), DCC
8767    08B2                    ADD                          UBA  ADD       RB   DCC         STORE (S-1), DCC
8769    08B3                    ADD             NEXT  SP3B        ADD       RA   DCC       NEXT; STORE (S), DCC
8771            *
8772            ***************************************************************
8773            *     QASR (Quadruple arithmetic shift right) Instruction       *
8774            ***************************************************************
8775            *
8776            $LUT INSTR=QASR:0 001 101 111 XX,ENTRY=QASR,SR=4,DSPL=6
8777            *
8778    08B4  QASR X    DSPL JSB  QAS2       UNC          RA   ADD       SP3B       UBA:=SHIFT COUNT, JSB; SP3B:=(S)
8780    08B5                UBA  QASR                         UBB  LINK            DCTR CTR0  QASR UNTIL CTR IS DECREMENTED TO 0
8782    08B6                SREG ADD       RD   CCA          SREG JSB  QAS1           UNC   STORE (S-3), CCA; SREG:=(S-2), JSB TO FINISH
8784    08B7  QAS2           RB   ADD       SP0A      003F UBA  ANDL      CTR              SP0A:=(S-1); CTR:=SHIFT COUNT MOD 64
8786    08B8                RD   ADD             RSB          RC   REPC            DCTR CTR0  UBA:=(S-3), RSB;
8788                                                                                       UBB:=(S-2), REPC IF CTR <> 0, DCTR
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8790           ********************************************************************
8791           *    Delete B (DELB) Instruction                                   *
8792           ********************************************************************
8793           *
8794           $LUT INSTR=DELB:0 000 000 001 xxx xxx, ENTRY=DELB, SR=2
8795           *
8796    08B9   DELB            JSZ  NEXT      UNC         RA   ADD       RB   EPOP       Jump to NEXT; New (S) := (S), pop 1 TOS
8798           *
8799           ********************************************************************
8800           *    Double Delete (DDEL) Instruction                              *
8801           ********************************************************************
8802           *
8803           $LUT INSTR=DDEL:0 000 000 010 xxx xxx, ENTRY=DDEL, SR=2
8804           *
8805    08BA   DDEL            JSZ  NEXT      UNC             ADD             EPP2       Jump to NEXT; Pop 2 TOS
8807           *
8808           ********************************************************************
8809           *    Delete A (DEL) Instruction                                    *
8810           ********************************************************************
8811           *
8812           $LUT INSTR=DEL:0 000 100 000 xxx xxx, ENTRY=DEL, SR=1
8813           *
8814    08BB   DEL             JSZ  NEXT      UNC             ADD             EPOP       Jump to NEXT; Pop 1 TOS
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
8817			**********												*	
8818			*												*	
8819			* Set Bounds Flag and Set Bounds Limit - Priv. Mode Instrs												*	
8820			*												*	
8821			**********												*	
8822	08BC	SBF		RA	ADD						JSZ	PULM			SRZ	UBA := RA; Pull 1 TOS if SR = 0
8824	08BD			UBA	ADD			EPOP			JSL	NEX1			UNC	Pop 1 TOS; Jump to exit routine
8826	08BE	SBL		RA	ADD						JSZ	PULM			SRZ	UBA := RA; Pull 1 TOS if SR = 0
8828	08BF			UBA	ADD			EPOP			JSL	NEX1			UNC	Pop 1 TOS; Jump to exit routine

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8831           ************************************************************************
8832           *                                                                      *
8833           *    Allows a user to disable CNTL 'B' recognition by the DCU.         *
8834           *    This will allow binary data to be transmitted through the         *
8835           *    terminal connected to the DCU without causing problems.           *
8836           *                                                                      *
8837           *    Entered with CTR = !19                                            *
8838           *                                                                      *
8839           ************************************************************************
8840           *
8841    08C0   DDE        RA   ADD                          JSZ  PULM          SRZ   UBA := RA; JSB to pull 1 TOS if SR = 0
8843    08C1                  ADD                     UBA   ADD        EPOP ZERO  ; Pop 1, skip if TOS = 0
8845    08C2                  ADD                001A       ADDL       CTR       ; CTR := DCUSTOR to disable
8847    08C3                  ADD       REGN             JSL  NEX1          UNC   Inform DCU; Jump to exit routine
```

```
RECORD  C S          ************ ALU A ************  ************** ALU B ***************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8850            ********************************************************************
8851            **                                                                **
8852            **      MCMD:  Message Command        020104 000004               **
8853            **                                                                **
8854            **      |------------|------------|------------|-----------|      **
8855            ** RA   | 0  0  0  0  0  0  |     BUSOP    |TO CODE |IB|BF|RE|      **
8856            **      |-------------------------------------------------|      **
8857            ** RB   |                  LSW OF MESSAGE                 |      **
8858            **      |-------------------------------------------------|      **
8859            ** RC   |                  MSW OF MESSAGE                 |      **
8860            **      |-------------------------------------------------|      **
8861            **                                                                **
8862            **                  IB = IGNORE BUSY                              **
8863            **                  BF = GO BUSY FLAG                             **
8864            **                  RE = REPLY EXPECTED                           **
8865            **                                                                **
8866            **      Entered with SP4A = 3                                     **
8867            **                                                                **
8868            ********************************************************************
8869            *
8870    08C4    MCMD            JSZ  PULM      SRL2            JSZ  PUL2          SRZ    IF SR=1 THEN PULL 1; IF SR=0 THEN PULL 2
8872    08C5         SR   SP4A  JSZS PULM      NEG   0C02       ADDL      RH             IF SR<3 THEN PULL 1; RH:= GO BUSY CMD
8874    08C6                    JSZ  PSHM      SR7             ADD             CCA      PUSH 1 IF SR=7; SET CCE
8876    08C7         0090       ADDL      RG          RH       REPN            BUSC 0003  RG:=CCPX MASK; GO BUSY ,REPEAT NEXT 3 TIMES
8878    08C8         0802       ADDL      SP4A        RC   OPB XFRR       BKX4 DCTR CTR0  SP4A:=BUSC CMD; BKX4:=MSW OF MSG
8880                                                                                    WAIT HERE TO GO BUSY
8881    08C9         0038       ADDL      RH              RB   ADD             WRX4       RH:=TIMEOUT CONSTANT; YREGB:=LSW OF MSG
8883    08CA                    JSZ  IRDN      MSGI  FFFE      ADDL                       HANDLE ANY EXISTING MSGI; UBB=CMD WRD MASK
8885    08CB              RA    JSB  MCM1      ODD   RREG RA   AND             BUSC       SEND CMD WORD; JMP IF RE BIT =1
8887    08CC                    JSB  MCM2      UNC             ADD             EPP2       JUMP TO FINISH ; POP 2 WORDS
8889    08CD    MCM1            JSB  MCM2      MSGI  RH        JSBI *-1  RH          ZERO   SKIP IF REPLY MSG RCVD; EXIT IF TIMEOUT
8891    08CE                    JSBC *-1       CCA           SP4A ADD             BUSC       SET CCL (IN CASE OF TIMEOUT); READ REPLY LSW
8893    08CF    MCM2 0400       ADDL                         RG   ADD             CCPX       UBA := read upper word, go unbusy; Clr MSGI
8895    08D0                    ADD            CCA   UBA     OPB  XFRR            BUSC       SET CCE ; READ MSW OF REPLY,GO UNBUSY
8897    08D1                    ADD                          SREG ADD         RB  EPOP              ; RB:=LSW OF REPLY,POP CMD WORD
8899    08D2                    ADD                          OPB  JSL  NEX1 RC          UNC    RC:=MSW OF REPLY
8901                                                                                    READ NEXT INSTRUCTION
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8903            ************************************************************************
8904            *                                                                      *
8905            *    CCPP - Communicate with Channel Program Processor                 *
8906            *                                                                      *
8907            *           This is a machine-dependent instruction which              *
8908            *           communicates with the Channel Program Processor via        *
8909            *           three parameters passed and returned on the TOS            *
8910            *                                                                      *
8911            ************************************************************************
8912            *
8913    08D3  CCPP 0003      ADDL      RH                    JSL  ADJS          UNC   Call #ADJS to make exactly 3 TOS valid
8915    08D4       0008      ADDL      RD                    ADD       BKX3 SF4B      BKX3,RD = DRTBANK @8;
8917                                                                                    Set F4 so no NRSM if module not there
8918    08D5       0180 RA   ANDL      RE                    JSL  L2PF BKX5        UNC   Mask out IMB#; Convert to physical
8920    08D6            UBA  JSB  CCPA      NEG               ADD       SP3B CF4B      If module not there abort (UBA from subr.);
8922                                                                                    Setup for #IOMS
8923    08D7       0C02      ADDL                            ADD                      Constant to set BUSY
8925    08D8                 ADD                 UBA         REPN            BUSC 0003  ; Go BUSY, repeat next line 3 times
8927    08D9                 ADD                             ADD             DCTR CTR0  ; Wait here to go BUSY
8929    08DA                 JSZ  IRDN      MSGI             ADD                      Exit to interrupt handler if MSGINT present
8931    08DB       RE   RE   ASL       RE                    JSL  IOMS          UNC   Adjust RE left 2 bits; Read IMBI reg# 0
8933    08DC       2000 RG   ANDL                        RD  ADD             ROX3       Mask out CPP bit; Read DRTBANK into OPB
8935    08DD            UBA  JSB  CCPA      ZERO  1000      ADDL      SP3B           Abort if no CPP; Set SP3B for write command
8937    08DE            RD   INC           ROB3        OPB ADD       BKX3           Read DRTOFFSET; Save DRTBANK in BKX3 (2319)
8939    08DF       0018 RE   ADDL                          ADD                      Add offset for MB0; BKX3 := DRTBANK  (2319)
8941    08E0                 ADD                 UBA   OPB  ADD       RD             ; BKX6 RD := MB0
8943    08E1            UBB  ADD           ROX3        UBB  INC             ROX3       Semaphore read MB0; Also read MB1    (2319)
8945    08E2                 CAD           DATA              CAD             CCA        Write semaphore to lock; Set CCL
8947    08E3            OPA  ADD       SPOA        F008      ADDL      RG             Skip if sem not locked; RG := CPPATN cmnd
8949    08E4                 ADD                       UBA   JSL  NEX1          NEG    ... else return CCL to user
8951    08E5            RA   JSB  CCPW      POS         OPB  ADD       RF             If GET = 0 then write; Save MB1
8953            *
8954            ************************************************************************
8955            *                                                                      *
8956            *    CCPR    is the read path of CCPP                                  *
8957            *                                                                      *
8958            ************************************************************************
8959            *
8960    08E6  CCPR 0006 RD   ADDL                            ADD             DATA       UBA := @MB2; Zero MB1
8962    08E7            UBA  ADD           ROX3        UBA  INC             ROX3       Read MB2, Read MB3
8964    08E8                 ADD           CCA         RF   ADD       RC             Set CCE; RC := MB1
8966    08E9                 ADD           DATA        OPB  ADD       RA             Zero MB2; RA := MB3
8968    08EA            OPA  ADD       RB          8007 UBB  ANDL                   RB := MB2; Zero MB3 error bits
8970    08EB            RD   ADD           WRX3        UBB  ADD             DATA       Write MB0 with orig. data; MB3 w/ no error
8972    08EC       SPOA      ADD           DATA              JSL  NEX1          UNC   ... to unlock semaphore; Done
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

8975            ********************************************************************
8976            *                                                                  *
8977            *     CCPW    is the write path of CCPP                            *
8978            *                                                                  *
8979            ********************************************************************
8980            *
8981    08ED  CCPW 0C02       ADDL                        ADD                        Send command routine
8983    08EE                  ADD                 UBA     REPN          BUSC 0003    Go BUSY; Repeat next line 3 times
8985    08EF                  ADD                         ADD           DCTR CTR0    ; Wait here to go BUSY
8987    08F0                  JSB  CCPI      MSGI         ADD                        Unlock semaphore and handle interrupt
8989    08F1                  ADD                         JSL  IOMS           UNC    ; Send CPPATN to CPP
8991    08F2          0007 RD ADDL                        ADD
8992    08F3                  ADD             CCA     UBA ADD           ROX3       Set CCE; Read CPP status (MB3)
8994    08F4             RD   ADD             WRX3  8000  ADDL                     Setup to write MB0; UBB := bit 0
8996    08F5          007F    ADDL                UBB OPB IOR           DATA       Isolate cmd; Rewrite MB3 w/ upper bit = 1
8998    08F6          UBA  RA AND             DATA        JSL  NEX1          UNC    Clear semaphore lock and send command; Done
9000            *
9001            ********************************************************************
9002            *                                                                  *
9003            *     CCPI    is the restart path of CCPP.  The semaphore is       *
9004            *             unlocked and control passes to #IRDN to handle a     *
9005            *             pending MSG interrupt.  In this event, the CCPP      *
9006            *             instruction is eventually restarted from the         *
9007            *             beginning.                                           *
9008            *                                                                  *
9009            ********************************************************************
9010            *
9011    08F7  CCPI       RD   ADD             WRX3        ADD                        Setup semaphore address
9013    08F8          SP0A    ADD             DATA        JSZ  IRDN          UNC    Restore semaphore; Handle interrupt
9015            *
9016            ********************************************************************
9017            *                                                                  *
9018            *     CCPA    is the abort path of CCPP.  TOS is set to -1 if the  *
9019            *             abort is due to a missing IOA, else it is set to zero*
9020            *             (indicating the IOA is present, but that a CPP is not*
9021            *             installed).                                          *
9022            *                                                                  *
9023            ********************************************************************
9024            $NOWARN
9025            *
9026    08F9  CCPA       XR9  ADD                         ADD           RB           No CPP, abort; RB := 0
9028    08FA                  ADD         RC          UBA ADD                CCZ  POS  RC := 0; Set CCG & setup store of RA
9030    08FB                  ADD         RA              CAD           RA           RA := -1 else RA := 0
9032    08FC                  ADD                         JSL  NEX1          UNC    ; Jump to exit routine
9034            $WARN
9035    08FD                  ADD                         ADD                       NOP to keep system happy                (2555)
9037    08FE                  ADD                         ADD                       NOP to keep system happy                (2555)
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9041          %0900
9042          ********************************************************************
9043          *                    INSTRUCTION JUMP TABLE                        *
9044          ********************************************************************
9045    0900  ITAB           ADD                    UBA  SP2B JSB  FCH1 SP2B       NZRO  0 REL JMP;ADD TO <POINT>.
9047    0901  INT1      XR0  JSB  INTH       UNC   0FFF UBA  ANDL       RH         NF1   1 INTERRUPT/HALT/RUN,XR0=CPVA COUNT;INT CODE
9049    0902                 JSBC WAIT       UNC         XR27 ADD        RH   WRS        2 WAIT,-1 ;DRT0 AS ADDR
9051    0903      0060 XR12  ADDL      SP4A              XR27 JSL  READ                UNC   3 READ SEC ADDR %4060;DRT0 ADDR
9053    0904      0060 XR12  ADDL      SP4A              XR27 JSL  WRIT                UNC   4 WRITE SEC ADDR %4060;DRT0 ADDR
9055    0905      C001       ADDL      XR2               XR26 JSB  DSRJ                UNC   5 DEV-SPECIFIED REL JMP,REC 1 BYTE;DEV#
9057    0906      C002       ADDL      XR2               XR26 JSB  IDY  XR0            UNC   6 IDENTIFY,XR2:=REC 2 BYTE;DEV#
9059    0907      0070 XR12  ADDL      SP4A              XR27 JSL  READ                UNC   7 READ CONTROL SEC ADDR %4070;DRT0 ADDR
9061    0908      0070 XR12  ADDL      SP4A              XR27 JSL  WRIT                UNC   8 WRITE CONTROL SEC ADDR %4070;DRT0 ADDR
9063    0909      003F XR12  ADDL      SP4A                   JSB  CLER                UNC   9 CLEAR,UNLISTEN CMD %403F.
9065    090A           OPA   ADD       XR2  SF3A              JSB  RMW                 ZERO  A READ-MOD-WRITE;F3A FOR RMW RETURN TO FCH1
9067    090B                 JSB  RRG       UNC                 ADD                        B READ REGISTER.
9069    090C                 JSB  IOW       UNC               UBA  ADD        RG   SF4B       C WRITE REG;F4B FOR IOW TO FCH1;M(<POINT>)
9071    090D                 JSB  CMD       UNC   4000          ADDL       RH              D COMMAND HP-IB;HP-IB CMD FORMAT
9073    090E           SP2B  INC       ROX3                    JSL  XDMA BKX6       UNC   E XDMA,READ M(<POINT>+1);DATA CHAIN
9075    090F                 JSB  WRIM      UNC         RH      ADD  SWAB            NEG   F WRT REL IMMED; SKIP IF DSP<0
9077    0910  CRCJ 003F XR12  ADDL      SP4A              XR22 JSB  CRCI SP3B       UNC   10 CRC INIT,SP4A:=%403F,UNLISTEN;REG#7
9079    0911  CRCD 0071 XR12  ADDL      XR2               XR22 JSB  CRCC SP3B       UNC   11 CRC COMPARE,SEC ADDRESS %4071;REG#7
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9082           ********************************************************************
9083           *                                                                  *
9084           *                     OBSI RESPONSE---                             *
9085           *                                                                  *
9086           *         0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15           *
9087           *        |---------|--|--|--|--|--|--|--|----------|---------|      *
9088           *        | 0  0  0 |TO|DA| 0 |DR|CR|NV| CHAN NO. | DEV NO. |      *
9089           *        |---------|--|--|--|--|--|--|--|----------|---------|      *
9090           *                     \  \     \  \  \--1="NOT VALID"  (JSB CPEX)  *
9091           *                      \  \     \  \----1="CHANNEL REQ"(JSB DMCK)  *
9092           *                       \  \     \------1="DEVICE REQ" (JSB EXAM)  *
9093           *                        \  \                                      *
9094           *                         \  \----------1="DMA ABORT"              *
9095           *                          \-----------1="TIMEOUT"                 *
9096           *                             IF BIT(6:2)=0 (JSB CHR)              *
9097           *                                                                  *
9098           ********************************************************************
9099           *
9100   0912    PAUL       RG   ADD  LSR       CF3A  0300      ADDL      XR18           SHIFT CHAN# LEFT TWICE; REG#3
9102   0913               UBA  ADD  LSR       CF2   0700      ADDL      XR22           SHIFT CHAN# TO BITS 13-15; REG#7
9104   0914               UBA  ADD  LSR             0200      ADDL      XR17           SHIFT CHAN#; REG#2
9106   0915         000F  UBA  ANDL      REGN        0A00      ADDL      XR25           SAVE CHAN# TO BE SERVICED; REG#A
9108   0916   GDRT  0008       ADDL                        BNKS ADD       XR15           UBA:=ADR OF DRT PTRS; XRB15:=BNKS
9110   0917               UBA  ADD             ROB3        RG   ADD  SWAB        POS    READ DRT BANK; SKIP IF NO DNV
9112   0918               UBA  INC             ROB3             JSB  CPEX        UNC    READ DRT OFFSET; EXIT IF DNV
9114   0919         FFF8       ADDL                        OPB  ADD       BNKS           UBA:=-8; BNKS:=DRT BANK#
9116   091A         RRFG XR9   ADD             ZERO        BKX5 REPC      RH             UBA:=MOD#-8,SKIP IF 0; REPEAT TIL MOD=0
9118   091B         RREG UBA   ADD             ZERO  0080  UBB  ADDL      RH             MOD#-8,SKIP IF 0; UBB:=CDEV#+80
9120   091C         RG         ADD   LRZ  SPOA  NF3A  RH   RH   ADD  LSL           F5B    SPOA:=OBSI INFO, SKIP IF F3=0 (IN CPGM)
9122                                                                                    ;UBB:=MCDEV*4, SKIP IF NOT IN CPGM
9123   091D                    ADD             RSB   UBB  OPB  ADD       XR27 ROS       RSB IF CALLED FROM #IRX
9125                                                                                    ; UBB:=MCDEV# *4 + OFFSET ;READ DRT0
9126   091E         UBB        INC             ROS   0007 RG   ANDL      XR26           READ M(<DRT1>),CPVA ADDR;DEV#
9128   091F               UBB  ADD       XR6             OPB  ADD       SP2B ROX3      DEV#;SP2B:=<POINT>-1,READ M(<POINT>-1)
9130   0920               OPA  ADD       XR3       1000  BKX5 IORL      XR19           M(<DRT1>):CDEV# FOR IOW
9132   0921         0003       ADDL      XR10                  INC  SWAB  SP3B CF4B      %0003:REG#1
9134   0922         UBA  OPA   INC             WRX3  UBA  XR27 ADD       XR28 ROAS      M(<DRT1>)+5 FOR(DMCK):READ M(<DRT3>),STATUS
9136   0923         SPOA       ADD   LSR  RH                  OPB  JSB  IOR  SP1B      UNC    SP4A(15:1):=OBSI(6:1);SP1B:=M(<POINT>-1)
9138   0924               RH   ADD   LSR  RG                  RG   REPN           0003   XRO(15:1):=DRT3(5:1);SHIFT <REG#1> RIGHT 4X
9140   0925               OPA  ADD       XR4             UBB       ADD  LSR       DCTR CTRO  <DRT3>:UBB:=<REG#1> SHIFTED 4 TIMES
9142   0926         UBB        ADD       XR8   F1HB  UBA       CSR       XR1  CF1            XR8:=%8000+<SHIFTED REG#1>;CSR STATUS
9144   0927         E000 UBB   ANDL                  0004  UBB  ANDL                ZERO   MASK DRT3.(15:3); CHECK IF FROM DATA SUSP
9146   0928         RG         ADD   LSR  SP4A            UBA  ADD           SF2            SP4A(15:1):=OBSI(4:1);SET F2 IF DATA SUSP
9148   0929         SPOA       JSB   DMCK        ODD   C000 UBB  SUBL      XRO            DMCK IF CHAN REQ;DRT3(10.XXX1)???
9150   092A                    JSB   EXAM        UNC         RH   JSB  CHR           EVEN   CHR IF ILLEGAL CSRQ;EXAM IF DEV REQ
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9153          ************************************************************************
9154          *                    OBSI(7:1)=1,IF REQ IS DUE TO COMPLETION OF        *
9155          *                    DMA TRANSFER,OR OCURRENCE OF SOME ANTICIPATED     *
9156          *                    EVENTS OF THE CHANNEL.                            *
9157          *                    IN GENERAL,EITHER OF OBSI(3:2) WILL ALSO SET,     *
9158          *                    IF THERE IS AN ERROR.                             *
9159          *                    1. CHECK PGM  STATUS IF OBSI(3:2)=00.             *
9160          *                    2. GO HANDLE THE HARDWARE TIMEOUT IF OBSI(3:1)=1, *
9161          *                       JSB CHR,ERROR CODE E004                        *
9162          *                    3. THEN,STORE DMA EXTENDED ADDR (FROM REG#8) TO   *
9163          *                       CPVA LOCATION4,                                *
9164          *                    4. STORE DMA ADDRESS (FROM REG#9) TO CPVA         *
9165          *                       LOCATION5,                                     *
9166          *                    5. READ REG#B,&SHIFT RIGHT 3 TIMES,&'IOR' %C000   *
9167          *                       AS THE DMA ABORT MESSAGE.                      *
9168          ************************************************************************
9169  092B  DMCK E004      ADDL      XR13        SP3B XR25 JSB  IOR  SP3B       NZRO  TIMEOUT ERR CODE;READ REG#B,DMA STATUS
9171  092C           RG    ADD  LSR  RH              SP4A ADD  LSR             EVEN  REG#B(SHIFTED ONCE);SKIP IF OBSI(3:1)=0
9173  092D           UBA   ADD  LSR  RH                   JSB  CHR             UNC   REG#B(SHIFTED TWICE);CHR IF OBSI(3:2)=1X
9175  092E      E020       ADDL      XR13        RH       ADD       BKX7 SF1        ERROR CODE;BKX7:=SHIFTED (1) REG#B
9177  092F           SP4A JSB  EXAM      EVEN             ADD                       EXAM IF OBSI(3:2):=00;
9179  0930           RH   ADD  LSR  SP4A        SP3B XR18 JSBS IOR  SP3B       NZRO  REG#B(SHIFTED 3X);READ REG#8,EXT DMA ADDR
9181  0931           RG   ADD            DATA        XR24 JSB  IOR  SP3B       NZRO  DRT1+4:=<REG#8>;READ REG#9,DMA MEMORY ADDR
9183  0932      RG        ADD            DATA             ADD            CF1       <CPVA>+5:=<REG#9>,EXT MEM ADDR
9185  0933  DMAB C000 SP4A IOR'         SP4A             JSL  DCER        ZERO  ABORT WITH INTERRUPT WITH ERROR CODE
9187          ************************************************************************
9188          *                    ROUTINE TO EXAMINE THE PROGRAM STATUS AND         *
9189          *                    DECIDE HOW TO SERVICE THE CSRQ BASED ON THE       *
9190          *                    RESULTS OF 'OBSI' AND DRT3(FOURTH DRT WORD).      *
9191          *                    1. READ THE DRT3,& MASK DRT3(12:4)                *
9192          *                    2. JSB NRUN IF DRT3(0:1)=0 HALT OF CHAN PGM ,     *
9193          *                    3. JSB STRT IF DRT3(0:2)=11  (START OF CHAN PGM,  *
9194          *                       AND/OR THE SIOP/HIOP HAS STARTED BUT NOT YET   *
9195          *                       SERVICED).                                     *
9196          *                    4. JSB AE IF DRT3(0:2)=10    &(DRT3(12:4)=0000 OR*
9197          *                       BIT15=1);MEANING REQUEST OF START CHAN PGM,    *
9198          *                       & (IF CP IS IN WAIT STATE OR NO CP SUSPENDED.  *
9199          *                    5. JSB CHR IF OBSI(7:1)=0    ,NO CHAN REQUEST.    *
9200          ************************************************************************
9201  0934  EXAM 000F XR4  ANDL      SP4A             ADD                       MASK DRT3(12:4);UBB(15:1):=-DRT3(13:1)
9203  0935           XR4  JSB  NRUN       POS   0F00      ADDL      XR30        NRUN IF DRT3:=<0XX..XXXX>;REG#F
9205  0936           SP4A JSB  AE         ODD   UBA  UBA  JSB  STRT BKX6  NEG   AE IF <10X..0001>;STRT IF <11X..XXXX>
9207  0937                JSB  CHR        NF1        SP4A JSB  AE          ZERO  CHR IF OBSI(7:1)=0,DEV REQ,AE IF <10X..0000>
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9210           ************************************************************************
9211           *                    JSB DMAM IF DRT3(14:1)=1,CP SUSPENDED WHILE       *
9212           *                    CHAN DMA TRANSFER IN PROGRESS                     *
9213           ************************************************************************
9214     0938  EXM1 0040 RG   ANDL      RH              XR1  JSL  DMAM           ODD   MASK REG#B(9:1);DMAM IF DMA,DRT3(14:1)=1
9216           ************************************************************************
9217           *                    CHECK DRT3(12:2);THERE IS ONLY ONE                *
9218           *                    DRT3(12:4) CAN BE SET AT ANY ONE TIME).           *
9219           ************************************************************************
9220     0939  AD        XR6  ADD       RG              XR30 JSB  IOW  SP3B      NZRO  WRT REG#F:=DEV#
9222           ************************************************************************
9223           *                  MASK OFF THE DRT3(0:3),   & RETURN IT BACK          *
9224           *                  TO DRT3 ADDRESS,ENABLE RECOGNITION OF PHI           *
9225           *                  CONDITIONS BY WRITING REG#3:=%7FFF.                 *
9226           *                  JSB CRCI IF THE FIRST INSTR BIT8 SET,OTHERWISE,     *
9227           *                  GO FETCH. SP2B:=CP POINTER.                         *
9228           ************************************************************************
9229     093A  AE              CAD  LSR  RG              XR18 JSB  IOW  SP3B      NZRO  WRITE TO REG#3:=%7FFF
9231     093B        E000 XR4  ANDL      XR4             XR28 ADD            WRS       MASK DRT3;DRT3 ADDR
9233     093C             SP2B ADD                ROB3   UBA  ADD            DATA      READ M(<POINT>-1);SAVE DRT3
9235     093D        0040 SP1B ANDL                    XR0  JSB  FCH0           NZRO  MASK;GO FETCH
9237     093E             UBA  JSB  CRCJ      NZRO      SP2B INC       SP2B       DO CRCJ IF SP1B.(9:1) SET;INC <POINT>
9239     093F                  JSB  FCH1      ZERO      SP2B ADD                  FCH1 IF DRT3:=<10..XXX1>;<POINT>
9241     0940  NRUN      XR4  JSB  NRNB      NF1       SREG ADD            CF1       NRNB IF OBSI(7:1)=0,<POINT>-1 FOR FCH0
9243     0941             SP4A JSB  CHRA      ODD   UBA UBA  JSB  CHR            POS   HLT PGM IF <01X.0001>;CHR IF<00X XXXX>
9245     0942                  JSB  EXM1      UNC       SP4A JSB  CHRA           ZERO  GO CONTINUE PROGRAM;HLT PGM IF<01X...0000>
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9248            ************************************************************************
9249            *                   STARTING THE CHANNEL PROGRAM,DRT3:=<11X.XXXX>      *
9250            *                   1. ISSUE 'SIOP' FOR THE CHAN WITH BIT12 SET IN     *
9251            *                      DATA TO CLEAR THE CSRQ.                         *
9252            *                   2. SET DRT3=%8000,REQUEST START OF THAT CHAN PGM   *
9253            *                   3. JSB AD IF NOT CONTROLLER IN CHARGE.             *
9254            *                   4. IF CIC,ASSERT 'REMOTE ENABLE' AND THE PARITY    *
9255            *                      LOCKUP. REG#6:=%0060.                           *
9256            *                   5. THEN,ENABLE DEVICE PARALLEL POLL RECOGNITION    *
9257            *                      SET CORRESPONDING PPRN# IN REG#4.               *
9258            *                   6. JSB AD. SET DRT3(0:1)=1.                        *
9259            ************************************************************************
9260    0943    STRT      XR14 ADD       XR4  CF1   4000      ADDL      SP3B            SET DRT3:=%8000;SP3B:=%4000,SIOP CMD
9262    0944                   JSB  IOW       UNC   0008 XR26 ADDL      RG              SEND THE SIOP CMD
9264    0945              XR8  JSB  AD        EVEN  0060      ADDL      RG              JSB IF NOT CONTROLLER;%0060
9266    0946        0018  XR6  ADDL      XR2                   XR21 JSB  IOW  SP3B      NZRO  %0018+DEV#;WRITE TO REG#6
9268    0947                   JSB  RMW       UNC   0400      ADDL      SP3B            SET BIT FOR DEV TO 1 IN REG#4
9270            ************************************************************************
9271            *                   OBSI(7:1)=0(NF1),WITH DRT3=<0X..XXXX>.             *
9272            ************************************************************************
9273    0948    NRNB      UBB  JSB  CHR       POS             ADD       BKX6 SF1        CHR IF DRT3(0:2)=00;BKX6 FOR RMW RETURN
9275    0949              SP4A JSB  HLTP      ZERO        SP4A JSB  HLTP          ODD   HLTP IF DRT3(12:4)=0 OR BIT15=1
9277            ************************************************************************
9278            *                   CHECK THE CPVA ADDRESS.                            *
9279            *                   IF < %40,NO INTERRUPT IS GENERATED IN CHR,         *
9280            *                   SINCE THAT WOULD WRITE INTO CPU RESERVED MEMORY    *
9281            *                   AREA;OTHERWISE CONTINUE WITH INTERRUPT.            *
9282            ************************************************************************
9283    094A    CHR   FFE0 XR3  ANDL                  0C00      ADDL      SP3B            TEST IF THE INTERRUPT IS WITHIN %20;REG#C
9285    094B               SREG ADD                   WRX3      UBA  JSB  CHR1      ZERO  <DRT1> AS ADDR;CHR1 IF THE INT ADDR <%20
9287    094C               XR13 JSB  IOW       DATA  0008 XR26 ADDL      RG              CPVA:=ERROR CODE;WRITE TO REG#C:=%0008+DEV#
9289    094D    CHR1               JSB  CHRA      UNC                   JSB  HLTP XR9   F1    CHRA IF CONTINUE W/O INT;HLTP RTN(XR9-CLFF)
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9292           ************************************************************************
9293           *                    INSTRUCTION FETCH ROUTINES                        *
9294           *                    THIS ROUTINE FETCHES THE NEXT CHANNEL INSTRUCTION*
9295           *                    TESTS FOR POWER FAILURE INTERRUPT,CHECKS VALIDITY*
9296           *                    OF THE INSTRUCTION CODE AND BRANCHES TO THE      *
9297           *                    EXECUTE CODE THROUGH THE DECODE TABLE.           *
9298           *                    FCH4 = WILL INCREMEMT POINTER BY 3               *
9299           *                    FCH1 = WILL INCREMENT POINTER BY 1               *
9300           *                    FCH0 = WILL INCREMENT POINTER BY 1               *
9301           ************************************************************************
9302    094E   FCH4           JSB  FCH1       ZERO  0003 SP2B ADDL       SP2B            INC POINTER BY 3
9304    094F   FCH1           ADD             CF2        SP2B INC            ROX3        ;INC POINT AND READ M(<POINT>-1)
9306    0950   FCH0 0008 CPX1 ANDL                       UBB  INC       SP2B ROA3        CHECK POWER FAIL;READ M(<POINT>)
9308    0951        UBA       JSB  CPEX       NZRO       OPB  ADD  LRZ  RG   CF5B        JSB IF POWER FAIL,UPPER BYTE
9310    0952        E001      ADDL      SP4A             ITAB UBB  ADDL LBL             ERROR CODE(FOR JSB V);JSB ITAB IF VALID
9312    0953        UBB       ADD             WRX3       OPB  ADD  RLZ  SP1B CF4B        ITAB ADDR;SP1B:=SWAPPED M(<POINT>-1)
9314    0954             UBB  ADD  LRZ  RH    RAR   OF00 UBB  ANDL      SP3B            SEND ADDR TO RAR,RH:=OPB(LOWER BYTE);REG#
9316    0955             XR5  ADD       RG    CF3A       SP1B CAD       SP1B CF1         <DRT3>;
9318    0956        0003 RH   ANDL      XR0        FFEE RG   ADDL      XR23       NEG   CPVA COUNTS;CHECK VALID I/O INSTR
9320    0957             OPA  ADD       SP0A RSB                JSL  V             UNC   SAVE M(<POINT>);V TO ABORT IF INVALID INST
9322           *
9323           ************************************************************************
9324           *                    READ REGISTER INSTRUCTION                         *
9325           *                    I----------I----------I----------I----------I*
9326           *                    I 0  0  0  0  1  0  1  1  X  X  X  X  REG.NUMBERI*
9327           *                    I----------I----------I----------I----------I*
9328           *                    I                RETURNED WORD               I*
9329           *                    I----------I----------I----------I----------I*
9330           ************************************************************************
9331           *
9332    0958   RRG        SP2B ADD             WRX3            JSB  IOR            UNC   <POINT> AS ADDR;JSB TO READ THE REG
9334    0959              RG   ADD             DATA            JSB  FCH1           UNC   <POINT>:=<WORD>;REFETCH NEXT INSTR
9336           *
9337           ************************************************************************
9338           *                    READ-MODIFY-WRITE INSTRUCTION                     *
9339           *                    I----------I----------I----------I----------I*
9340           *                    I 0  0  0  0  1  0  1  0  X  X  X  X  REG.NUMBERI*
9341           *                    I----------I----------I----------I----------I*
9342           *                    I X  X  X  X  X  X  X  X  X  X  X  S  BIT NUMBERI*
9343           *                    I----------I----------I----------I----------I*
9344           ************************************************************************
9345           *
9346    095A   RMW  000F XR2  ANDL      RH                      JSB  IOR            UNC   MASK WD(12:4);JSB TO READ
9348    095B        XR2  ADD                       002F RH   SUBL      CTR              CTR:=MASK ADDRESS
9350    095C        FF9F      ADDL      RH                  0010 UBA  ANDL       ZERO   TIMER FOR CHRS(ABOUT 20US);SKIP IF CLEAR
9352    095D             ADD                       RG   REGN IOR       RG       UNC   ; RG:=RG WITH BIT SET
9354    095E             JSB  IOW       UNC   UBB  REGN XOR       RG                 JSB TO WRITE TO REG#; RG:=RG WITH BIT CLEAR
9356    095F             JSB  FCH1      F3A                JSB  CHRS           F5B   FCH1 IF FROM RMW;CHRS IF FROM INT/HLT(CIC)
9358    0960             JSB  CHRX      UNC        BKX6 JSB  AD             NEG   CPX1 IF FROM NRNB;FOR STRT RETURN
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9361            ********************************************************************
9362            *                   COMMAND HP-IB INSTRUCTION                      *
9363            *                   I----------I----------I----------I----------I*
9364            *                   I 0  0  0  0  1  1  0  1  X  X  X  X  COUNT  I*
9365            *                   I----------I----------I----------I----------I*
9366            *                   I     COMMAND NO.1       I     COMMAND NO.2     I*
9367            *                   I----------------------I----------------------I*
9368            *                   I     COMMAND NO.3       I     COMMAND NO.4     I*
9369            *                   I----------------------I----------------------I*
9370            *                   I     COMMAND NO.5       I     COMMAND NO.6     I*
9371            *                   I----------------------I----------------------I*
9372            *                   I     COMMAND NO.7       I     COMMAND NO.8     I*
9373            *                   I----------------------I----------------------I*
9374            *                   FIRST, CHECK IF FIFO EMPTY                     *
9375            *                   THEN XFER POINTER TO SPOA,UNPACK THE COMMAND   *
9376            *                   WORDS, AND SEND THEM OUT OVER HP-IB VIA WRITING*
9377            *                   TO REGISTER#0 (FIFO DATA BUFFER)               *
9378            ********************************************************************
9379    0961    CMD       SP2B ADD       SPOA       0007 RH   ANDL      SP1B      NZRO  <POINT>;SKIP IF NOT 0
9381    0962                   JSB  SUSP       UNC   RREG      INC       SP1B            CHECK FOR FIFO ROOM;SP1B:=8
9383    0963    CMD0      OPA  ADD  LRZ                        ADD       SP3B            SETUP LEFT BYTE CMD;REG#0
9385    0964              OPA  ADD  RRZ  SP4A UNC   UBA  RH    IOR       RG   CF1  UNC   SETUP RIGHT BYTE CMD;RG:=(ATN+LBYTE),CF1
9387    0965    CMD1 SPOA      INC       SPOA ROX3  RH   SP4A IOR       RG   SF1        NEXT MEM READ;RG:=(ATN+RBYTE),SF1
9389    0966                   JSB  IOW        UNC   FFFF SP1B ADDL      SP1B            GO TO WRITE REG#0;DECREMENT COUNT
9391    0967                   JSB  CMD1       NF1         SP1B JSB  FCH4            ZERO  GET RIGHT BYTE IF NF1;FCH4 IF DONE
9393    0968                   ADD                                JSB  CMD0            UNC   CMD0 FOR THE NEXT BYTE
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9396           ;*****************************************************************
9397           ;                   WAIT INSTRUCTION                            *
9398           ;                   I----------I----------I--I--I--------I--I-----I*
9399           ;                   I 0  0  0  0  0  0  1  0 CC CI  X  X  X  S  CPVAI*
9400           ;                   I----------I----------I--I--I--------I--I-----I*
9401           ;                   I 0  0  0  0 OISERIAL POLL RESPONSE/INTERRUPT CODEI*
9402           ;                   I----------I-----------------------------------I*
9403           ;                   THIS INSTRUCTION CAUSES A SUSPENSION OF THE     *
9404           ;                   CHANNEL PGM EXECUTION UNTIL THE DEV AGAIN       *
9405           ;                   REQUESTS SERVICE.  IT MERELY ALLOWS THE CHANNEL  *
9406           ;                   TO START POLLING ALL THE DEVICES.               *
9407           ;                   1. FIRST UPDATE THE DRT TABLE;SAVING DRT3(0:3),  *
9408           ;                      & SET BIT15,SAVE THE POINTER IN DRT0         *
9409           ;                   2. IF DRT3=<0X..XXXX>,JSB CHR;WHICH MEANS THE   *
9410           ;                      THE PGM IS IN NON-RUN STATE AND TERMINATES   *
9411           ;                      ITSELF WITH INTERRUPT CODE %A000.            *
9412           ;                   3. IF BIT13 OF INSTR. IS 1,THE CHAN WILL WAIT FOR*
9413           ;                      WHATEVER CONDITIONS ARE SELECTED PRIOR TO THE *
9414           ;                      WAIT INSTR TO CAUSE ASSERTION OF CSRQ (E.G.   *
9415           ;                      DMA COMPLETION OR SOME CONDITIONS SELECTED IN *
9416           ;                      REG#3 OF THE GIC).THIS MODE IS INVOKED TO     *
9417           ;                      FACILITATE OPERATING THE CHANNEL AS AN HP-IB  *
9418           ;                      DEVICE.THIS IS ACCOMPLISHED BY WRITING       *
9419           ;                      REG#F:=%0080+DEV.# OF THE CURRENT CHANNEL PGM.*
9420           ;                   4. IF BIT13=0,AND IF IT IS NOT CONTROLLER-IN-   *
9421           ;                      CHARGE,SAME AS 3.ABOVE,EXCEPT NO CHG IN REG F *
9422           ;                   5. IF BIT13=0,AND IS CONTROLLER-IN-CHARGE,THEN  *
9423           ;                      THE CHANNEL WILL BE SET TO ISSUE CSRQ WHEN   *
9424           ;                      PARALLEL POLL,SRQ,SIOP,OR HIOP DEV REQ OCCURS *
9425           ;                      OR IF AN HP-IB STATUS CHANGE OCCURS ON THE   *
9426           ;                      CHANNEL CONTROLLER,REG#3:=%80A0.             *
9427           ;                      THIS WILL HAPPEN PROVIDED BIT8 OF 1ST INSTR  *
9428           ;                      IS 0 AND OUTBOUND FIFO IS IDLE.              *
9429           ;                      IF NOT,BITS8,9 OF THE INSTR MAY BE USED TO   *
9430           ;                      PERFORM CRC INITIALIZATION AND COMPARISON.   *
9431           ;                   6. AFTER ALL THESE, WAIT ABOUT 2 USEC TO ALLOW  *
9432           ;                      PRIOR CSRQ TO UNASSERT BEFORE EXIT.          *
9433           ;*****************************************************************
9434  0969 WAIT 0080 XR6  ADDL      RG          UBA  SP2B ADD           DATA       %0080+DEV#;DRT0:=<POINT>-1
9436  096A      2000 XR14 ADDL      SP4A            RG   JSL  DCER          POS    PGM IN NON-RUN STATE;SP4A:=ERROR CODE
9438  096B      RH   XR10 ADD               WRS   E000      ADDL                   DRT3;MASK
9440  096C      UBB  XR4  AND                     0400 SP1B ANDL      RH     NZRO   DRT3 AS ADDR;TEST M(<POINT>-1)(13:1)
9442  096D           UBA  INC               DATA       XR30 JSB  IOW   SP3B   NZRO   DRT3=PGM STATUS+WAIT;REG#F:=%0080+DEV#
9444  096E           XR8  JSB  CPX2        EVEN       RH   JSB  CPX2           ZERO   CPX2 IF NOT CIC;CPX2 IF INSTR(13:1)=1
9446  096F                JSB  SUSP        SF3A       SP1B JSB  CRCD           POS    CHECK FIFO;CRCD IF M(<POINT>-1)<0 (F3A)
9448  0970      00A0 XR14 IORL      RG                 XR18 JSB  IOW   SP3B   NZRO   WRITE TO REG#3:=%80A0
9450  0971 CPX2 FFF6      ADDL      RH                      JSB  CPX1           ZERO   Allow about 2 us to unassert prior CSRQ
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9453           ********************************************************************
9454           *                  IDENTIFY INSTRUCTION                            *
9455           *                  I----------I----------I----------I----------I*
9456           *                  I 0  0  0  0  0  1  1  0  X  X  X  X  X  X  XI*
9457           *                  I----------------------I----------------------I*
9458           *                  I  FIRST RETURNED BYTE  I SECOND RETURNED BYTE  I*
9459           *                  I----------------------I----------------------I*
9460           ********************************************************************
9461    0972   IDY   001F       ADDL      SP4A      403F      ADDL      SP1B           SP4A:=DEV ADDR;UNLISTEN CMD
9463    0973   DSR              JSB  SUSP      UNC         ADD                        CHECK FIFO EMPTY
9465    0974                    ADD                 0004 OPB  ANDL                     CHECK REG# 2.(13:1)
9467    0975            UBB     JSB  IDYB      NZRO        XR23 JSB  *+1       ODD  NEXT LINE IF NOT IDY INSTR ELSE #IDYB IF
9469                                                                                    REG# 2.(13:1) = 1
9470    0976       003E XR12    ADDL      RG                      JSB  IOW  SP3B      ZERO  %403E,ADDRESS CHAN TO LISTEN,REG#0
9472    0977   TLK1             ADD                               ADD       SP3B PSHR      ;REG#0,ADJUST THE RAR
9474    0978                    JSB  IOW       ZERO  4040 SP4A   ADDL      RG             JSB TO SEND UNTALK, ALL DEVICES AT STANBY
9476    0979                    JSB  IOW       UNC   4060 XR0    ADDL      RG             JSB TO SEND THE SECONDARY ADDRESS
9478    097A            XR2     ADD       RG                      JSB  IOW       UNC   JSB TO RECEIVE  BYTE(S) FROM REG#0
9480    097B                    ADD                               ADD            POPR   ;POPR
9482    097C                    ADD                               ADD                   SKIP IF NF5B
9484    097D                    JSB  DSJB      F3A                ADD                   RSB  RETURN IF FROM DSRJ/CRCC;RETURN
9486    097E                    ADD           CF2                 JSL  SUSB             UNC  CF2; SUSPEND CHAN PGM FOR DATA TO BE RETRNED
9488    097F   IDYB       SP2B ADD           WRX3                JSB  IOR  SP3B        ZERO <POINT> AS ADDR;READ REG#0
9490    0980            RG      ADD  RLZ  SPOA                   JSB  IOR              UNC  SPOA:=1ST BYTE;READ 2ND BYTE
9492    0981       005E XR12    ADDL                        RG    ADD  RRZ              SET CHAN TO TALK CMD;SAVE THE 2ND BYTE
9494    0982       SPOA UBB     IOR           DATA          UBA   JSB  IOW  RG       UNC  RETURN TO M(<POINT>);ADDR CHAN TO TALK
9496    0983                    JSB  IOW       UNC           SP1B  ADD       RG   SF4B  JSB TO UNLISTEN AND REFETCH (F4B)
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 9499           ********************************************************************
 9500           *                    CLEAR INSTRUCTION                             *
 9501           *                    I-----------I-----------I-----------I-----------I*
 9502           *                    I 0  0  0  0  1  0  0  1    CONTROL BYTE        I*
 9503           *                    I-----------------------I-----------------------I*
 9504           *                    I X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  XI*
 9505           *                    I-------------------------------------------------I*
 9506           *                    THIS INSTRUCTION COMMANDS THE DEVICE TO RESET ITS*
 9507           *                    HARDWARE TO A PRE-DEFINED STATE BIT 15 OF THE    *
 9508           *                    INSTRUCTION IS THE HP-IB PARITY ERROR HANDSHAKE  *
 9509           *                    LOCKUP(0=NO LOCKUP,1=ENABLED).WHEN ENABLED, THE  *
 9510           *                    DEVICE WILL NOT ACCEPT ANY HP-IB CMD ON WHICH IT *
 9511           *                    SEES EVEN PARITY.WILL CAUSE HP-IB TO FREEZE AND  *
 9512           *                    TIMEOUT WILL OCCUR.THE SEQUENCE OF CLEAR IS:     *
 9513           *                    1. CHECK OUTBOUND FIFO ROOM AVAILABLE            *
 9514           *                    2. ADDRESS THE CHANNEL TO TALK                   *
 9515           *                    3. ADDRESS THE DEVICE TO LISTEN                  *
 9516           *                    4. ISSUE THE SECONDARY ADDRESS(MEANING,USE THE   *
 9517           *                       FOLLOWING DATA BYTE TO MODIFY THE DEVICE CLEAR*
 9518           *                    5. SEND THE CONTROL DATA BYTE TO DEVICE          *
 9519           *                    6. ISSUE THE SELECTED DEVICE CLEAR               *
 9520           *                    7. UNLISTEN                                      *
 9521           *                    8. REFETCH NEXT INSTRUCTION.                     *
 9522           ********************************************************************
 9523     0984  CLER 0070 XR12 ADDL      SP0A                 JSB  SUSP SP1B       ZERO  SEC ADDR %4070;CHECK OUTBOUND FIFO
 9525     0985       001F SP4A ADDL      RG                   JSB  IOW  SP3B       ZERO  JSB TO SEND MESSAGE-ADDRESS CHAN TO TALK
 9527     0986  LSTA           JSB  IOW       ZERO  4020 XR26 ADDL      RG             JSB TO ADDRESS DEVICE TO LISTEN
 9529     0987       SP0A      ADD       RG                   JSB  IOW                 UNC  JSB TO SEND SECONDARY ADDRESS
 9531     0988                 JSB  IOW       UNC   8000 RH   ADDL      RG             JSB TO SEND CONTROL DATA BYTE TO DEVICE
 9533     0989                 JSB  IOW       UNC   4004 SP1B ADDL      RG             JSB TO SEND SELECTED DEVICE CLEAR
 9535     098A  FIUL           JSB  IOW       UNC        SP4A ADD       RG   SF4B      SEND UNLISTEN CMD& REFETCH
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9538            ********************************************************************
9539            *                  DEVICE-SPECIFIED RELATIVE JUMP                  *
9540            *                  I----------I----------I----------I----------I*
9541            *                  I 0  0  0  0  1  0  1  1   MAXIMUM RESPONSE (N) I*
9542            *                  I----------I----------I----------I----------I*
9543            *                  I X  X  X  X  X  X  X  X   RETURNED BYTE       I*
9544            *                  I----------I----------I----------I----------I*
9545            *                  I    RELATIVE DISPLACEMENT FOR BYTE = 0        I*
9546            *                  I----------I----------I----------I----------I*
9547            *                  I    RELATIVE DISPLACEMENT FOR BYTE = 1        I*
9548            *                  I----------I----------I----------I----------I*
9549            *                  I                                             I*
9550            *                  I                                             I*
9551            *                  I----------I----------I----------I----------I*
9552            *                  I    RELATIVE DISPLACEMENT FOR BYTE = N        I*
9553            *                  I----------I----------I----------I----------I*
9554            *                  NOTE: IF RETURNED BYTE > N, THEN A DISPLACEMENT  *
9555            *                        OF 0 IS ASSUMED.                          *
9556            *                  THIS INSTRUCTION HAS A VARIABLE NUMBER OF WORDS  *
9557            *                  & THE LEFT BYTE OF THE 1ST WD IS THE MAX RESPONSE*
9558            *                      & THE RETURNED BYTE IN THE 2ND WD(SP4A) IS   *
9559            *                  READ FROM REG#0;THE SEQUENCE IS AS FOLLOWS.      *
9560            *                  1. JSB DSRJ IF NOT FROM SUSPEND;DSJB IF FROM     *
9561            *                     SUSPEND(WHEN F2=1)                            *
9562            *                  2. IF DSRJ,FIRST CHECK THE FIFO(SUSP ROUTINE),   *
9563            *                     THEN,ADDRESS THE DEVICE TO TALK ROUTINE(TLK), *
9564            *                  3. THEN,ONE DATA BYTE IS RECEIVED FROM THE DEV   *
9565            *                     (READ REG#0), AND STORED INTO THE RETURNED    *
9566            *                     BYTE FIELD OF THE INSTRUCTION IN MEMORY       *
9567            *                  4. THE ENTRY IN THE JUMP DISPLACEMENT TABLE IS   *
9568            *                     FOUND,WHICH CORRESPONDS TO THE RETURNED DATA  *
9569            *                     BYTE, AND ADDS THAT 'REL DSP' WORD TO THE     *
9570            *                     ADDRESS OF THE CHAN INSTR IMMEDIATELY         *
9571            *                     FOLLOWING THE(DSJ). THE RESULT IS THE NEW ADDR*
9572            *                     WHICH WILL BE EXECUTED NEXT.                  *
9573            *                     HOWEVER,IF THE MAX RESPONSE IS LESS THAN THE  *
9574            *                     RETURNED BYTE, THE INSTRUCTION IMMEDIATELY    *
9575            *                     AFTER THE THE DSJ WILL BE  DONE NEXT.         *
9576            *                  5. JSB TO UNLISTEN,SP2B:=NEW CP POINTER;REFETCH. *
9577            ********************************************************************
9578  098B  DSRJ       UBB  ADD       SP4A SF3A  0010      ADDL      XR0             SP4A:=DEV#;SEC ADDR
9580  098C            JSB  DSR       NF2                   ADD       SP3B CF5B       DSR IF NF2;ELSE DSJB,REG#0
9582  098D  DSJB       SP2B ADD            WRX3            JSB  IOR            UNC   <POINT> AS ADDR;JSB TO READ REG#0
9584  098E       RG         ADD  RRZ  SP4A DATA  405F      ADDL      RG              RETURNED BYTE TO <POINT>;UNTALK CMD
9586  098F       UBA  SP2B INC            ROX3            JSB  IOW            UNC   READ <REL DSPL>;SEND UNTALK CMD
9588  0990       RH   SP4A SUB            NEG   403F      ADDL      RG              MAX RESPONSE>RETURNED BYTE? UNLISTEN CMD
9590  0991       RREG OPA  ADD       RH                  JSB  IOW            UNC   NO,RH:=M(<POINT>+1+REL DSP);JSB UNLISTEN
9592  0992                 ADD                 RH   SP2B JSBI FCH1 SP2B       NZRO  ,<POINT>:=<POINT>+1+N+REL DSP
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9595            ***************************************************************
9596            *                    INTERRUPT INSTRUCTION                    *
9597            *                    I-----------I-----------I---I-------------I-----I*
9598            *                    I 0  0  0  0  0  0  0  1 R/H X  X  X  X  X  CPVAI*
9599            *                    I-----------I-----------I---I-------------I-----I*
9600            *                    I 0  0  0  0         INTERRUPT CODE            I*
9601            *                    I-----------I-----------------------------------I*
9602            *                    THIS INSTRUCTION WILL CAUSE THE CHANNEL TO ASSERT*
9603            *                    AN EXTERNAL INTERRUPT REQUEST TO THE CPU ON      *
9604            *                    BEHALF OF THE DEVICE.                           *
9605            *                    1ST WD: BIT8 INDICATES WHETHER THIS INTERRRUPT   *
9606            *                            WILL CAUSE THE CHANNEL PGM TO HALT OR    *
9607            *                            CHANNEL PGM TO CONTINUE AFTER THE        *
9608            *                            INTERRUPT REQUEST IS INITIATED.          *
9609            *                            BITS14-15 INDICATES CPVA LOCATION FIELD, *
9610            *                            THE CHANNEL PGM CAN SPECIFY INTO WHICH   *
9611            *                            OF THE RESERVED FOUR WORDS THE INTERRUPT *
9612            *                            CODE WILL BE PLACED.                     *
9613            *                    2ND WD: BITS4-15 INDICATES THE INTERRUPT CODE.   *
9614            *                            TO RELAY INFORMATION TO SOFTWARE ABOUT   *
9615            *                            WHICH TYPE OF INTERRUPTS                 *
9616            *                            WHEN THE INTERRUPT CODE IS WRITTEN INTO  *
9617            *                            THE CPVA. BITS0-3 WILL SET TO 1000.      *
9618            *                    THE SEQUENCE OF THIS INSTRUCTION IS AS FOLLOWS:  *
9619            *                    1. READ THE CPVA ADDRESS(DRT WORD 1),            *
9620            *                       SAVE INT CODE (WD 2) ,COMPLIMENT IT IF        *
9621            *                       FROM CRC MISMATCH.                            *
9622            *                    2. IF WD 1 BIT8=1,JSB TO CHECK THE FIFO;SAVE     *
9623            *                       THE <POINT>+1 IN DRTADR0.                     *
9624            *                    3. IF WD 1 BIT8=0,SET F4B.                       *
9625            ***************************************************************
9626  0993  INTH      SP1B ADD            HBF2  RREG UBB  XOR     - RH   CF5B        SET F2 IF INT/HLT;ERR CODE
9628  0994               JSB  SUSP        NF2         XR23 JSBI INL2           ZERO  CHECK FIFO IF INT/H;INL2 IF FROM CRCC
9630  0995  INL2    XR0  ADD                                ADD                     CPVA ADDR;
9632  0996      UBA  XR3  ADD            WRX3        XR27 ADD            WRS  NF2   DRT1+CPVA COUNTS AS ADDR;DRT0 AS ADDR
9634  0997      RH   XR14 ADD            DATA        XR30 ADD            SF4B F2    %8000+ERROR CODE;SKIP IF INT/RUN,F4 FOR FCH1
9636  0998           UBB  ADD                        SP2B INC            DATA       ;DRT0:=<POINT>+1
9638            ***************************************************************
9639            *                    4. PLACE THE INTERRUPT CODE+%8000 INTO THE       *
9640            *                       CPVA LOCATION SPECIFIED BY WORD1 (14:2)       *
9641            *                       ADDED TO THE CPVA ADDRRESS OF DRTADR1.        *
9642            *                    SET INTERRUPT REQUEST                            *
9643            *                    BY WRITING INTO REG#C:=%0008+DEV#.               *
9644            *                    GO FETCH IF IT IS INTERRUPT/RUN,SP2B:=CP POINTER *
9645            ***************************************************************
9646  0999       0008 XR6  ADDL      RG          UBA  XR18 JSBS IOW  SP3B       NZRO  REG#C:=%0008+DEV#
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9649            ********************************************************************
9650            *                  NOW IT IS THE INTERRUPT/HALT ROUTINE,READY TO   *
9651            *                  CLEAN THE CHANNEL TO ABORT THE CURRENT CP CLEANLY*
9652            *                  1. READ THE REG#B,AND MASK BITS9-11 +DEV#,      *
9653            *                  2. &WRITE IT BACK TO REG#F,& SET REG#3:=%7FFF,  *
9654            *                     ENABLING PHI INTERRUPT.                      *
9655            *                  3. JSB TO CHRC IF IT IS CONTROLLER IN CHARGE,   *
9656            *                     ELSE CONTINUE.                               *
9657            ********************************************************************
9658    099A    CHRA            JSB  IOR       MEDJ  0B00      ADDL      SP3B           READ REG#B
9660    099B         0070 RG    ANDL      SP0A       0300      ADDL      SP3B           MASK REG#B(9:3);REG#3
9662    099C                    CAD  LSR  RG   CF2                 JSB  IOW        UNC   WRITE TO REG#3:=%7FFF;
9664    099D         SP0A XR6   IOR       RG                XR30 JSB  IOW  SP3B   NZRO  WRT REG#F:=REG#B(9:3)+DEV#
9666    099E              XR8   JSB  CHRC      ODD          XR17 ADD       SP3B CF5B      JSB IF CIC;CF5B BEFORE CLFF,REG#2(CHRC)
9668            ********************************************************************
9669            *                  FOR HP-IB DEVICE(NON CONTROLLER IN CHARGE)      *
9670            *                  CLEAR DEVICE REQUEST BITS AND KEEP PARITY FREEZE,*
9671            *                  BY WRITING REG#6:=(BIT 9,PARITY FREEZE THE SAME).*
9672            ********************************************************************
9673    099F                    JSB  IOR       MEDJ       XR21 ADD       SP3B SF5B      READ REG#6;SF5B FOR RMW RETURN
9675    09A0         0040 RG    ANDL      RG                      JSB  IOW        UNC   KEEP ONLY PARITY,CLEAR DEV REQUESTS;WRT
9677            ********************************************************************
9678            *                  HALT SUBROUTINE                                 *
9679            *                  THIS SUBROUTINE SERVES BOTH CIC AND NON-CIC     *
9680            *                  1. ISSUE THE HIOP CMD WITH IT DATA BIT12=1,&    *
9681            *                     THE DEV# IN BITS 13-15.                      *
9682            *                  2. SAVE DRT3(2:1) BACK TO DRT3.                 *
9683            ********************************************************************
9684    09A1    HLTP 0008 XR6   ADDL      XR2        6000      ADDL      SP3B           %0008+DEV#SEND HIOP CMD WITH BIT12=1
9686    09A2                    ADD                            UBA  JSB  IOW  RG   UNC   SEND HIOP;%0008+DEV#
9688    09A3         2000 XR4   ANDL                           XR28 ADD            WRS   MASK DRT3(2:1),DRT3 AS ADDR
9690    09A4              XR8   ADD                 EVEN  UBA       ADD            DATA  SKIP IF NON-CIC;SAVE DRT3(2:1)
9692    09A5                    JSB  RMW       CF1   0400      ADDL      SP3B           RMW IF CIC;REG#4 FOR RMW
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9695            ********************************************************************************
9696            *                    ROUTINE FOR EXITING THE CHANNEL PGM.                      *
9697            *                    WRITE TO IMBI TO ENABLE THE CSRQ/IRQ MASKS.               *
9698            ********************************************************************************
9699      09A6  CPEX 0003 XR15 ANDL      RH          1000      ADDL      BKX5                  RH:=LOAD AND DUMP BIT; BKX5:=IMBI REGISTER
9701      09A7       DC00      ADDL      RG                    INC  LSL        CF2             RG:=MASK TO ENABLE CSRQ ; UBB:=2,F2:=0
9703      09A8       FFF8 XR9  ADDL                      UBB   REPC                            UBA:=MOD#-8; UBB:=1
9705      09A9       RREG UBA  ADD             HBF2      UBB   CSR             CF1  F2         REPEAT UNTIL MOD#<0; SHIFT DIV MASK RIGHT
9707      09AA       UBB  XR15 AND             NZRO      XR15  ADD       BNKS                  RRGA=DIV MASK FOR THIS MOD#, SKIP IF DIV
9709      09AB            RG   ADD       RG   F1HB              JSB  IOR  SP3B       ZERO    RG:=%5C00 TO ENABLE IRQ AND CSRQ IF NO DIV
9711                                                                                          ;SEND MESSAGE
9712      09AC       1000 STA  ANDL                      XR21  ADD  LSL        BUSC            UBA:=RIGHT STACKOP BIT; GO UNBUSY
9714      09AD            UBA  JSZ  TSRT        UNC       RH   JSL  CLLP             NZRO    UBA := right stackop bit, restart instr
9716                                                                                            thru measurement code;  Return if from
9717                                                                                            coldload or dump w/o measurement
9718      09AE  CPX1            JSB  CPEX        UNC       RH   JSBI CPX1 RH         NZRO    WAIT TO FINISH THE BUSOP
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9721            ************************************************************
9722            *                  IF IT IS THE CIC,CLEAR THE PHI INTERRUPT,REG#2=-1*
9723            *                  RESETTING PHI INTERRUPT FLAGS.                   *
9724            ************************************************************
9725    09AF    CHRC            CAD       RG                    JSB  IOW  XR9       ZERO  REG#2=%FFFF;XR9:=0,SLOW JUMP
9727            ************************************************************
9728            *                  CLEAR FIFO SUBROUTINE                            *
9729            *                  THIS ROUTINE CLEARS THE  OUTBOUND                *
9730            *                  FIFOS ON THE GIC (REG#0) TAKING INTO ACCOUNT THE *
9731            *                  WORKING OF THE HP-IB INTERFACE.  THIS ROUTINE IS *
9732            *                  USED ONLY WHEN THE CHANNEL IS CIC                *
9733            *                  FIRST, IT CHECK THE REG#2 BIT 14 (OUTBOUND FIFO) *
9734            *                  IF BIT14=1, DO UNADDRESS HP-IB ROUTINE.          *
9735            *                  ELSE JSB IFC . THE REASON TO DO HP-IB ROUTINE IS *
9736            *                  TO ASSURE THAT THE HP-IB DATA IS NOT MISTAKEN FOR*
9737            *                  COMMANDS WHEN CLEARING THE OUTBOUND FIFO WHEN    *
9738            *                  IFC IS NOT USED.                                 *
9739            *                  IFC IS ASSERTED IF THE OUTBOUND FIFO IS NOT EMPTY*
9740            ************************************************************
9741    09B0    CLFF            JSB  IOR       MEDJ  0200       ADDL       SP3B
9743    09B1        005F XR12   ADDL       RG              RG   ADD  LSR        LBF5
9745    09B2                    JSB  IOW       MEDJ            JSB  IFCC SP3B       NF5B  JSB TO WRITE REG#0 UNTALK;IFCC IF BIT14=0
9747    09B3        003F XR12   ADDL       RG              RG   JSB  IOW          UNC   UNLISTEN
9749    09B4        0060        ADDL       RG              XR21 JSB  IOW  SP3B       NZRO  WRITE TO REG#6:=REN,PARITY FREEZE
9751    09B5    IFCC            ADD                             JSB  IFC          NF5B  IFC IF REG#2(14.1) =0
9753            ************************************************************
9754            *                  THIS ROUTINE CHECKS THE INBOUND FIFO,            *
9755            *                  AND CLEARS THE INBOUND FIFO BY CHECKING REG#2    *
9756            *                  BIT13 (=1,BYTE STILL AVAILABLE,=0 IF EMPTY).     *
9757            *                  KEEP READING REG#0 UNTIL THE BIT13=0.            *
9758            *                  IT IS STILL CONTROLLER IN CHARGE,THEN JSB TO     *
9759            *                  HLTP(TO SEND HIOP CMD), AND CLEAR CORRESPONGING  *
9760            *                  BIT# FOR THE DEVICE ON REG#4,PARALLEL POLL MASK  *
9761            *                  REGISTER,& SERIAL POLL DISABLED(REG#0=%4019)     *
9762            ************************************************************
9763    09B6    CLIF            JSB  IOR       MEDJ  0200       ADDL       SP3B              READ REG#2
9765    09B7        0004 RG     ANDL                            ADD        SP3B CF5B         MASK REG#2(13:1),BYTE AVAILABLE?;REG#0
9767    09B8             UBA    ADD                ZERO     UBA  JSB  IOR          NZRO  SKIP IF ZERO;ELSE READ REG#0 AGAIN
9769    09B9                    JSB  CLIF      UNC   4019       ADDL       RG            JSB TO CHECK AGAIN;FOR CHRC,WRT TO REG#3
9771    09BA                    JSB  CHRX      F3A       XR9    JSL  RTCL         ODD   CHRX IF CHRS;RTCL FOR READ TERMINATION
9773    09BB                    JSB  IOW       MEDJ            ADD        SP3B SF5B         WRITE TO REG#0;INT1 IF FROM CRCC.
9775    09BC             OPA    JSB  INT1      CF2             JSB  HLTP         NF1   JSB HLTP,BIT11=0 TO CLE,FOR RMW.
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9778           ********************************************************************
9779           *                    THIS ROUTINE ASSERTS IFC ON THE HP-IB FOR ABOUT  *
9780           *                    100 USEC TO CLEAR THE BUS.                       *
9781           *                    1. READ THE REG#6,HP-IB CONTROL REGISTER.        *
9782           *                    2. MASK OFF THE PARITY FREEZE & REMOTE ENA BITS  *
9783           *                    3. ASSERT REMOTE ENABLE(IF DEV IS LOCAL) AND SET *
9784           *                       IFC VALUE,SO THAT THE HP-IB INTERFACE CLEAR   *
9785           *                       LINE IS ASSERTED.                             *
9786           *                    4. SEND THE REG#6 THE ABOVE CONTROL BITS, & WAIT *
9787           *                       ABOUT 100 USEC TO CLEAR THE BUS.              *
9788           *                    5. CLEAR THE OUTBOUND FIFO BY ASSERTING BITS11,  *
9789           *                       15=11 ON THE REG#6. THIS WILL TERMINATE ANY   *
9790           *                       HANDSHAKES IN PROGRESS IN THE PHI WITH THE    *
9791           *                       HP-IB.                                        *
9792           *                    NOTE: IFC MUST BE ASSERTED FIRST BEFORE CLEARING *
9793           *                    OUTBOUND FIFO, DUE TO THE 'ASYNCH TAKE CONTROL'  *
9794           *                    SEQUENCE OF GIC AND ANY DATA ON THE HP-IB WITH   *
9795           *                    'DAV' LINE TRUE COULD BE INTERPRETED AS A CMD    *
9796           *                    BY DEVICES NOT INVOLVED IN THE DATA TRANSFER.    *
9797           ********************************************************************
9798   09BD    IFC            ADD                                ADD          PSHR        UPDATE RAR
9800   09BE                   JSB  IOR       MEDJ       XR21    ADD       SP3B           READ REG#6 IF OUTBOUND FIFO IS NOT IDLE
9802   09BF         0060 RG   ANDL                 FDFF         ADDL      XR1            SAVE REN,PARITY;TIMER FOR 100US
9804   09C0         0030 UBA  IORL      RG                      JSB  IOW           UNC    SET IFC & CLEAR OUTBOUND FIFO,WRT TO REG#6
9806   09C1   IFC1            JSB  IOR       MEDJ       XR1     JSBI IFC1 XR1      NZRO   CLEAR IFC & FIFO;WAIT 100USEC TO CLEAR BUS
9808   09C2                   ADD                                ADD          POPR        ADJUST RAR
9810   09C3         0011 RG   XORL      RG                      JSB  IOW           UNC    REG#6:=CLR FIFO,NO IFC,KEEP PARITY FREEZE
9812   09C4                   ADD                RSB             ADD                      RETURN
9814           ********************************************************************
9815           *                    THIS ROUTINE WILL CHECK THE OUTBOUND FIFO EMPTY, *
9816           *                    BY TESTING THE BIT14 OF REG#2.                   *
9817           *                    IF BIT NOT SET,JSB CLFF IF TIMEOUT;              *
9818           *                    IF EMPTY,SET REG#3:=%80A0;FETCH NEXT MACHINE     *
9819           *                    INSTRUCTION.                                     *
9820           ********************************************************************
9821   09C5    CHRS           JSB  IOR       MEDJ  0200           ADDL      SP3B           READ REG#2
9823   09C6             RH   INC       RH    SF3A       RG        ADD  LSR       CF5B ODD  CLFF IF COUNT(8:1)=1;SKIP IF BIT14=0,CF5B
9825   09C7             UBB  JSB  CHRS       EVEN       UBA       JSL  CHRZ           ZERO   TRY AGAIN IF BIT14=0;CHRZ IF COUNT:=0
9827   09C8   CHRX 00A0 XR14 IORL      RG                      XR18 JSB  IOW  SP3B      NZRO   WRT REG#3:=%80A0,ENA POLLS &STATUS CHANGE
9829   09C9                  JSB  CPEX       UNC                 ADD          CF5B        CPEX;CF5B
```

```
RECORD   C S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9832           ************************************************************************
9833           *                    SENDING & RECEIVING I/O MESSAGES                  *
9834           *                    THE I/O READ OR WRITE SEQUENCES ARE AS FOLLOWS    *
9835           *                    1. SEND OUT THE 1ST WD WITH BKX4 AS BANK REG,     *
9836           *                       USUALLY THE 1ST WD=REGISTER#+CHAN#.            *
9837           *                    2. SEND OUT THE 2ND WD WITH WRX4,                 *
9838           *                       USUALLY 'DON'T CARE' IN CASE OF I/O READ       *
9839           *                       AND WILL BE DATA IN CASE OF I/O WRITE.         *
9840           *                    3. SEND CMD WD(0284);SEND WORD + MOD #, AND       *
9841           *                       IGNORE BUSY.                                   *
9842           *                    4. CHECK IF MESSAGE IS SENT BY TESTING 'MESS',    *
9843           *                       IF NOT,LOOP UNTIL MESSAGE SENT OR TIMEOUT.     *
9844           *                    5. THEN CHECK IF RETURNED MESSAGE IS PENDING      *
9845           *                       BY TESTING CPX1(9:1)=1,MSGINT.                 *
9846           *                    6. CMD CODE(0C02) TO RECEIVE 1ST WD.              *
9847           *                       IT WILL ALWAYS RETURN MESSAGE WHETHER OR NOT   *
9848           *                       IT IS I/O READ OR WRITE.(0C02) MEANS TO REC    *
9849           *                       THE 'FROM' CODE CMD,AND GO BUSY.               *
9850           *                    7. CLEAR THE MESSAGE PENDING BUFFER BY SENDING    *
9851           *                       CCPX WITH !0090;                               *
9852           *                    8. CHECK IF THE 'FROM' CORRESPONDING TO THE       *
9853           *                       'TO' MOD#                                      *
9854           *                    9. IF FROM CORRECT MOD#, CMD CODE(0402) TO REC    *
9855           *                       THE 1ST(UPPER) WD TO OPB,AND KEEP BUSY.        *
9856           *                   10. CHECK RETURNED 1ST WD VALIDITY; SUCH AS        *
9857           *                       TIMEOUT,PARITY AND DATA NOT VALID              *
9858           *                   11. CMD CODE(0802) TO REC 2ND(LOWER) WD TO OPB     *
9859           *                       AND STORE IN RG AS READ DATA.                  *
9860           *                       IN CASE OF I/O WRITE,IT IS 'DON'T CARE'.       *
9861           *                    JSB TO FETCH NEXT INSTRUCTION IF F4B SET,ELSE     *
9862           *                    RETURN AS SUBROUTINE.                             *
9863           ************************************************************************
9864           *
9865  09CA IOW             ADD                  SP3B XR19 IOR       BKX4 PSHR UNC    ;I/O WRITE
9867  09CB IOR             INC       RG         SP3B BKX5 IOR       BKX4 PSHR        RG:=-(MESS TIMEOUT); WD1:=REG# + CHAN#, PSHR
9869  09CC      0282 XR9   IORL                       RG   ADD             WRX4        UBA := put mod# in command word; Send 2nd wd
9871  09CD      E000 XR4   ANDL      XR5          UBA       ADD             BUSC        MASK DRT3.(0:2); SEND CONTROL MESSAGE
9873  09CE LOOP      RG   JSZ  MSTO       ZERO  0600      ADDL      XR21              MSTO IF TIMEOUT; XR21:=REG# 6
9875  09CF DNVL            ADD            MSGI        RG   JSBI LOOP RG        MESS  SKIP IF MSG RECEIVED; LOOP IF MESS NOT SENT
9877  09D0                 JSB  DNVL       UNC         RG   JSZI MRTO RG        ZERO  LOOP IF NOT TIMEOUT; SUSPEND IF TIMEOUT
9879  09D1      0090       ADDL                         XR21 INC  LSL       BUSC        UBA:=MASK TO CLEAR MSGINT;
9881                                                                                   READ FROM CODE (UBB:=%0C02)
9882  09D2      0401       ADDL                 UBA         ADD             CCPX        UBA.=RCV UPPER WD;CLEAR MSGINT
9884  09D3           XR5   INC       XR5        UBA         INC             BUSC        <DRT3>+'WAIT'; TO REC 1ST WD
9886  09D4      001D       ADDL                 RREG        ADD  LSL        BUSC        MASK FOR PARITY/TIMEOUT/DNV; TO REC 2ND WD
9888  09D5      00C3 XR12  ADDL       XR11       UBA  OPB   AND       RG   POPR NZRO  %40C3; CHECK VALID RTN
9890  09D6      UBB        JSZ  IOER       NZRO        OPB  ADD       RG              IOER IF ERROR; SAVE 2ND (LOWER) WORD
9892  09D7           XR11  ADD            RSB                 JSB  FCH1 BKX3   F4B   %40C3(SUSP),RETURN;JSB IF F4B,SET MEM BANK=0
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9895            ********************************************************************
9896            *                    SUSPEND CHECK ROUTINE                         *
9897            *                    THIS ROUTINE CHECKS THAT THE OUTBOUND FIFO IS  *
9898            *                    EMPTY BEFORE ALLOWING EXECUTION OF THE CURRENT *
9899            *                    INSTRUCTION.IT IS CALLED BY EACH INSTRUCTION   *
9900            *                    WHICH WRITES TO REGISTER#0 ON THE CHANNEL.     *
9901            *                    IF THE FIFO IS NOT EMPTY, THEN THE PROGRAM IS  *
9902            *                    SUSPENDED UNTIL THE FIFO IS EMPTY.             *
9903            *                    FIFO IS CHECKED BY READING THE REG#2 BIT14.    *
9904            *                    F2 IS SET IFF SUSB IS THE ENTRY POINT,WHERE    *
9905            *                    IT IS SUSPENDED TO WAIT FOR DATA TO ARRIVE.    *
9906            ********************************************************************
9907    09D8  SUSP 00C1 XR12 ADDL      XR13                 XR27 ADD             PSHR       MASK BITS;ADJUST THE RAR,DRT3 ADDR
9909    09D9       UBB       ADD                WRS          XR17 JSB  IOR  SP3B      NZRO  WRITE TO DRT3 ADDR;READ REG#2
9911    09DA       RG   XR13 AND                ZERO  RG     UBA  AND       RG   POPR NZRO  SKIP IF ZERO;CHECK INTERRUPT BITS+FIFO BIT
9913    09DB            SP2B ADD           RH   NZRO         XR30 JSL  SUSC            UNC   SKIP IF REG#2(14:1)=0; SUSB IF ALL ZERO
9915    09DC            UBB  ADD                RSB   E000   SP2B SUBL      RG              RTN IF SUBROUTINE,REG#F(INTH);ERR CODE
9917    09DD       RG        ADD     SWAB SP4A  POS          RG   ADD       SP1B       EVEN  SKIP IF NO STATUS CHG;SKIP IF DEV CLEAR
9919    09DE       0200 RG   ADDL     RG              0100   SP2B ADDL      SP2B              ERR FOR STATUS CHG;ERR FOR DEV CLEAR
9921    09DF            SP4A ADD  LSL       POS          SP1B ADD  LSL                POS   SKIP IF NO HANDSHAKE ABORT;SKIP IF PARITY
9923    09E0       0080 RG   ADDL     RG              0400   SP2B ADDL      SP2B              ERR FOR HANDSHAKE ABORT;<POINT>
9925    09E1       RH        CAD                DATA         RH   ADD       SP2B              DRT0:=<POINT>-1;RESTORE <POINT>
9927    09E2       RG   SP2B ADD        XR13 CF1                  JSB  CHR             ZERO  ERROR CODE;JSB TO CHR
```

```
RECORD   C S        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

9930           ************************************************************************
9931           *                    CRC INITIALIZE INSTRUCTION                        *
9932           *                    I---------------------I---------------------I*
9933           *                    I 0  0  0  0  1  0  0  0  0  X  X  X  X  X  X  XI*
9934           *                    I---------------------I---------------------I*
9935           *                    I X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  XI*
9936           *                    I---------------------I---------------------I*
9937           *                    IT WILL FORCE THE CRC GENERATORS IN THE CHANNEL  *
9938           *                    CONTROLLER AND DEVICE TO A KNOWN INITIAL STATE.  *
9939           *                    THEN, THE GENERATORS WILL OPERATE ON ANY DATA    *
9940           *                    TRANSFERRED BETWEEN THESE DEVICES.               *
9941           *                    1 IT WILL CAUSE THE CHANNEL CONTROLLER TO BE     *
9942           *                      EXAMMINED TO CHECK WHETHER THE CONTROLLER HAS  *
9943           *                      THE CAPABILITY(REG#7(0.1)) BEFORE PROCEEDING.  *
9944           *                    2.IF NOT CAPABLE, THEN UNLISTEN & REFETCH.       *
9945           *                    3.IF IT'S,THEN I. ADDRESS THE DEV TO LISTEN,     *
9946           *                                   II. ISSUE THE SEC ADDRESS,        *
9947           *                                   III. ADDRESS THE CHAN TO LISTEN,  *
9948           *                                   IV. ISSUE THE SEC ADDRESS,        *
9949           *                                   V. THEN UNLISTEN,& REFETCH        *
9950           ************************************************************************
9951   09E3    CRCI C03F      ADDL     RH                    JSB  IOR          UNC   ADDR CHAN TO LISTEN;READ REG#7,DEV#
9953   09E4         RG       JSB  FIUL      POS             ADD       SP3B           FIUL IF HAS NO CRC,REG#0 FOR FIUL
9955   09E5         JSB  SUSP      UNC   006D               ADDL      SP1B           SUSP TO CHECK FIFO;SEC ADDR
9957   09E6         0071 XR12 ADDL      SP0A                JSB  LSTA SP3B      UNC   SEND SEC ADDR;SEND THE HP-IB CMDS
```

```
RECORD   C S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

 9960           ********************************************************************
 9961           *                    CRC COMPARE INSTRUCTION                       *
 9962           *                    I----------------------I--I--I-----------I-----I*
 9963           *                    I 0  0  0  1  0  0  0  1 RH IN X  X  X  X  CPVAI*
 9964           *                    I-----------I-----------I--I--I-----------I-----I*
 9965           *                    I X  X  X  X          INTERRUPT CODE           I*
 9966           *                    I-----------I-------------------------------I*
 9967           *                    THIS INSTRUCTION OBTAINS THE TWO-BYTE CRC FROM   *
 9968           *                    THE DEVICE AND CHANNEL AND COMPARES THEM.        *
 9969           *                    IF THEY MATCH, THE NEXT INSTRUCTION IS EXECUTED  *
 9970           *                    (BIT9 OF THE INSTR=0) OR INT/HALT INSTR. IS DONE *
 9971           *                    NEXT (BIT9=1). IF CRC BYTES DO NOT MATCH, THEN   *
 9972           *                    AN INT/HLT INSTR IS DONE, TAKING THE 1'S         *
 9973           *                    COMPLEMENT OF THE SPECIFIED INTERRUPT CODE IN    *
 9974           *                    2ND INSTR. WORD.                                 *
 9975           *                    CRC COMPARE AND CRC INITIALIZE SHOULD BE USED    *
 9976           *                    TOGETHER AND SHOULD NOT BE SEPARATED WITH        *
 9977           *                    INTERRUPT/HALT OR WAIT INSTRUCTIONS.             *
 9978           ********************************************************************
 9979    09E7   CRCC           CAD  LSL  SP4A                 JSB  IOR  XR9        UNC   -2;READ REG#7,SET XR9:=0
 9981    09E8                  JSB  SUSP       UNC   RG       JSB  CRC4           POS   CHECK OUTBOUND FIFO;CRC4 IF REG#7(0:1)=1
 9983    09E9       0004 RG    ANDL                      SP4A ADD       XR0  SF5B        TEST REG#2(13:1);XR0=-2,F5B FOR TLK1 RETURN
 9985    09EA                  JSB  TLK1       UNC       UBA  JSB  CRC2           NZRO  ADDR CHAN TO TALK,CRC2 IF FIFO NOT EMPTY
 9987    09EB                  JSB  IOW        UNC   8002     ADDL      RG             GO TO RECEIVE THE 2 BYTES FROM CHAN
 9989    09EC                  JSB  TLK1       UNC   FFE0 XR26 IORL      XR0            ADDR DEV TO TALK
 9991    09ED                  JSB  IOW        UNC   8002     ADDL      RG             GO TO RECEIVE THE 2 BYTES FROM DEV
 9993    09EE   CRC1           JSB  SUSP       UNC         XR0  JSBI CRC1 XR0       NZRO  WAIT 15 USEC FOR HP-IB TO SETTLE,CHECK FIFO
 9995    09EF   CRC2 005F XR12 ADDL      RG                      JSB  IOW  SP3B      ZERO  CLEAR HP-IB,TO UNTALK
 9997    09F0       003F XR12 ADDL      RG                      JSB  IOW           UNC   TO UNLISTEN
 9999    09F1                  JSB  IOR        UNC                 ADD           CF5B  F1 FOR CLIF TO INT1;READ THE 1ST CHAN BYTE
10001    09F2            RG    ADD  LLZ  SP4A SF1         RG   JSB  IOR  RH      UNC   CHECK SP4A.=1CBYTE(0:8);RE:=1CB,READ 2CB
10003    09F3            RG    ADD  RRZ  XR1              SP4A JSB  CLIF          NZRO  SAVE XR1:=2CB(8:8);CLIF IF 1CB(0:8)=0
10005    09F4                  JSB  IOR        UNC         RG   JSB  CLIF          POS   READ 1DB;CLIF IF 2CB(0:1)=0
10007    09F5       RH   RG    XOR       RH                RG   ADD               POS   COMPARE 1CB=1DB?;SKIP IF 1DV(0:1)=0
10009    09F6                  JSB  IOR        UNC   UBB   UBB  JSB  CRC4          NEG   READ 2DB;CRC4 IF 1DB(0:2)=11
10011    09F7            RG    ADD  RRZ                    RH   JSB  CLIF          NZRO  SAVE 2DB(8:8);CLIF IF 1CB#1DB
10013    09F8       UBA  XR1   JSBS CLIF       NZRO        RG   JSB  CLIF          POS   COMPARE 2CB=2DB?CLIF <>;CLIF IF 2DB(0:1)=0
10015    09F9   CRC4           ADD                     FFF0 XR23 SUBL                     ;CHECK IF FROM WAIT INSTR
10017    09FA                UBB  JSB  CHRS       ZERO        SP1B ADD  LSL             TEST 1ST INSTR(9:1);CHRS IF FROM WAIT
10019    09FB                OPA  JSB  INT1       CF1         UBB  JSB  FCH1          NEG   ELSE GO FETCH NEXT INSTR;INT1 IF BIT8=1
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10022           ************************************************************
10023           *                  WRITE RELATIVE IMMEDIATE                *
10024           *                  I----------I----------I----------I------I*
10025           *                  I 0  0  0  0  1  1  1  1     DISPLACEMENT I*
10026           *                  I----------I----------I----------I------I*
10027           *                  I                  DATA                  I*
10028           *                  I--------------------------------------I*
10029           ************************************************************
10030   09FC    WRIM FF00 RH   ADDL                RH   SP2B INC            WRX3 UNC   UBA:=%FF00+DSP IF RH<0;SP2B:=NEW DSP IF>0
10032   09FD          OPA  ADD                     UBA  SP2B INC            WRX3       M(<POINT>);SP2B:=NEW DSP IF RH<0
10034   09FE               JSB  FCH1       UNC          UBA  ADD            DATA       SEND DATA TO MEM
10036           ************************************************************
10037           *                    FOR INPUT(READ) RECORD MODE,            *
10038           *                    JSB TO DMON(DMA TURN-ON ROUTINE) IF NOT CIC,    *
10039           *                    IF CIC,THEN REG#0:=%C000,TO RECEIVE BYTES,      *
10040           ************************************************************
10041   09FF    RREC       XR0  ADD       SP0A            UBB  JSL  DMON           EVEN  RECOVER BCOUNT;DMON IF NOT CIC
10043   0A00    RRE1 C000       ADDL      RG                   JSL  IOW  SP3B      ZERO  REG#0:=%C000,TO RECEIVE BYTES
10045           ************************************************************
10046           *                    DMA TURN-ON ROUTINE                     *
10047           *                    ROUTINE:REG#A:=BYTE COUNT,              *
10048           *                            REG#9:=MEMORY ADDRESS,          *
10049           *                            REG#8:=EXTENDED MEMORY ADDRESS,         *
10050           *                            REG#B:=%0100+CRTL(8:8) IF CRTL(12:1)=1,*
10051           *                                  :=CRTL(8:8) IF CRTL(12:1)=0,      *
10052           *                                   (WITH DEV# IN BITS13-15),        *
10053           *                            DRT3:='OLD' DRT3(0:3)+%0002,            *
10054           *                            THEN, JSB CPEX TO LEAVE DMA ALONE       *
10055           *                                  TO FINISH,                        *
10056           ************************************************************
10057   0A01    DMON F000 RH   ANDL       RH                XR10 INC            ROA3       MASK THE <CRTL>;READ M(<POINT>+3),MEM ADDR
10059   0A02         UBA       ADD  SWAB RH           0800 SP1B ANDL                 ZERO  SWAP THE <CRTL>;TEST CRTL(4:1)
10061   0A03         0B00      ADDL      SP4A         0100 UBA  ADDL       RH              REG#B:%0100+<CRTL> IF CRTL(4:1)=1
10063   0A04         0800      ADDL                   0900      ADDL       SP1B            REG#8:=EXT MEM ADDR;REG#9:=MEM ADDR
10065   0A05         00FF SP1B ANDL       RG                   UBA  JSL  IOW  SP3B      NZRO  WRITE TO REG#8,**EXT ADDR IS EIGHT BITS**
10067   0A06              OPA  ADD        RG                   SP1B JSL  IOW  SP3B      NZRO  WRITE TO REG#9 THE MEM ADDR
10069   0A07         SP0A      ADD        RG                   XR25 JSL  IOW  SP3B      NZRO  REG#A:=BCOUNT
10071   0A08         RH   XR6  ADD        RG                   XR27 ADD                CF5B  NEW <CRTL>+DEV#;DRT0 AS ADDR,CF5B FROM TLK1
10073   0A09         UBB  XR10 ADD             WRS             SP4A JSL  IOW  SP3B      NZRO  DRT3 ADDR;WRITE TO REG#B
10075   0A0A              XR5  INC             DATA                 JSL  CPEX           UNC   SAVE DRT3(0:3,14:1=1);LEAVE ALONE TO FINISH
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10078           ********************************************************************
10079           *                  EXECUTE DMA INSTRUCTION (INITIATION)            *
10080           *                  I----------I----------I----------I----------I*
10081           *                  I 0  0  0  0  1  1  1  0  X  X  X  X  X  X  X  X I*
10082           *                  I-----------------------I----------------------I*
10083           *                  I          BYTE COUNT  /  RESIDUE COUNT        I*
10084           *                  I-----------------------I----------------------I*
10085           *                  I TERMINATE DISPLACEMENT  BURST LENGTH (0 =256) I*
10086           *                  I-----------------------I----------------------I*
10087           *                  IRB LR SP RW SA UD  X  X  EXTENDED MEMORY ADDR. I*
10088           *                  I--I--I--I--I--I--I-----I----------------------I*
10089           *                  I       MEMORY ADDRESS  /  RESIDUE ADDRESS     I*
10090           *                  I-----------------------------------------------I*
10091           *                  BIT ABBREVIATIONS:                              *
10092           *                    RB : 0 = "RECORD MODE", 1 = "BURST MODE".     *
10093           *                    LR : 0 = "START ON LEFT BYTE", 1 = "START ON  *
10094           *                         RIGHT BYTE".                             *
10095           *                    SP : READ : 1 = "TERMINATE ON ASCII LINE-FEED *
10096           *                                    (GIC ONLY)".                  *
10097           *                         WRITE: 1 = "DO NOT TAG HP-IB END MESSAGE *
10098           *                                    TO LAST BYTE".                *
10099           *                    RW : 0 = "READ(INPUT)", 1 = "WRITE(OUTPUT)".  *
10100           *                    SA : 1 = "SEND/REC. ALL DATA WITH SINGLE MEMORY*
10101           *                         ADDRESS".                                *
10102           *                    UD : 1 = "DO NOT UPDATE INSTRUCTION WORDS AFTER*
10103           *                         EXECUTION".                              *
10104           *                    JSB TO WRXD IF CRTL(3:1)=1,FOR OUTPUT(WRITE). *
10105           *                    JSB TO RREC IF CRTL(0:1)=0,RECODE MODE.       *
10106           *                    CONTINUE ON IF BURST MODE.                    *
10107           ********************************************************************
10108  0A0B  XDMA E002      ADDL     SP4A       UBA       INC       XR10 ROX3        ERROR CODE;READ <POINT>+2
10110  0A0C                 CAD  RRZ       SF1       SP2B ADD       RG               MASK BITS;<POINT>
10112  0A0D       UBA  OPA  AND      XR1  NZRO      XR27 ADD            WRS          SKIP IF BURST<>0;DRTO ADDR
10114  0A0E                 INC  SWAB XR1       RG        CAD            DATA         SET BURST:=%0100 IF 0;<DRTO>:=<POINT>
10116  0A0F       SPOA      ADD       XR0                 OPB  ADD       SP1B         SAVE BCOUNT;<CRTL>
10118  0A10                 ADD                           UBA  JSL  DCER         ZERO  CHR IF DATA CHAIN=0;
10120  0A11  DMA       XR1  ADD       RG         1000      ADDL                       BURST LENGTH;MASK
10122  0A12       SPOA XR1  SUB       SP4A CRRY  UBB  SP1B AND       XR1          NZRO  SKIP IF BCOUNT>=BURST;SKIP IF WRITE
10124  0A13       SPOA      ADD       XR1  UNC   48FF SP1B ANDL      RH           NZRO  BURST:=BCOUNT;<CRTL> FOR READ
10126  0A14             XR1  ADD       SPOA                XR1  JSB  WBXD         NZRO  BCOUNT:=BURST;WBXD IF 'WRITE'
10128           ********************************************************************
10129           *                  FOR INPUT(READ) BURST MODE WITH F2=1,           *
10130           *                  WRITE TO REG#0:=<BURST> IF CRTL(2:1)=1,         *
10131           *                                          TERMINATE ON LINE-FEED.*
10132           *                                 =%8000+<BURST> IF CRTL(2:1)=0.   *
10133           ********************************************************************
10134  0A15             XR8  ADD                           SP1B ADD                   NEG   CIC REG;SKIP IF CRTL(0:1)=1
10136  0A16       UBB  UBB  ADD  LSL         POS          UBA  JSL  RREC         UNC   UBB(0:1)=CRTL(2:1),LF;RREC IF RECORD,CIC
10138  0A17             XR1  ADD       RG    UNC          UBB  JSL  DMON         EVEN  REC UNTIL LINE-FEED DETECTED;DMON IF NON-CIC
10140  0A18       UBA  XR14 IOR       RG                      JSL  IOW  SP3B         ZERO  REC BYTES UNTIL END
10142  0A19                 ADD                               JSL  DMON         UNC   JSB DMON FOR DMA TURN-ON ROUTINE
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10145           ************************************************************************
10146           *                    OUTPUT(WRITE) ROUTINE                             *
10147           *                    JSB WRXD IF RECORD MODE,OR BCOUNT>BURST LENGTH.   *
10148           *                    FOR BURST MODE:1.READ REG#6,HP-IB CONTROL BITS.   *
10149           *                                   2.WRITE TO REG#6:=REG#6'AND'%0060,*
10150           *                                     +%0022                          *
10151           *                                   3.WRITE TO REG#3:=%8002,SETUP PHI,*
10152           *                                   4.IF DATA CHAIN=0,RH:=<CRTL>+%0010*
10153           *                                     <CRTL>=EXTENDED MEM ADDR+L/R+S/P*
10154           *                                           +R/W+S/A.                 *
10155           *                                   5.THEN JSB TO DMON.               *
10156           *                    FOR RECORD MODE: IDENTICAL TO BURST MODE,EXCEPT  *
10157           *                                     <CRTL>=EXTENDED MEM ADDR+L/R+R/W*
10158           *                                           +S/A.                     *
10159           ************************************************************************
10160   0A1A  WBXD        SP1B JSB  WBX1       POS   58FF SP1B ANDL      RH              WRXD IF 'RECORD' MODE;<CRTL>
10162   0A1B        RG    XR0  JSBS WRX1       CRRY  2000 SP1B ANDL      RG              WRX1 IF BCOUNT<=BURST LENGTH;TEST CRTL(2:1)
10164   0A1C                   ADD                     78FF SP1B ANDL                    WRX1 IF BC=BL;<CRTL>
10166   0A1D        UBB        ADD       RH    UNC           ADD                       <CRTL> FOR BC>BL;
10168   0A1E  WBX1        XR0  ADD       SPOA                XR21 JSL  IOR   SP3B      NZRO  RECOVER BCOUNT;READ REG#6
10170   0A1F        1000  RH   IORL      RH                  BKX6 ADD                  ZERO  RH:=%1000+<CRTL>;SKIP IF DC:=0
10172   0A20        0060  RG   ANDL                    3000  RH   IORL      RH              MASK REG#6(PARITY FRZ+REMOTE ENABLE)
10174   0A21        0022  UBA  IORL      RG                  JSL  IOW                 UNC   SETUP DMA+REN TO REG#6;
10176   0A22        0002  XR14 ADDL      RG                  XR18 JSL  IOW   SP3B      NZRO  REG#3:=DMA CSRQ ON OUTBOUND FIFO IDLE
10178   0A23                   ADD                           JSL  DMON                UNC   ;JSB DMON
10180   0A24  WRX1       RG   JSB  WRXE       ZERO          JSB  WBX1                F1    WRXE IF CRTL(10:1):=0;WBX1 IF XDMA INSTR
10182   0A25  WRXD            JSB  WBX1       UNC   58FF SP1B ANDL      RH              <CRTL>;
10184   0A26  WRXE            ADD                           ADD       BKX6            SET DC:=0
10186   0A27                   JSB  WBX1       UNC   78FF SP1B ANDL      RH              WBX1 WITH <CRTL>
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10189           ********************************************************************
10190           *                DMA ROUTINE-REFETCH DMA INSTRUCTION AND           *
10191           *                CONTINUE EXECUTION                                *
10192           ********************************************************************
10193   0A28  DMRR          ADD       SP0A                JSB  DMR1          UNC   SET BCOUNT=0;JSB DMR1
10195   0A29  DMAM          CAD  LSR  RG   CF1       XR18 JSL  IOW  SP3B      NZRO  WRITE REG#3:=%7FFF
10197   0A2A        000A    ADDL      XR0            SP1B ADD  LRZ  XR9             MASK BIT;XR0(8:8):=1ST INSTR(0:8)
10199   0A2B        FFFD UBB  ADDL      SP0A      000E UBB  SUBL      RG        NZRO  TEST FOR READ,RCON,WRIT,WCON &XDMA
10201   0A2C        E001      ADDL      SP4A           SP1B ADD  LLZ  SP1B            ERROR CODE(JSB V);NO DC/MODI-XDMA
10203   0A2D        SP0A XR0  AND       XR2  ZERO      SP2B INC       SP2B ROA3       SKIP IF READ/RCON/WRIT/WCON;<POINT>,&READ
10205   0A2E              RG   JSB  V         NZRO  00F0 SP1B ANDL      SP1B            V IF NOT XDMA;MASK OFF DC BITS
10207   0A2F              OPA  JSB  CGE2      UNC   SP3B BKX7 AND       XR0  PSHR      CGE2 FOR DATA CHAIN UPDATE;MASK REG#B(5:2)
10209   0A30        SP0A  XR1  JSBS DMRR      NCRY           SP1B JSB  DMRR            POS   DMRR IF BCOUNT<BURST; DMRR IF CRTL(0:1)=0
10211   0A31  DMR1         UBA  ADD       SP0A               XR24 JSL  IOR  SP3B      NZRO  BCOUNT;READ REG#9,MEM ADDR
10213   0A32              RG   ADD       XR1                XR25 JSL  IOR  SP3B      NZRO  SAVE MEM ADDR;READ REG#A DMA BYTE COUNT
10215   0A33              SP2B ADD       SP0A WRX3  0400 SP1B ANDL                    ZERO  <POINT> AS ADDR;TEST CRTL(5:1),UPDATE?
10217   0A34        SP0A  RG   ADD       SP0A CF2           JSB  DMR2 RH            UNC   SP0A:=BCOUNT+<REG#A>;DMR2 IF UPDATE
10219   0A35        SP0A  UBA  XFRS            DATA  BFFF SP1B ANDL      RG              STORE NEW BCOUNT TO <POINT>,<CRTL>
10221   0A36        RREG  XR10 ADD             WRX3       RH   ADD  RLZ  RH              <POINT>+3 ADDR;MASKED REG#B(9:1)
10223   0A37              XR1  ADD             DATD           ADD                       SAVE MEM ADDR TO <POINT>+3
10225   0A38        RH    RG   ADD             DATA           ADD       RH              <POINT>+3:=MEM ADDR;RH:=0
10227   0A39  DMR2         XR2  JSB  WDT       ZERO      XR9  JSB  RDT           ODD   WDT IF WRIT/WCON;RDT IF READ/RCON,XR9(CLF1)
10229           ********************************************************************
10230           *                DMA INSTRUCTION (TERMINATION FOR XDMA)            *
10231           *                1.UPDATE REG#6,HP-IB CONTROL REG;CLEAR OUTBOUND   *
10232           *                  FIFO,RESET DMA DIRECTION,KEEP 'REN' AND PARITY. *
10233           *                2.IF WRITE(OUTPUT):                               *
10234           *                  IF 'RECORD',DO REFETCH & EXECUTE NEXT INSTR.    *
10235           *                  IF 'BURST',& BCOUNT=0,DO REFETCH & EXECUTE      *
10236           *                  NEXT INSTR.                                     *
10237           *                  IF 'BURST',& BCOUNT<>0,DO REFETCH & SKIP NEXT   *
10238           *                  INSTR.                                          *
10239           *                3.IF READ(INPUT):JSB XTIN                         *
10240           *                  IF 'RECORD',& DMST(5:2)=00,REFETCH AND EXECUTE  *
10241           *                  NEXT INSTR.                                     *
10242           *                  IF 'REOCRD',&DMST(5:2)=X1,JSB DMAB FOR ABORT.   *
10243           *                  IF 'RECORD',&DMST(5:2)=10,REFETCH AND EXECUTE   *
10244           *                  THE INSTR INDICATED BY <POINT>+TERM DSPL.       *
10245           *                  IF 'BURST', & DMST(5:2)=00,REFETCH AND EXECUTE  *
10246           *                  NEXT INSTR.                                     *
10247           *                  IF 'BURST', & DMST(5:2)=11,JSB DMAB FOR ABORT.  *
10248           *                  IF 'BURST', & DMST(5:2)=01/10,& IF BCOUNT<>0,   *
10249           *                  SKIP NEXT INSTR;HOWEVER,IF BCOUNT=0,REFETCH     *
10250           *                  AND EXECUTE THE INSTR INDICATED BY <POINT>+     *
10251           *                  TERMINATE DISPLACMENT.                          *
10252           ********************************************************************
10253   0A3A              OPA  ADD  LRZ  RH            XR21 JSL  IOR  SP3B      NZRO  RH:=TERMINATE DSPL;READ REG#6,HP-IB CONTROL
10255   0A3B        0060 RG   ANDL                1000 SP1B ANDL      SP1B            CLEAR FIFO,RESET DMA ,KEEP 'REN',PARITY
10257   0A3C        UBA       INC       RG           SP1B JSL  IOW  CTX       UNC   RETURN TO REG#6;<CRTL>
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10260           ********************************************************************
10261           *                  DMA TERMINATION FOR XDMA WRITE(OUTPUT)          *
10262           *                  CRTL(3:1)=1.                                    *
10263           ********************************************************************
10264   0A3D          SP1B JSB  XTIN      ZERO      XRO  ADD  LSL  SP1B            XTIN IF CRTL(3:1)=0;SP1B:=REG#B(5:2)
10266   0A3E       SPOA    JSB  FCHL      ZERO      CTRS JSL  FCH4          POS   FETCH IF BCOUNT=0;& IN 'RECORD' MODE
10268   0A3F               JSB  FCHL      UNC   0002 SP2B ADDL      SP2B        FETCH & SKIP NEXT INSTR IF BCOUNT<>0
10270           ********************************************************************
10271           *                  DMA TERMINATION FOR XDMA READ(INPUT)            *
10272           *                  BURST MODE.                                     *
10273           ********************************************************************
10274   0A40  XTIN     OPA  ADD             POS       BKX7 ADD  LSR  RG              SKIP IF TERM DSP>0;SHIFT RIGHT(2ND) REG#B
10276   0A41       FF00 RH  ADDL       RH        SP3B SP1B XOR       SP1B            TERM DSP IF <0;TEST DMST(5:2)
10278   0A42            RG  ADD  LSR   SP4A            SP1B ADD                     SHIFT(3RD) REG#B;
10280   0A43       0600 UBB ANDL       RG             CTRS JSB  XTIR          POS   DMST(5:2):=1 ii?XTIR IF CRTL(0:1):=0
10282   0A44            UBA  JSB  FCHL      ZERO      SP1B JSL  DMAB          ZERO  FCHL IF DMST(5:2)=00;DMAB IF DMST(5:2)=11
10284   0A45       SPOA    JSB  FCHL      NZRO  0002 SP2B ADDL      SP2B        FETCH IF BCOUNT<>0;& SKIP NEXT INSTR
10286   0A46  XT11         JSB  FCHL      UNC   RH   SP2B ADD       SP2B        FETCH NEXT INSTR AT <POINT>+TERM DSPL
10288           ********************************************************************
10289           *                  DMA TERMINATION FOR XDMA READ(INPUT)            *
10290           *                  CRTL(3:1)=0,RECORD MODE.                        *
10291           ********************************************************************
10292   0A47  XTIR     RG   JSB  FCHL      ZERO      BKX7 ADD  SWAB                FETCH IF DMST(5:2)=00;UBB(15:1)=DMST(6:1)
10294   0A48               JSB  XT11      UNC       UBB  JSL  DMAB          ODD   DMAB IF DMST(5:2):=X1;ELSE XT11
10296           ********************************************************************
10297           *                  DMA INSTRUCTION (TERMINATION FOR READ)          *
10298           *                  1.JSB TO RDT1 IF IT IS RECORD MODE.             *
10299           *                  2.BURST MODE TERMINATION:                       *
10300           *                    I.JSB TO CLFF TO CLEAR THE FIFOS,             *
10301           *                    II.JSB TO RDBE IF DMST(5:2)=01,               *
10302           *                    III.JSB TO RDEN IF DMST(5:2)=00,              *
10303           *                    IV. ELSE JSB TO DMAB WITH ERROR CODE IN SP4A. *
10304           ********************************************************************
10305   0A49  RDT      SP1B JSB  RDT1      POS       BKX7 ADD  LSR  RG              RDT1 IF RECORD;SHIFT(2) REG#B
10307   0A4A       UBB      ADD  SWAB SP4A SF2            JSL  CLFF          UNC   DMST(6):CLFF TO CLEAR FIFOS
10309   0A4B  RDTB    RG   ADD  LSR  SP4A            XRO  JSB  RDEN          ZERO  ERR CODE FOR DMAB F2(RDEN):RDEN DMST(5:2)=00
10311   0A4C               JSB  RDEN      CF2       SP4A JSL  DMAB          ODD   RDBE IF DMST(5:2):=01;DMAB IF DMST(5:2):=1X
10313           ********************************************************************
10314           *                  DMA INSTRUCTION (TERMINATION FOR READ)          *
10315           *                  FOR RECORD MODE.                                *
10316           *                  1.JSB TO RDEN IF DMST(5:2)=00,                  *
10317           *                  2.JSB TO DMAB IF DMST(5:2)=X1 WITH ERROR IN SP4A.*
10318           *                  3.JSB TO RDT3 IF DATA CHAIN<>0,                 *
10319           *                  4.ELSE JSB TO CLFF TO CLEAR FIFOS,THEN          *
10320           *                    JSB TO NEW <POINT>.                           *
10321           ********************************************************************
10322   0A4D  RDT1     RG   ADD  SWAB      SF2       XRO  JSB  RDEN          ZERO  UBA(0:1):=DMST(6:1);RDEN IF DMST(5:2):=00
10324   0A4E           RG   ADD  LSR  SP4A CF2       UBA  JSL  DMAB          NEG   DMAB IF DMST(5:2)=X1 WITH ERROR CODE
10326   0A4F           OPA  ADD  LRZ       BIT8      XR27 ADD       ROAS          SKIP IF TERM DSPL<0;READ DRTO ADDR FOR DCPU
10328   0A50       UBA      ADD       RH   UNC       BKX6 JSB  RDT3          NZRO  NEW <POINT>;RDT3 IF DC<>0
10330   0A51       FF00 UBA ADDL      RH                 JSL  CLFF          UNC   NEW <POINT> IF TD<0;CLFF TO CLEAR FIFOS
10332   0A52  RDTA         JSB  FCHL      UNC   RH   SP2B ADD       SP2B        ,REFETCH NEXT INSTR AT <POINT>+TERM DSPL
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10335            ***************************************************************
10336            *                  DMA INSTRUCTION (TERMINATION FOR READ)       *
10337            *                  TO CONTINUE THE EXECUTION BY SETTING UP NECESSARY*
10338            *                  REGISTERS FOR TRANSFER.                      *
10339            ***************************************************************
10340     0A53   RDT3 08FF XR12 ADDL      RH                    JSB  DCPU          UNC   %48FF;DATA CHAIN POINTER UPDATE
10342     0A54                JSBI DCNB       UNC         XR2  ADD       SP2B ROA3       JSB TO NEXT DATA CHAIN BLOCK;NEW <POINT>
10344     0A55        RH   UBB  AND       RH                    JSL  DMON          ZERO  MASKED <CRTL>;DMON TO SET UP FOR XFER(SLOWJ)
10346     0A56   RTCL            ADD                           ADD       XR9             CLEAR XR9 IF NOT FROM WAIT
10348     0A57                JSB  RDTA       UNC         BKX7 JSB  RDTB RG          F2    RETURN FROM CLFF ROUTINE;RG:=SHIFTED REG#B
10350            ***************************************************************
10351            *                  DMA INSTRUCTION (TERMINATION FOR WRITE)      *
10352            ***************************************************************
10353     0A58   WDT       SP1B JSB  WDTB       NEG         XR27 ADD            ROAS      WDTB IF BURST;READ DRTO FOR DCPU
10355     0A59                JSB  WUNL       UNC         BKX6 JSB  DCPU          NZRO  DCPU IF DC<>0;ELSE WUNL
10357     0A5A                JSBI DCNB       UNC         XR2  ADD       SP2B ROA3       DCNB FOR THE NEXT CHAIN,+1;READ NEW <POINT>
10359     0A5B           OPA  ADD        XR0        00F0 OPB  ANDL      BKX6            SAVE THE BCOUNT FOR WRXD;NEW DATA CHAIN
10361     0A5C                ADD                           JSB  WRXD          UNC     WRXD TO SETUP THE XFER
10363            ***************************************************************
10364            *                  DATA CHAIN ERROR WITH ERROR CODE:=%E002      *
10365            ***************************************************************
10366     0A5D   DCER      SP4A ADD        XR13 CF1             JSL  CHR           ZERO  DATA CHAIN ERROR,WITH ERROR %E002
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10369          ********************************************************************
10370          *                  EXECUTE READ INSTRUCTION (INITIATION)           *
10371          *                  I----------I----------I----------I-----------I*
10372          *                  I 0  0  0  0  0  0  1  1 BLOCKS LEFT   MODIFIER I*
10373          *                  I----------------------I-----------------------I*
10374          *                  I          BYTE COUNT  /  RESIDUE COUNT       I*
10375          *                  I----------------------I-----------------------I*
10376          *                  I TERMINATE DISPLACEMENT  BURST LENGTH (0 =256) I*
10377          *                  I----------------------I-----------------------I*
10378          *                  IRB LR SP XX SA UD  X  X  EXTENDED MEMORY ADDR. I*
10379          *                  I--I--I--I--I--I--I-----I----------------------I*
10380          *                  I       MEMORY ADDRESS  /  RESIDUE ADDRESS      I*
10381          *                  I-------------------------------------------------I*
10382          *                  BIT ABBREVIATIONS:                              *
10383          *                    RB : 0 = "RECORD MODE", 1 = "BURST MODE".     *
10384          *                    LR : 0 = "START ON LEFT BYTE", 1 = "START ON  *
10385          *                         RIGHT BYTE".                             *
10386          *                    SP : READ : 1 = "TERMINATE ON ASCII LINE-FEED *
10387          *                              (GIC ONLY)".                        *
10388          *                         WRITE: 1 = "DO NOT TAG HP-IB END MESSAGE *
10389          *                              TO LAST BYTE"                       *
10390          *                    RW : 0 = "READ(INPUT)", 1 = "WRITE(OUTPUT)".  *
10391          *                    SA : 1 = "SEND/REC. ALL DATA WITH SINGLE MEMORY*
10392          *                         ADDRESS".                               *
10393          *                    UD : 1 = "DO NOT UPDATE INSTRUCTION WORDS AFTER*
10394          *                         EXECUTION".                             *
10395          *                  NOTES:                                          *
10396          *                  THE SEQUENCE AS FOLLOWS:                        *
10397          *                  1.CHECK THE DATA CHAIN BLOCK.                   *
10398          *                  2.ADDRESS THE CHAN TO LISTEN & DEV TO TALK,     *
10399          *                    WITH SECONDARY ADDRESS FROM 'MODIFIER'.       *
10400          *                  3.SAVE BURST LENGTH ,& SET TO %0100             *
10401          *                    IF 0.                                         *
10402          *                  4.JSB TO RRE1 IF 'RECORD' MODE                  *
10403          *                  5.JSB TO DMA IF BURST LENGTH <>1                *
10404          *                    & CONTINUE TO RBYT IF BURST LENGTH=1.FOR      *
10405          *                    SINGLE BYTE DATA BURST WITHOUT USING THE DMA  *
10406          *                    HARDWARE.                                     *
10407          ********************************************************************
10408   0A5E   READ      UBB  JSB  DCH       WRS   000F RH   ANDL                  DCH TO CURRENT BLOCK DRTO AS ADDR;
10410                                                                             UBB:=MODIFIER
10411   0A5F        EFFF SP1B ANDL                FFE0 XR26 ADDL      XR0         MASK OFF CRTL (3:1); FORM TALK CMD FOR DEV
10413   0A60                  CAD  LSL  SP4A            UBA  JSL  TLK1 SP1B   NF2  LISTEN CMD FOR CHAN; SEND HP-IB CMDS IF NF2
10415   0A61             XR1  ADD                     48FF SP1B ANDL      RH        UBA:=BUSRT LENGTH, RH:=<CRTL>
10417   0A62        UBA       JSBC DMA       NZRO       SP1B JSL  RRE1        POS  DMA IF BURST<>1, RRE1 IF 'RECORD' MODE
10419   0A63             SP1B JSB  RBYB      F2         XR10 INC         ROA3      RBYB IF F2, UBA:=CRTL
10421                                                                             READ M(<POINT>+3), MEM ADDR
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10423            ********************************************************************
10424            *                    THIS ROUTINE IS CALLED BY READ INSTRUCTION     *
10425            *                    TO DO SINGLE BYTE DATA BURST WITHOUT USING THE *
10426            *                    DMA HARDWARE.                                  *
10427            *                    1.WITH INITIAL MEM ADDR FROM WORD 4&5,DATA FROM*
10428            *                      DEV IS READY FOR TRANSFER.                   *
10429            *                    2.IF FROM SUSPEND,CHECK CRTL(2:1),LINE-FEED    *
10430            *                      DETECT.IF CRTL(2:1)=1,WRITE REG#0:=%0001,ELSE*
10431            *                      %C001,TERMINATE ON 'END' ONLY.               *
10432            *                    3.READ THE BYTE FROM REG#0,<WORD>.             *
10433            *                    4.READ THE MEM LOCATION SPECIFIED FROM THE MEM *
10434            *                      ADDRESS;CHECK CRTL(1:1).                     *
10435            *                        IF CRTL(1:1)=1,MASK OFF THE LEFT BYTE &SF1;*
10436            *                        IF CRTL(1:1)=0,MASK OFF THE RIGHT BYTE & CF1.
10437            *                        & SWAP THE UPPER & LOWER BYTE OF <WORD>.   *
10438            *                    5.SEND THE <WORD>+MASKED BYTE FROM MEM BACK TO *
10439            *                      THE MEM ADDR.                                *
10440            *                    6.JSB TO BYTE RESTORE (BYRS).                  *
10441            *                    7.UNADDRESS THE DEVICE.                        *
10442            *                    8.IF <WORD>(1:1)=1,CONTINUE AT RDEN,ELSE RDBE. *
10443            ********************************************************************
10444  0A64      UBA  UBA  ADD  LSL       NEG              INC       RG        ZERO  SKIP IF CRTL.(2:1)=1;
10446                                                                                 REG#:=%0001, SLOW DOWN JSB
10447  0A65      C001      ADDL      RG                    JSL  IOW  SP3B      ZERO  REG#0:=%C001 IF LINE-FEED; MEM ADD
10449  0A66 RBYB      OPA  ADD       SP4A                  JSL  IOR  SP3B      ZERO  SP4A:=M(<POINT>+3); READ REG#0,MEM ADDR
10451  0A67      RG        ADD  LSL       HBF2  00FF SP1B ANDL      BKX4            SET F2 IF REG#0.(1:1)=1;
10453                                                                                BKX4:=EXTENDED ADDR BANK#, CF5B
10454  0A68      005F XR12 ADDL      RG                SP4A ADD       XR0  ROX4       RG:=UNTALK CMD; XR0:=ADDR, READ DATA
10456  0A69      RG        ADD  RRZ  RH                SP2B INC            ROA3       RH:=<WORD>:=REG#.(8:8);
10458                                                                                READ M(<POINT>+1), TD/BL(RDBE)
10459  0A6A      RH   RH   JSB  RBYA      NEG         OPB  ADD  LLZ  SP3B SF1        RBYA IF CRTL.(1:1)=1; SP3B:=<ADDR>(0:8), SF1
10461  0A6B      RH        ADD  SWAB RH                SREG ADD  RRZ  SP3B CF1        RH:=<SWAPPED WORD>; SP3B:=<ADDR>.(8:8), CF1
10463  0A6C RBYA 0002 SP2B ADDL      XR0               XR0  ADD            WRX4       XR0:=<POINT>+2; MEM ADDR
10465  0A6D                JSB  BYRS      UNC   SP3B RH   IOR            DATA       BYRS FOR BYTE RESTORE; <WORD>+<ADDR> TO ADDR
10467  0A6E                ADD       RH                    JSL  IOW  SP3B      ZERO  ; SEND UNTALK
10469  0A6F      003F XR12 ADDL      RG                    JSL  IOW                 UNC  RG:=UNLISTEN CMD; SEND
10471            ********************************************************************
10472            *                    1.DO UNADDRESS ROUTINE IF 'REORD' MODE.        *
10473            *                    2.THEN,IF BCOUNT<>0,EXECUTE THE NEXT INSTR.    *
10474            *                    3.& IF DATA CHAIN=0,ALSO EXECUTE THE NEXT ISNTR.*
10475            *                    4.& IF DATA CHAIN<>0,DO DATA CHAIN POINTER UPDATE*
10476            *                      AND EXECUTE THE NEXT INSTR.                  *
10477            ********************************************************************
10478  0A70 RDEN      SP1B JSB  RDE1      NEG         BKX6 JSB  RDBE SP1B      NF2   RDE1 IF CRTL(0:1)=1;RDBE IF NF2(REG#0(1:1))
10480  0A71      005F XR12 ADDL      RG                    JSL  IOW  SP3B      ZERO  UNTALK
10482  0A72      003F XR12 ADDL      RG                    JSL  IOW                 UNC  UNLISTEN
10484  0A73 RDE1 SP0A      JSB  FCHR      NZRO        XR27 ADD            ROAS       REFETCH IF BCOUNT:=0;READ DRTO(FOR DCPU)
10486  0A74                JSB  DCPU      UNC         BKX6 JSB  FCHR      ZERO       UPDATE DATA CHAIN IF DC<>0; FCH4
10488  0A75                ADD                               JSB  FCHR      UNC       REFETCH NEXT INSTR
```

RECORD NO	C S ADDR	********** ALU A ********** LABL	RREG	SREG	FUNC	SFNC	STOR	SPSK	********** ALU B ********** RREG	SREG	FUNC	SFNC	STOR	SPEC	SKIP	COMMENT
10491		**********														
10492		* NO 'END' READ TERMINATION *														
10493		**********														
10494	0A76	RDBE		OPA	ADD	LRZ	RG			XR27	ADD			ROAS		TERM DSPL;READ DRTO ADDR
10496	0A77				INC	LSL	RH		UBA		ADD	SWAB			POS	SET UP <POINT> TO SKIP;TEST TERMINATE DSPL
10498	0A78		SPOA		JSB	FCHR		NZRO	FF00	RG	ADDL		RG			FCHR IF BCOUNT<>0;TERM DSPL
10500	0A79				JSB	DCPU		UNC		SP1B	JSB	FCRT			ZERO	DCPU IF DATA CHAIN<>0; FCRT IF SP1B = 0
10502	0A7A	FCHR			ADD				,	RH	SP2B	JSB	DCJP	SP2B	UNC	;DCJP FOR UPDATED <POINT>
10504	0A7B	FCHL			ADD			CF2				JSL	FCH4		UNC	FETCH NEXT INSTR;FOR DC<>0
10506	0A7C	FCRT			JSB	FCHL		UNC	RG	SP2B	ADD		SP2B			SETUP <POINT>+TERM DSPL TO SKIP NEXT INSTR

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10509           ********************************************************************
10510           *                  EXECUTE WRITE INSTRUCTION (INITIATION)          *
10511           *                  I---------I---------I---------I-----------I     *
10512           *                  I 0  0  0  0  0  1  0  0  BLOCKS LEFT  MODIFIER I*
10513           *                  I---------------------I---------------------I   *
10514           *                  I          BYTE COUNT  /  RESIDUE COUNT       I*
10515           *                  I---------------------I---------------------I   *
10516           *                  I X X X X X X X X X X X   BURST LENGTH (0 =256) I*
10517           *                  I---------------------I---------------------I   *
10518           *                  IRB LR SP XX SA UD  X  X  EXTENDED MEMORY ADDR. I*
10519           *                  I--I--I--I--I--I--I--I-----I---------------------I*
10520           *                  I       MEMORY ADDRESS  /  RESIDUE ADDRESS      I*
10521           *                  I-----------------------------------------------I*
10522           *                  BIT ABBREVIATIONS:                              *
10523           *                    RB : 0 = "RECORD MODE", 1 = "BURST MODE".     *
10524           *                    LR : 0 = "START ON LEFT BYTE", 1 = "START ON  *
10525           *                         RIGHT BYTE".                             *
10526           *                    SP : READ : 1 = "TERMINATE ON ASCII LINE-FEED *
10527           *                              (GIC ONLY)".                        *
10528           *                         WRITE: 1 = "DO NOT TAG HP-IB END MESSAGE *
10529           *                              TO LAST BYTE".                      *
10530           *                    RW : 0 = "READ(INPUT)", 1 = "WRITE(OUTPUT)".  *
10531           *                    SA : 1 = "SEND/REC. ALL DATA WITH SINGLE MEMORY*
10532           *                         ADDRESS".                               *
10533           *                    UD : 1 = "DO NOT UPDATE INSTRUCTION WORDS AFTER*
10534           *                         EXECUTION".                             *
10535           *                    XX :   = DON'T CARE                           *
10536           *                    NOTES:                                        *
10537           *                    THE SEQUENCE AS FOLLOWS:                      *
10538           *                    1.CHECK THE DATA CHAIN BLOCK.                 *
10539           *                    2.ADDRESS THE CHAN TO TALK & DEVICE TO LISTEN,*
10540           *                      & WITH SECONDARY ADDRESS FROM 'MODIFIER'.   *
10541           *                    3.SAVE BURST LENGTH IN XR1(A),&SET TO %100 IF 0*
10542           *                    4.JSB TO WRXD IF 'RECORD' MODE;& JSB TO DMA   *
10543           *                      IF BURST LENGTH <>1.                        *
10544           *                    5.IF BURST LENGTH=1 THEN CONTINUE TO WBYT.    *
10545           *                      ROUTINE FOR SINGLE BYTE DATA BURST WITHOUT  *
10546           *                      USING THE DMA HARDWARE.                     *
10547           ********************************************************************
10548    0A7D  WRIT      UBB  JSB  DCH        WRS   000F RH   ANDL                  JSB DCH TO CURRENT BLOCK,DRTO ADD;MODIFIER
10550    0A7E       1000 SP1B IORL                 FFC0 XR26 ADDL      XR0         MASK OFF CRTL(3:1);FORM LISTEN CMD FOR DEV
10552    0A7F       001E      ADDL      SP4A             UBA  JSL  TLK1 SP1B      UNC  TALK CMD;SEND HP-IB,XR1 FOR WBXD IN (DMA)
```

```
 RECORD  C S                 ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10555            ************************************************************************
10556            *                    THIS ROUTINE IS CALLED BY WRITE INSRTUCTION       *
10557            *                    TO DO SINGLE BYTE DATA BURST WITHOUT USING THE    *
10558            *                    DMA HARDWARE.                                     *
10559            *                    1.WITH INITIAL MEM ADDR FROM WORDS 4&5,DATA IS    *
10560            *                      READY FOR TRANSFER STARTING AT THAT ADDRESS.    *
10561            *                    2.CRTL(1:1) IS CHECKED TO DETERMINE WHETHER THE   *
10562            *                      THE FIRST BYTE OF THE TRANSFER WILL BE TAKEN    *
10563            *                      FROM LEFT(0) OR RIGHT(1) BYTE OF THE MEMORY     *
10564            *                      ADDRESS BUFFER                                  *
10565            *                    3 CRTL(2:1) WILL DETERMINE WHETHER SUPPRESS END   *
10566            *                      MESSAGE IS TAGGED.                              *
10567            *                      IF CRTL(2:1)=0,OR BCOUNT=1 & DC=0,THEN WRITE    *
10568            *                      TO REG#0:=%8000+<WORD>;                         *
10569            *                      OTHERWISE,REG#0:=<WORD>.                        *
10570            *                      <WORD> =DATA FROM MEMORY ADDRESS.               *
10571            *                    4.JSB TO BYRS FOR BYTE TRANSFER REGISTER UPDATE;  *
10572            *                      DECREMENT BCOUNT,& CHECK IF NEED TO ENABLE      *
10573            *                      MEMORY UPDATE IF IT DOES,COMPLEMENT CRTL(1:1),  *
10574            *                      LEFT/RIGHT BYTE,&STORE BACK TO CRTL ADDRESS.    *
10575            *                    5.F1 IS USED TO CHECK WHTETHER THE LAST BYTE IS   *
10576            *                      LEFT OR RIGHT.                                  *
10577            *                    6.INCREMENT THE MEMORY ADDRESS BY 1;IF CRTL(4:1)  *
10578            *                      =0,RETURN THE UPDATED MEM ADDR TO WD5           *
10579            *                      THEN CHECK IF IT IS ADDR ROLL-OVER,WITH ERROR   *
10580            *                      CODE %E800.                                     *
10581            *                    7.IF BCOUNT=0 & DC=0,RESET DMA DIRECTION BY       *
10582            *                      WRITING REG#6:=%0060;THEN UNLISTEN DEVICE       *
10583            *                      ON HP-IB.THEN,REFETCH.                          *
10584            *                    8.IF BCOUNT OR DC IS NOT 0,DO UPDATE ON THE       *
10585            *                      DATA CHAIN BLOCKS,UPDATE THE CP<POINT>,REPEAT 7.*
10586            ************************************************************************
10587  0A80            SP1B JSB  WRXD       POS        XR10 INC        RG   ROX3        WRXD IF 'RECORD';DMA IF BURST<>1
10589  0A81            XR1  ADD             CF1   00FF SP1B ANDL       BKX4             BURST LENGTH;CF5B FROM TLK1,BANK ADDR
10591  0A82       UBA       JSBC DMA        NZRO       OPB  ADD        XR0  ROX4        DMA IF <>1;READ MEM ADDR,<WORD>,XR0=MEM ADDR
10593  0A83       RG        CAD       XR0               SP1B ADD  LSL             POS   <POINT>+2 ADDR;SKIP IF CRTL(1:1)=0,LEFT BYTE
10595  0A84       UBB       ADD  LSL  NEG               OPB  ADD  RRZ  RG   SF1  UNC   SKIP CRTL(2:1)=1,TAG WITH 'END';RG:=<WORD>
10597  0A85                 JSB  WBEN UNC               SREG ADD  LRZ  RG             WBEN IF NOT SUPPRESS END;RG:=<WORD>
10599  0A86       SPOA      CAD                         BKX6 ADD                      <BCOUNT>-1;DATA CHAIN
10601  0A87            UBB  ADD             NZRO       UBA  JSB  WBNE            NZRO  SKIP IF DC<>0;WBNE IF BCOUNT<>1
10603  0A88  WBEN RG   XR14 ADD       RG                    ADD                       RG:=<WORD>+%8000;
10605  0A89  WBNE           ADD       RH                    JSL  IOW            UNC   ;WRITE TO REG#0:=RG
10607  0A8A                 ADD                      EFFF SP1B ANDL       SP1B        MASK OFF THE WRITE BIT
10609  0A8B                 JSB  BYRS       UNC        XR27 ADD             ROAS      BYRS FOR BYTE XFER UPDATE;READ DRTO FOR DCPU
```

```
 RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10612          ***************************************************************
10613          *                  WRITE INSTRUCTION (TERMINATION)              *
10614          *                  1.IF BCOUNT<>0,UPDATE <POINT> TO SKIP NEXT INSTR;*
10615          *                  2.& IF DC<>0,UPDATE DC POINTER;ALSO SKIP NEXT  *
10616          *                    INSTR                                        *
10617          *                  3.IF BCOUNT=0 & DC=0 THEN UPDATE <POINT> TO    *
10618          *                    EXECUTE NEXT INSTR.                          *
10619          ***************************************************************
10620    0A8C  WDTB SPOA      JSB  WUN       NZRO       BKX6 ADD       SP1B          WUN IF BC<>0;DATA CHAIN FOR DCJP
10622    0A8D                 JSB  WUNL      UNC        UBB  JSB  DCPU          NZRO  DCPU IF DATA CHAIN<>0;ELSE WUNL
10624    0A8E  WUN            JSB  DCJP      UNC             INC  LSL  RH            DCJP TO END OF DC; RH := 2
10626    0A8F  WUNL 0060      ADDL      RG                   XR21 JSL  IOW  SP3B      NZRO  RESET DMA DIRECTION IN REG#6
10628    0A90       003F XR12 ADDL      RG                        JSL  IOW  SP3B      ZERO  TO UNTALK HP-IB
10630    0A91                 JSB  CRCS      ZERO       XR10 ADD                ROX3       CHECK IF CRC SEND; READ <CRTL>
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10633           ************************************************************************
10634           *                    DATA CHAIN SUBROUTINES                            *
10635           *                    THIS ROUTINE CHECKS THAT THE OUTBOUND FIFO IS     *
10636           *                    EMPTY,THEN RESTORES THE INSTR POINTER TO MEMORY   *
10637           *                    AND COMPUTES THE ADDRESS OF THE CURRENT DATA      *
10638           *                    CHAIN BLOCK BEING EXECUTED.                       *
10639           *                    SEQUENCE AS FOLLOWS.                              *
10640           *                    1.SPOA:=BCOUNT;CHECK FIFO IF FROM SUSPEND         *
10641           *                    2.SAVE <POINT>-1 IN DRTO ADDRESS.                 *
10642           *                    3.INCREMENT <POINT> OVER THE NUMBER OF DATA CHAIN*
10643           *                      BLOCKS INDICATED IN BITS 8-11 OF 1ST INSTR. &   *
10644           *                      READ THE NEW BCOUNT IN CASE THE 'OLD' BCOUNT=0.*
10645           *                    4.IF 'OLD' BCOUNT=0,JSB CHR IF DATA CHAIN=0,OR    *
10646           *                      THE NEW BCOUNT IS ALSO 0,WITH ERROR CODE=%E002.*
10647           *                    5.IF 'OLD' BCOUNT<>0,OR IF DATA CHAIN 'AND' NEW   *
10648           *                      BCOUNT<>0,THEN MASK OFF THE MODIFIER BITS 12-15*
10649           *                      & FORM THE SECONDARY HP-IB CMDS;SAVE DATA CHAIN*
10650           *                      BITS IN XR12(A);AND SP1B:=<SWAPPED CRTL>,WD4.   *
10651           ************************************************************************
10652   0A92    DCH  UBB  SP4A ADD       XR2  CF1        SP2B ADD            PSHR      MODIFIER;<POINT>
10654   0A93         UBB       CAD            DATA            JSL  SUSP          NF2  DRTO:=<POINT>-1;SUSP IF FROM SUSPEND
10656   0A94              OPA  ADD                      00FO RH   ANDL      SP1B      BCOUNT;DATA CHAIN
10658   0A95    CGE2 E002      ADDL           SP4A            UBA      JSB  CGET          NZRO  DATA CHAIN ERROR;CGET IF BCOUNT<>0;
10660   0A96                   JSB  DCJP       UNC             SP1B JSL  DCER          ZERO  UPDATE DATA CHAIN POINTER;DCER IF DC=0
10662   0A97              OPA  JSB  DCER       ZERO                 ADD                      CHR IF NEW BCOUNT=0;
10664   0A98    CGET      SP2B INC            SP4A ROX3            CAD       RG   POPR       READ M(<POINT>+1),TO/BURST;
10666   0A99              OPA  ADD            SP0A                 INC       XR10 ROX3       SAVE BCOUNT;READ <POINT>+2;<CRTL>
10668   0A9A                   CAD  RRZ                 RG   SP2B ADD            ROX3       %00FF;READ M(<POINT>-1)
10670   0A9B         UBA  OPA  AND            XR1  NZRO       OPB  ADD       SP1B SF5B  <BURST LENGTH>&SKIP IF =0;<CRTL>,F5B(TLK1)
10672   0A9C                   INC  SWAB       XR1             0005 SP2B ADDL       XR2       CRRY  <BURST LENGTH> IF <>ZERO,NEXT CHAIN <POINT>
10674   0A9D         SP0A      ADD            XR0             00FO OPB  ANDL       BKX6      RSB   BCOUNT;<SWAPPED CRTL),SF5B FOR TLK1 RETURN
10676           ************************************************************************
10677           *                    ROUTINE FOR NEXT DATA CHAIN BLOCK                 *
10678           *                    XRB2:=SP2B + 5 (SET UP IN DCH)                    *
10679           ************************************************************************
10680   0A9E    DCNB E002      ADDL           SP4A            UBB  UBA  INC       XR10 ROX3       ERROR CODE;READ M(<POINT>+2),<CRTL>
10682   0A9F              OPA  JSB  DCER       ZERO  RREG SREG SUB            ROX3       DCER IF BCOUNT=0;READ M(<POINT>-1) FOR DC
10684   0AA0              OPA  ADD            SP0A RSB         OPB  ADD       SP1B       BCOUNT;NEW <CRTL>
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10687           ;*****************************************************************
10688           ;                   DATA CHAIN POINTER UPDATE ROUITNE             *
10689           ;                   1.GET THE <ADDR> OF 1ST INSTR FROM <DRTO ADDR>;  *
10690           ;                     & COMPARE IT WITH THE CURRENT <POINT>       *
10691           ;                   2.IF<DRTO ADDR> > <POINT>,THEN DATA CHAIN ERROR. *
10692           ;                     IF<DRTO ADDR> = <POINT>,MASK 1ST WD WITH %FF0F *
10693           ;                     IF<DRTO ADDR> < <POINT>,DO NOT MASK,THEN IT IS *
10694           ;                     NOT IN THE FIRST BLOCK,JUST INCREMENT <POINT>  *
10695           ;                   3.IF DATA CHAIN BLOCK LEFT=%00F0,THEN DATA CHAIN *
10696           ;                     BITS TOO LARGE,& WITH DATA CHAIN ERROR.       *
10697           ;                   4.ELSE,RESTORE THE NEW DATA CHAIN TO 1ST WD OF   *
10698           ;                     THE INSTRUCTION.                              *
10699           ;*****************************************************************
10700   0AA1  DCPU E002       ADDL      SP4A            SP2B ADD                      DATA CHAIN ERROR;<POINT>
10702   0AA2            OPA   ADD             ROB3  UBB       CAD       RG          READ 1ST INSTR;<POINT>-1(CURRENT POINTER)
10704   0AA3            OPA   ADD             WRX3  UBB  UBA  JSBS DCER             NCRY C<POINT> ADDR;DCER IF C<POINT> < CP <POINT>
10706   0AA4      UBA   RG    XOR             NZRO        OPB  ADD       RG          SKIP C<POINT>=CP<POINT>;RG:=DATA CHAIN
10708   0AA5      FF0F  UBB   ANDL      RG          00F0 SREG ANDL                  RG:=DATA CHAIN (0);DATA CHAIN
10710   0AA6      0010        ADDL                  RREG UBB  JSBS DCER             ZERO INCR IF NOT IN 1ST BLOCK;DCER IF DC:=%00F0
10712   0AA7      UBA   RG    ADD             DATA        ADD                       RSB  REWRITE NEW DATA CHAIN BITS;RETURN
10714           ;*****************************************************************
10715           ;                   BYTE RESTORE ROUTINE                          *
10716           ;                   THIS ROUTINE IS CALLED BY BOTH WBYT AND RBYT,    *
10717           ;                   WHICH USE IT TO UPDATE THE VALUES OF <CRTL>    *
10718           ;                   (TOGGLE THE LEFT/RIGHT BIT),<BCOUNT>(DECREMENT   *
10719           ;                   BY 1),AND <ADDR>(INCREMENT BY 1 IF F1,INDICATING *
10720           ;                   LAST BYTE REFERENCED WAS THE RIGHT BYTE) AND   *
10721           ;                   RESTORE THEM TO MEMORY. ADDRESS ROLL-OVER IS ALSO*
10722           ;                   CHECKED WITH ERROR CODE %E800.                *
10723           ;*****************************************************************
10724   0AA8  BYRS SP0A       CAD       SP0A            0400 SP1B ANDL                ZERO BCOUNT-1;SKIP IF CRTL(5:1)=0
10726   0AA9            XR0   ADD             WRX3  RREG SP1B XFRS            CF1   RSB  WD4 ADDRESS; CF1 AND RETURN
10728   0AAA      UBB   XR12  XOR             DATA  RREG       ADD  LSL       PSHR       COMPLEMENT CRTL(1:1); %0800, PUSH SUBR STK
10730   0AAB      UBB   SP1B  AND             NZRO        XR0  JSBI BYR1 XR0          NF1  SKIP IF CRTL(4:1)=1;
10732                                                                                    XR0:=MEMADDR + 1, NO WR IF NF1
10733   0AAC            UBB   ADD       SP4A  DATA            CAD  LSL       CF1        SP4A:=MEMADDR + 1, STORE BACK; UBB:=-2, CF1
10735   0AAD  BYR1 UBB   XR0   ADD             WRX3                ADD       POPR       WRITE BACK THE BCOUNT; POP SUBR STK
10737   0AAE      SP0A        ADD             DATA  E800          ADDL                   M(POINT-2):=BCOUNT; ADDR ROLL OVER ERROR
10739   0AAF      UBB         ADD       SP4A  RSB         XR0  JSL  DCER             ZERO RETURN; CHR IF ADDR ROLL-OVER
10741           ;*****************************************************************
10742           ;                   END OF THE DATA CHAIN BLOCK                   *
10743           ;*****************************************************************
10744   0AB0  DCJP      SP1B  ADD   LSR   SP0A            SP1B REPN                 03   SHIFT RIGHT;SETUP TO SHIFT RIGHT 4 TIMES
10746   0AB1      SP0A        ADD   LSR               UBB       ADD  LSR       DCTR CTR0 UBA:=4 X DATA CHAIN;UBB:=DATA CHAIN X 1
10748   0AB2      UBA   UBB   ADD               CRRY  E002      ADDL                    DATA CHAIN X 5;DATA CHAIN ERR CODE
10750   0AB3      UBB         ADD       SP4A  RSB   UBA  SP2B ADD       SP2B ROA3      ERR CODE (CGE2),RETN;READ NEW BCOUNT,<POINT>
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10753           ********************************************************************************
10754           *                    ERROR RECOVERY SECTION                                    *
10755           ********************************************************************************
10756   0AB4  V         SP2B ADD            CF1         XR27 ADD             WRS         <POINT>;DRTO ADDRESS
10758   0AB5                 JSB  DCER      UNC   UBA       CAD             DATA        DCER;M(<DRTADRO>):=<POINT>-1
10760           ********************************************************************
10761           *                  THIS ROUTINE WILL ALLOW TO RELEASE THE HANDSHAKE *
10762           *                  AND THE SYSTEM TO PROCEDE NORMALLY FROM POINT OF *
10763           *                  LOCKUP(AFTER THE TIMEOUT ABORT INTERRUPT.        *
10764           *                  BASICALLY,IT WILL CHECK TO SEE IF OUTBOUND FIFO  *
10765           *                  IS LOCKUP,THEN INSTEAD OF CLEARING IT DIRECTLY   *
10766           *                  IT WILL TAKE PHI OFF-LINE TO THE HP-IB,THEN BRING*
10767           *                  ONLINE AGAIN BY ASSERTING IFC TO REGAIN CONTROL, *
10768           *                  CLEAR STATUS BITS AND CONTINUE.                  *
10769           ********************************************************************
10770   0AB6  CHRZ           ADD                             JSL  IFC             UNC   ASSERT IFC TO CLEAR HP-IB
10772   0AB7                 ADD       RG              XR22  JSL  IOW  SP3B       NZRO  TAKE PHI OFFLINE
10774   0AB8       0069      ADDL      RG              XR21  JSL  IOW  SP3B       NZRO  SELECT POLL REQ,REN,PARITY,CLEAR FIFO
10776   0AB9       0080 XR12 ADDL      RG              XR22  JSL  IOW  SP3B       NZRO  GET ONLINE AND ENABLE CRC
10778   0ABA                 ADD                             JSL  IFC             UNC   ASSERT IFC TO REGAIN CONTROL AS CIC
10780   0ABB                 CAD       RG              XR17  JSL  IOW  SP3B       NZRO  CLEAR PHI INTERRUPTS
10782   0ABC                 ADD                             JSL  CHRX            UNC   ALLOW CSRQ ONLY FOR PPOLL OR STATUS CHG
10784           ********************************************************************
10785           *                  IF SUSB IS THE ENTRY POINT OTHER THAN FROM SUSP,*
10786           *                  F2 WILL BE NORMALLY SET. IT IS SUSPENDED FOR    *
10787           *                  WAITING DATA TO ARRIVE AT THE CHANNEL.          *
10788           *                  1. IF F2 SET: WRITE REG#3:=%8004,               *
10789           *                                WRITE REG#F:=%0080+DEV#,          *
10790           *                                SET DRT3(12:1):=1.                *
10791           *                  2. IF NF2   : WRITE REG#3:=%8002,               *
10792           *                                WRITE REG#F:=%0080+DEV#,          *
10793           *                                SET DRT3(13:1):=1.                *
10794           *                     SAVE CHANNEL PROGRAM POINTER IN DRT0.        *
10795           ********************************************************************
10796   0ABD  SUSC           ADD             CF2             ADD                      CF2;
10798   0ABE  SUSB      SP2B ADD                         XR27 ADD             WRS         <POINT>;DRTO AS ADDR
10800   0ABF                 INC  LSL  RH          UBA       CAD             DATA        %0002;DRTO:=<POINT>-1
10802   0AC0       UBA  XR14 ADD       RG   NF2   0004      ADDL      SP1B           IF NF2,WRITE TO REG#3:=%8002;%0004
10804   0AC1       RREG UBA  ADD       RG                XR18 JSL  IOW  SP3B       NZRO  IF F2,WRITE TO REG#3:=%8004;
10806   0AC2       0080 XR6  ADDL      RG                XR30 JSL  IOW  SP3B       NZRO  WRITE REG#F:=%0080+DEV#
10808   0AC3       E000 XR4  ANDL                        XR28 ADD             WRS  F2    MASK DRT3(0:3);DRT3 AS ADDR,SKIP IF F2
10810   0AC4                 ADD                   UBA   SP1B ADD             DATA UNC   ;M(<DRT3>):=DRT3(0:3)+%0004 IF NF2
10812   0AC5                 ADD                   UBB   SP1B ADD             DATA       ;M(<DRT3>):=DRT3(0:3)+%0008 IF F2
10814   0AC6                 ADD                             JSL  CPEX            UNC   EXIT CHANNEL PROGRAM
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10817           ************************************************************************
10818           *                                                                      *
10819           *     START OF THE CHANNEL PROGRAM--                                   *
10820           *     1.  SEND OUT SPOLL2 TO DETERMINE THE CHANNEL(S) REQUESTING       *
10821           *         THE SERVICE.                                                 *
10822           *     2.  #CHCK TO CHECK IF CHANNEL# 0 REQUESTS SERVICE; THIS IS       *
10823           *         MAINLY FOR RUNNING THE DIAGNOSTICS WITH ADCC.                *
10824           *     3.  DO ROUND-ROBIN ON THE SPOLL2 RESPONSE AND THE CHANNEL        *
10825           *         NUMBER LAST SERVICED IN XRA127.                              *
10826           *     4.  ISSUE OBSI COMMAND TO THE CHANNEL, TO OBTAIN THE TYPE        *
10827           *         OF SERVICE REQUEST.                                          *
10828           *     5.  READ REG# 1 TO DETERMINE IF THE CHANNEL IS CONTROLLER-       *
10829           *         IN-CHARGE; IF NOT PROCEED AS NORMAL CHANNEL PROGRAM.         *
10830           *     6.  IF CONTROLLER-IN-CHARGE CHECK IF THE OBSI RESPONSE IS        *
10831           *         CHANNEL REQUEST ( DMA COMPLETION ); IF TRUE PROCEED AS       *
10832           *         NORMAL CHANNEL PROGRAM.                                      *
10833           *     7.  IF NOT ( DEV REQ ), ENABLE THE PHI INTERRUPT MASK BY         *
10834           *         WRITING !7FFF TO REG# 3.                                     *
10835           *     8.  DO ANOTHER OBSI TO DETERMINE IF THERE IS A TRUE SERVICE      *
10836           *         REQUEST FOR THE DEV# FROM THE PREVIOUS OBSI.                 *
10837           *     9.  IF OBSI'2.(8:1) = 0, A TRUE REQUEST, SET BIT MASK FOR        *
10838           *         THE DEVICE#; OTHERWISE IGNORE IT.                            *
10839           *    10.  READ REG# 0 FOR THE PPOLL RESPONSE & 'OR' IT WITH THE        *
10840           *         DEV BIT MASK IF ANY.                                         *
10841           *    11.  READ REG# B FOR THE DEV# LAST SERVICED; DO ROUND-ROBIN       *
10842           *         AND DETERMINE THE NEXT DEV# TO BE SERVICED.                  *
10843           *    12.  PROCEED WITH THE NORMAL CHANNEL PROGRAM WITH                 *
10844           *                 RG = OBSI RESPONSE                                   *
10845           *               BKX5 = CHANNEL DEVICE                                  *
10846           *                CTR = REGN POINTER AT LAST CHANNEL SERVICED           *
10847           *                                                                      *
10848           ************************************************************************
10849           *
10850   0AC7  CSRQ 00FF       ADDL                    E000       ADDL      SP3B          GET THE CHAN# LAST SERVICED; SPOLL2 CMD
10852   0AC8                  ADD             SF1         UBA  JSL  IOR  CTR       UNC   SF1; SEND SPOLL2 CMD, REGN @ CHAN#
10854   0AC9       000F REGN  ANDL      RH                 BNKS JSL  CKCH XR15      UNC   CHAN# LAST SERVICED; #CPEX IF NO CHAN REQ
10856                                                                                      #CKCH FOR ADCC ONLY; SHOULD ALWAYS BE
10857                                                                                      CHAN# 1, SAVE S-BANK
10858   0ACA                  ADD       XR14 F1HB         RH   ADD       CTR  CCTX      !8000; SET UP CTR FOR LAST CHANNEL SERVICED
10860   0ACB            UBA   ADD  LSR  XR12              RG   REPC            DCTR CTRO XRA12:=!4000, REPEAT LOOP LAST CHAN SERVICED
10862   0ACC                  ADD                       UBB  CSL       RG   DCTR CTRO ; CIRCULAR SHIFT LEFT THE CHAN JUST SERV
10864   0ACD       000F       ADDL      RH                 RH   ADD       CTR  CF5B     RH:=!F; SET UP CTR
10866   0ACE            RG    ADD             CF2              REPC                    REPEAT LOOP FOR NEXT CHANNEL TO BE SERVICED
10868   0ACF            UBA   CSL             HBF2  RH   CTRS AND             ICTR F2  ; SKIP IF F2
10870   0AD0       UBB        ADD  LSL                 4000       ADDL      SP3B        SHIFT CHANNEL#; OBSI CMD
10872   0AD1       00FF       ADDL                     UBA  UBA  ADD  LSL  BKX5        MASK, CHAN# IN BITS 9-12
10874   0AD2       0100       ADDL      SP4A               UBA  JSL  IOR  CTR       UNC   ; SEND OBSI, SET CTR TO SAVE CHAN# IF #PAUL
10876   0AD3       0200 RG    ANDL      SP4A               RG        ADD  RRZ  BKX5      CHECK DEV REQ; CHAN#
10878   0AD4            RG    ADD       SP0A               SP4A JSL  IOR  SP3B      NZRO  SAVE OBSI RESPONSE; READ REG# 1
10880   0AD5       0010 RG    ANDL                     1300       ADDL      SP3B        MASK CIC; WRITE REG# 3
10882   0AD6       SP0A       ADD       RG                 UBA  JSL  PAUL            ZERO  RESTORE OBSI; #PAUL IF NON-CIC
10884   0AD7                  CAD  LSR                     SP4A JSL  PAUL            ZERO  !7FFF, #PAUL IF DEV REQ
10886   0AD8            XR12  ADD       SP4A               UBA  JSL  IOR  RG        UNC   4000; WRITE REG#3:=7FFF
10888   0AD9                  ADD                          SP4A JSL  IOR  SP3B      NZRO  ; DO ANOTHER OBSI
10890   0ADA       SP0A       ADD  RRZ                 RG        ADD  SWAB          NEG   RESTORE 'OLD' OBSI RESP; SKIP IF BIT 8 SET
10892   0ADB       0007 RG    ANDL      SP4A                     JSL  OBSO          ZERO  RESTORE 'OLD' OBSI RESPONSE
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
10894	OADC		00FF		ADDL		SP4A				JSL	IOR	SP3B		ZERO	MASK; READ REG# 0
10896	OADD	OBSR	RG		ADD	RRZ			RG		ADD	RLZ				PPOLL MASK
10898	OADE		SP0A		ADD		RG		UBA	UBB	IOR		SP1B		NZRO	RESTORE OBSI; #PAUL IF NO PPOLL RESPONSE
10900	OADF		0B00		ADDL					SP4A	JSL	PAUL	CTR		UNC	REG# B; #PAUL IF NO PPOLL, SET CTR (XRA127)
10902	OAE0				ADD					UBA	JSL	IOR	SP3B		NZRO	; READ REG#B
10904	OAE1				ADD				0007	RG	ANDL		CTR			; MASK LAST DEV# SERVICED
10906	OAE2			UBB	ADD		SP0A			SP1B	REPC			DCTR	CTR0	; REPEAT LOOP
10908	OAE3			UBA	ADD					UBB	CSL		SP1B	DCTR	CTR0	
10909	OAE4				ADD				UBA		CAD		CTR			
10910	OAE5			SP1B	ADD			CF2		BKX5	REPC		RH			; CHAN DEV#
10912	OAE6			UBA	CSL			HBF2			ADD			ICTR	F2	
10913	OAE7				ADD				FFF8	CTRS	ADDL				NEG	
10914	OAE8				ADD					UBB	ADD		CTR			
10915	OAE9		0078	RH	ANDL				00FF		ADDL		CTR			MASK CHAN#; REGN POINTER
10917	OAEA				ADD				UBA	CTRS	IOR		BKX5			; CHAN DEV#
10919	OAEB		0200	UBB	IORL		RG				JSL	PAUL			UNC	SETUP AS CHAN REQ; START CHANNEL PROGRAM
10921	OAEC	OBS0			ADD				0027	SP4A	SUBL		CTR			; MASK BIT
10923	OAED				ADD						JSL	IOR	SP3B		ZERO	; READ REG# 0
10925	OAEE			RG	ADD	RRZ				REGN	ADD					PPOLL RESPONSE
10927	OAEF		UBA	UBB	IOR		RG				JSL	OBSR			ZERO	DEV BIT MASK FOR ROUND-ROBIN
10929	OAF0	CRCS			ADD				1000	OPB	ANDL					MASK <CRTL>(3:1)
10931	OAF1			UBB	JSB	FCHL		ZERO	RH	SP2B	ADD		SP2B			#FCHL IF NOT CRC SEND; UPDATE POINTER
10933	OAF2		005E	XR12	ADDL		RG				JSL	IOW	SP3B		ZERO	REG# 0 := !405E
10935	OAF3				ADD					XR22	JSL	IOR	SP3B		NZRO	READ REG# 7
10937	OAF4		0040	XR6	ADDL		SP0A		RG		ADD				POS	SP0A := !4040; SKIP IF CRC DISABLED
10939	OAF5		0071	XR12	ADDL		RG				JSL	IOW	SP3B		ZERO	REG# 0 := !4071
10941	OAF6			XR12	ADD	LSL	RG				JSL	IOW	SP3B		ZERO	REG# 0 := !8000
10943	OAF7		SP0A	XR12	ADD		RG				JSL	IOW			UNC	REG# 0 := !4040 + DEV#
10945	OAF8		0071	XR12	ADDL		RG				JSL	IOW			UNC	REG# 0 := !4071
10947	OAF9			XR12	INC	LSL	RG				JSL	IOW			UNC	REG# 0 := !8002
10949	OAFA				JSB	FCHL		UNC			ADD					FETCH NEXT INSTRUCTION

```
RECORD  C.S.                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10952           ********************************************************************
10953           *                                                                  *
10954           *         CSRQ Measurement Routine                                 *
10955           *                                                                  *
10956           ********************************************************************
10957   0AFB  TCSQ           JSZ  TIME      UNC   00A4      ADDL      SP2B           Add to normal execution count
10959   0AFC                 ADD                          ADD                      Wait 1 clock to avoid XR store/read conflict
10961                                                                                  #TIME returns w/ CTR = A5
10962   0AFD           REGN INC       REGN           REGN LINK      REGN           Increment # of CSRQ's (XR37)
10964   0AFE                 ADD                          JSB  CSRQ CTR       UNC   ; JSB to handle CSRQ, CTR := 0
10966   0AFF                 ADD                          ADD                      NOP to keep system happy              (2555)
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

10969           ************************************************************************
10970           **                                                                    **
10971           **            !0B00-!0B1F is reserved for MPL                         **
10972           **                                                                    **
10973           ************************************************************************
10974           *
10975           %0B00
10976           *
10977    0B00         0001 0002 0003 0004                    CON
10978    0B01         0001 0002 0003 0004                    CON
10979    0B02         0001 0002 0003 0004                    CON
10980    0B03         0001 0002 0003 0004                    CON
10981    0B04         0001 0002 0003 0004                    CON
10982    0B05         0001 0002 0003 0004                    CON
10983    0B06         0001 0002 0003 0004                    CON
10984    0B07         0001 0002 0003 0004                    CON
10985    0B08         0001 0002 0003 0004                    CON
10986    0B09         0001 0002 0003 0004                    CON
10987    0B0A         0001 0002 0003 0004                    CON
10988    0B0B         0001 0002 0003 0004                    CON
10989    0B0C         0001 0002 0003 0004                    CON
10990    0B0D         0001 0002 0003 0004                    CON
10991    0B0E         0001 0002 0003 0004                    CON
10992    0B0F         0001 0002 0003 0004                    CON
10993    0B10         0001 0002 0003 0004                    CON
10994    0B11         0001 0002 0003 0004                    CON
10995    0B12         0001 0002 0003 0004                    CON
10996    0B13         0001 0002 0003 0004                    CON
10997    0B14         0001 0002 0003 0004                    CON
10998    0B15         0001 0002 0003 0004                    CON
10999    0B16         0001 0002 0003 0004                    CON
11000    0B17         0001 0002 0003 0004                    CON
11001    0B18         0001 0002 0003 0004                    CON
11002    0B19         0001 0002 0003 0004                    CON
11003    0B1A         0001 0002 0003 0004                    CON
11004    0B1B         0001 0002 0003 0004                    CON
11005    0B1C         0001 0002 0003 0004                    CON
11006    0B1D         0001 0002 0003 0004                    CON
11007    0B1E         0001 0002 0003 0004                    CON
11008    0B1F         0001 0002 0003 0004                    CON
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11010           ***************************************************************
11011           **                                                           **
11012           **          IMB Message routine:  send 32 bit msg to IOA     **
11013           **                                                           **
11014           **          Entry:  XR9.(10:3) = TO code of desired IOA      **
11015           **                  F4         = skip error checks           **
11016           **                  SP3B       = read/write + GIC reg number **
11017           **                  BKX5       = CDEV #                      **
11018           **                  RG         = lower 16 bits of message    **
11019           **                  (upper 16 bits of msg = SP3B IOR BKX5)   **
11020           **                                                           **
11021           **          Exit:   RG         = lower 16 bits of returned msg **
11022           **                  BKX4, XR0B  destroyed                    **
11023           **                                                           **
11024           ***************************************************************
11025           *
11026           %0B20
11027           *
11028    0B20   IOMS           INC       RG          SP3B BKX5 IOR       BKX4 PSHR       Setup timer;  BKX4 := message command
11030    0B21          0282 XR9 IORL                  RG         ADD            WRX4       Put module# in command word; Data if write
11032    0B22                   ADD                   UBA        ADD            BUSC       Send control message, ignore busy
11034    0B23           RG   JSZ  MSTO       ZERO  0C02        ADDL      XR0             MSTO IF TIMEOUT;REC FROM CMD
11036    0B24   WFMS          ADD             MSGI  RG          JSBI *-1  RG        MESS  SKIP IF MSG RECEIVED;LOOP IF MESS NOT SENT
11038    0B25                  JSB  WFMS       UNC   RG          JSZI MRTO RG        ZERO  LOOP IF NOT TIMEOUT;MRTO IF MSG NOT RECEIVED
11040    0B26          0090    ADDL                                XR0 ADD            BUSC  Mask to clear MSGINT; Read FROM code
11042    0B27          XR9     INC                   UBA         ADD            CCPX F4B   UBA := MOD#+1; Clear MSGINT, skip if FLUSH
11044    0B28          0401    ADDL                  UBA   OPB    JSZC INVM         NZRO  REC UPPER WD CMD;INVM IF INCORRECT MOD#
11046    0B29                  ADD                   UBA         INC            BUSC       REC UPPER WD
11048    0B2A          001D    ADDL                  RREG        ADD   LSL      BUSC       MASK;REC LOWER WD
11050    0B2B                  ADD                   UBA   OPB    AND       RG   POPR NZRO  MASK I/O ERRORS & SKIP IF NOT ZERO
11052                                                                                      RG WILL CONTAIN I/O INFO                (2635)
11053    0B2C           UBB    ADD             ZERO        OPB   ADD       RG             SKIP IF I/O ERRORS;LOWER WD
11055    0B2D                  CAD             CCA   001C  RG    ANDL             F4B   SET CCL IF I/O ERRORS;MASK ERRORS
11057                                                                                      SKIP ERROR CHECK IF COMING FROM FLUSH INSTR,
11058                                                                                       CHANGED SREGB TO RG.                   (2635)
11059    0B2E                  ADD             RSB         UBB   JSZ  IOEE           NZRO  Return or #IOEE for I/O errors
```

```
 RECORD    C.S.          ********** ALU A **********  ********** ALU B **********
   NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11062              ********************************************************************
11063              *                                                                  *
11064              *   64-Bit Extended Floating Point instructions                    *
11065              *                                                                  *
11066              *   Format of a 64-bit floating point number:                      *
11067              *                                                                  *
11068              *   | 0 |  1  -  9  |            10  -  63           |             *
11069              *   -------------------------------------------------             *
11070              *   | S |    EXP    |                MAN              |            *
11071              *   -------------------------------------------------             *
11072              *                                                                  *
11073              *   S   - sign bit                                                 *
11074              *   EXP - exponent, 9 bits                                         *
11075              *   MAN - mantissa, 54 bits                                        *
11076              *                                                                  *
11077              ********************************************************************
11078              *
11079              $LUT INSTR=EXFP:0 010 000 100 00x xxx, ENTRY=EXFP
11080              *
11081   0B2F  EXFP 0007 CIR  ANDL                  ETBL      ADDL LBL                  UBA := CIR.(13:3); UBB := address of jmp tbl
11083   0B30       UBA  UBB  ADD            WRD   0008      ADDL                      YRA := jump target; UBB := bit 12
11085   0B31       UBB  CIR  AND            RAR             ADD                       UBA := bit 12, RAR := jump target from YRA
11087   0B32                 INC                        UBA  JSZ  TRP            ZERO  UBA := 1; If CIR.(12:1) = 0 then old 3 word
11089                                                                                    EFP instr, so jump to unimplemented trap
11090   0B33            UBA  CSR       XR8  RSB   8000      ADDL      XR8             XRA8 := %100000, RSB to jump table entry;
11092                                                                                    XRB8 := %100000
11093   0B34  ETBL           ADD             SF1             JSB  EBND           UNC   0:  Ext Floating Point Add (EADD), set F1
11095   0B35                 ADD             SF3A            JSB  EBND           UNC   1:  Ext Flt Point Subtract (ESUB), set F3
11097   0B36                 ADD                        Z    JSB  EBND XR3       UNC   2:  Ext Flt Point Multiply (EMPY)
11099                                                                                      ; Save Z in XRB3, (Z used by #EMPY as scr)
11100   0B37                 ADD                             JSB  EBND           UNC   3:  Ext Flt Point Divide (EDIV)
11102   0B38       0003      ADDL      SP0A                  JSB  ENEG           UNC   4:  Ext Flt Point Negate (ENEG), SP0A := 3
11104   0B39       0004      ADDL      SP0A                  JSB  ECMP SP1B      UNC   5:  Ext Flt Point Compare (ECMP), SP0A := 4;
11106                                                                                      SP1B := 0
11107   0B3A                 ADD                             JSZ  TRP            UNC   6:  Unimplemented instruction
11109   0B3B                 ADD                             JSZ  TRP            UNC   7:  Unimplemented instruction
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11112           ************************************************************************
11113           *                                                                      *
11114           *   EBND --  Called before doing EADD, ESUB, EMPY and EDIV.            *
11115           *            Bounds check operands ( U and V ) and result ( W ).       *
11116           *            Also adjusts and empties TOS registers.                   *
11117           *                                                                      *
11118           *            Entered with:                                             *
11119           *            RA = operand-2 adr (V) (DB+ relative)                     *
11120           *            RB = operand-1 adr (U)       "                            *
11121           *            RC = result adr    (W)       "                            *
11122           *            XRA3 = 2 (for SR check)                                   *
11123           *                                                                      *
11124           *            Bounds checking for U,V,W is:  E >= DL and E+3 <= S-3     *
11125           *            (the latter check prevents using the stacked              *
11126           *            addresses themselves as operands.)                        *
11127           *                                                                      *
11128           *            Exits with RA,RB = V,U absolute addresses,                *
11129           *            stack emptied so all TOS can be used as scratch,          *
11130           *            SR = 0, CTR = 0 and XRA3 = W absolute address.            *
11131           *                                                                      *
11132           ************************************************************************
11133           *
11134    0B3C   EBND 0003      ADDL     RH                  JSL  ADJS          NF2   Adjust # valid TOS to 3; Slow down jump
11136    0B3D        RA   DB  ADD      RA                  ADD       CTR  CLO       RA := V abs adr; CTR := 0
11138                                                                                 CLEAR OVERFLOW BEFORE GOING TO EADD  (2635)
11139                                                                                 ESUB, EDIV OR EMPY.                  (2635)
11140    0B3E        RC   DB  ADD      XR3       FFFD SM   ADDL      SP3B            XRA3 := W abs adr; SP3B := S - 6 (SR = 3)
11142    0B3F        RA   DL  BNDE               RC   DB   ADD       RH              BNDV if abs(V) < DL; RH := W abs adr
11144    0B40        RB   DB  ADD      RB        SP3B RA   BNDE                      RB := U abs adr; BNDV if abs(V) + 3 > SM - 3
11146    0B41        UBA  DL  BNDE               SP3B UBA  BNDE                      BNDV if abs(U) < DL or abs(U) + 3 > SM - 3
11148    0B42        RH   DL  BNDE               SP3B RH   BNDE            CLSR      BNDV if abs(W) < DL or abs(W) + 3 > SM - 3,
11150                                                                                  SR := ESR := 0
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11152           ***************************************************************
11153           *                                                             *
11154           *             Used by EADD, ESUB, EMPY and EDIV to fetch operands *
11155           *             and split up signs, exponents and mantissa.     *
11156           *                                                             *
11157           *             Entered from #EBND with RA,RB = V and U absolute *
11158           *             addresses and CTR = 0.                          *
11159           *                                                             *
11160           *             Fetches V0,V1,V2,V3 and places them into RA,RB,RC,RD; *
11161           *                     U0,U1,U2,U3 and places them into RE,RF,RG,RH. *
11162           *                                                             *
11163           *             Exits with SP4A = 0 if V = +-0, SP1B = 0 if U = +- 0. *
11164           *                                                             *
11165           ***************************************************************
11166           *
11167    0B43        RA         ADD        SP0A ROD   7FFF       ADDL       SP2B              SP0A := abs(V), read; SP2B := mask for -0
11169    0B44                   ADD                   RB         ADD        SP3B ROD          ; SP3B := abs(U), read
11171    0B45             OPA   ADD        REGN       0004       ADDL       CTR               Store V0 in RA; CTR := 4
11173    0B46        UBA  SP2B  AND        SP4A                  OPB  ADD   REGN              SP4A := V0 w/o sign bit; Store U0 in RE
11175    0B47        0003       ADDL       XR0        UBB  SP2B  AND        SP1B              XRA0 := loop count; SP1B := U0 w/o sign bit
11177    0B48  BNDL SP0A        INC        SP0A ROD   FFFD CTRS  ADDL       CTR               Inc V ptr, read; CTR := CTR - 3
11179    0B49        FFFF XR0   ADDL       XR0        SP3B       INC        SP3B ROD          Decrement loop counter; Inc U ptr, read
11181    0B4A             OPA   ADD        REGN       0004 CTRS  ADDL       CTR               Store Vx in TOS; CTR := CTR + 4
11183    0B4B        UBA  SP4A  IOR        SP4A                  OPB  ADD   REGN              Or in Vx for zero check; Store Ux in TOS
11185    0B4C             XR0   JSB  BNDL            NZRO  UBB  SP1B IOR   SP1B              If XRA0 = 0 then done; Or in Ux for zero chk
11187    0B4D                   JSB  ESUB            F3A        UBB  JSB  EADD SP3B      F1   if F1 then EADD else if F3 then ESUB
11189                                                                                          else break up into partial products
11190                                                                                          SP3B := U zero check word
11191           *    Split mantissa from sign and exponent; insert leading '1';
11192           *    XRA0 := sign, exp(V); XRB0 := sign, exp(U), save X register
11193           *
11194    0B4E  EFC3 003F RA    ANDL                  003F RE    ANDL                          UBA := mantissa of V; UBB := mantissa of U
11196    0B4F       0040 UBA   IORL       RA         0040 UBB   IORL                        RA := man(V) w/ leading 1;
11198                                                                                          UBB := man(U) w/ leading 1
11199    0B50       FFC0 RA    ANDL       XR0              UBB   ADD        RE                XR0A := sign & exp of V;
11201                                                                                          RE := man(U) w/ leading 1
11202    0B51       X          ADD        XR1   CF1  FFC0 RE    ANDL       XR0       NF1     Save X in XRA1, clear F1; XRB0 := sign &
11204                                                                                          exp(U), skip if not from EADD, ESUB
11205    0B52                  ADD                             ADD                    RSB     ; Return if from EADD, ESUB
11207    0B53            CIR   JSB  EDIV       ODD        XR0   ADD        SP2B              JSB if EDIV else fall thru to EMPY
11209                                                                                          SP2B <- SIGN ,EXP (U)                (2635)
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11211           ************************************************************************
11212           *                                                                      *
11213           *  EMPY -- W := U * V                                                  *
11214           *                                                                      *
11215           *  V = RA,RB,RC,RD  (V1 = RA,RB;  V2 = RC,RD)                          *
11216           *  U = RE,RF,RG,RH  (U1 = RE,RF;  U2 = RG,RH)                          *
11217           *                                                                      *
11218           *  The partial products will be summed into XR10, XR11, X and Z.       *
11219           *  XR4 will contain 8 guard bits (for normalization and rounding.)     *
11220           *                                                                      *
11221           *  The following diagram shows the relationships between the partial   *
11222           *  products and how they are summed into the above registers to        *
11223           *  produce the result (W).  There are 4 multiplies done and their      *
11224           *  products will be on UBA, UBB, SP0A and SP3B.  (Each group shown     *
11225           *  represents the contents of one of these registers).  The notation   *
11226           *  to the left of each partial is the naming convention used to        *
11227           *  designate each 16-bit part of that partial.                         *
11228           *                                                                      *
11229           *  V1*U1*2**64   00XX XXXX XXXX XX00                    A0,A1,A2,A3    *
11230           *  V1*U2*2**32  +          XXXX XXXX XXXX XX00          B0,B1,B2,B3    *
11231           *  U1*V2*2**32  +          XXXX XXXX XXXX XX00          C0,C1,C2,C3    *
11232           *  U2*V2        +                XX XXXX XXXX XXXX XX   D0,D1,D2,D3    *
11233           *                                ( DD DDDD DDDD DDDD DD )              *
11234           *                                ( 00 0011 1122 2233 33 )              *
11235           *                              ----------------------------------      *
11236           *  RESULT       00WW WWWW WWWW WWWW GGxx                W0,W1,W2,W3    *
11237           *                                                                      *
11238           *  After producing each partial, it will be summed into a 64-bit       *
11239           *  'pseudo-accumulator' using two 32-bit additions.  The guard bits    *
11240           *  are kept in XRA4.  XRA10, XRB11, X and Z forms the accumulator.     *
11241           *  Upon completion W0 = XRA10, W1 = XRB11, W2 = X and W3 = Z.          *
11242           *                                                                      *
11243           *  The algorithm is:                                                   *
11244           *                                                                      *
11245           *  1.  Form V1*U1                                   << form A >>       *
11246           *  1.1 XR10,XR11,SP0A,SP3B := UBA,UBB,SP0A,SP3B                        *
11247           *                           ; W0,W1,W2,W3 := A0,A1,A2,A3               *
11248           *  2.  Form V1*U2                                                      *
11249           *  2.1 X,Z := X,Z + UBA,UBB     ; sum B0,B1 into W2,W3                 *
11250           *  2.2 XR10,XR11 := XR10,XR11 + carry from 2.1                         *
11251           *                           ; propagate carry to W0,W1                 *
11252           *  2.3 XR4 := SP0A          ; first guard bits are B2                  *
11253           *                                                                      *
11254           *  3.  Form U1*V2                                                      *
11255           *  3.1 X,Z := X,Z + UBA,UBB     ; sum C0,C1 into W2,W3                 *
11256           *  3.2 XR10,XR11 := XR10,XR11 + carry from 3.1                         *
11257           *                           ; propagate carry to W0,W1                 *
11258           *  3.3 XR4 := XR4 + SP0A    ; sum C2 into guard                        *
11259           *  3.4 if carry from 3.3 then increment GCRY                           *
11260           *                           ; carry from guard handled later           *
11261           *  4.  Form V2*U2                                                      *
11262           *  4.1 X,Z := X,Z + LRZ(UBA)    ; sum msbyte D0 into W2,W3             *
11263           *  4.2 XR10,XR11 := XR10,XR11 + carry from 4.1                         *
11264           *                           ; propagate carry to W0,W1                 *
11265           *  4.3 XR4 := XR4 + RLZ(UBA)    ; sum lsbyte D0 into guard             *
11266           *  4.4 if carry from 4.3 then increment GCRY                           *
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11267           *                                  ; carry from guard                  *
11268           *  5.  XR10,XR11,X,Z := XR10,XR11,X,Z + GCRY                           *
11269           *                                                                      *
11270           ************************************************************************
11271     0B54  EMPY       SP4A JSB  EZRO      ZERO       SP1B JSB  EZRO          ZERO  Result is 0 if either V or U = 0
11273           *                    do partial 'A' -- V1 * U1
11274     0B55             RB   ADD       SP4A                RA   ADD       SP3B        Multiplier := V1
11276     0B56             ADD  ADD       SPOA                     REPN                0017  SPOA := 0; Repeat next line 24 times
11278     0B57       RE    UBA  MPAD                          RF   UBB  LINK          DCTR CTR0  Do V1 * U1
11280     0B58       UBA   ADD  ADD       XR10                UBB  ADD       XR11        XRA10,XRB11 := A0,A1
11282     0B59       SPOA  ADD  ADD       X          SP3B     ADD       Z           X,Z := A2,A3
11284           *                    do partial 'B' -- V1 * U2
11285     0B5A             RB   ADD       SP4A                RA   ADD       SP3B        Multiplier := V1
11287     0B5B             ADD  ADD       SPOA                     REPN                0017  SPOA := 0; Repeat next line 24 times
11289     0B5C       RG    UBA  MPAD                          RH   UBB  LINK          DCTR CTR0  Do V1 * U2
11291     0B5D       X     UBA  ADD       X     CTF1  Z      UBB  LINK      Z           Sum B0,B1 into W2,W3
11293     0B5E             XR10 ADD       XR10  F1TC       XR11 LINK      XR11          ... and propogate carry into W0,W1
11295     0B5F       SPOA  ADD  ADD       XR4                     ADD                 XRA4 := B2
11297           *                    do partial 'C' -- V2 * U1
11298     0B60             RF   ADD       SP4A                RE   ADD       SP3B        Multiplier := U1
11300     0B61             ADD  ADD       SPOA                     REPN                0017  SPOA:=XRB9:=0 (for GCRY), rept next 24 times
11302     0B62       RC    UBA  MPAD                          RD   UBB  LINK          DCTR CTR0  Do V2 * U1
11304     0B63       X     UBA  ADD       X     CTF1  Z      UBA  LINK      Z           Sum C0,C1 into W2,W3
11306     0B64             XR10 ADD       XR10  F1TC       XR11 LINK      XR11          ... and propogate carry into W0,W1
11308     0B65             ADD  ADD                                ADD                 Wait for extended register to be available
11310     0B66       SPOA  XR4  ADD       XR4   CTF1       RG   ADD       SP3B        XRA4 (GRD) := XRA4 + C2, set F1 if carry;
11312           *                    do partial 'D' --  U2 * V2
11313                                                                                    MSmultiplier := MS_U2
11314     0B67             RH   ADD       SP4A  F1TC       XR9  LINK      XR9           LSmultiplier := LS_U2; If carry from guard
11316                                                                                      then increment GCRY
11317     0B68             ADD  ADD       SPOA                     REPN                001F  SPOA := 0; Repeat next line 32 times
11319     0B69       RC    UBA  MPAD                          RD   UBB  LINK          DCTR CTR0  Do U2 * V2
11321     0B6A             UBA  ADD                                ADD                 All we need from this MPY is UBA (A0)
11323     0B6B             UBA  ADD   RLZ                          UBA  ADD   LRZ RH           UBA.(0:8) := A0.(8:8); RH.(8:8) := A0.(0:8)
11325     0B6C       UBA   XR4  ADD       SP4A  CTF1       XR9  ADD                         Sum ms (A0) into guard, set F1 if carry;
11327                                                                                      UBB := GCRY
11328     0B6D             UBB  ADD             F1TC       XR3  ADD       Z           If carry from guard then increment GCRY;
11330                                                                                      Restore Z register
11331     0B6E                  ADD                   Z    UBA  LINK                    ; UBB := W3 + GCRY.  Link so that if a carry
11333                                                                                      is produced UBA will = 1, (else 0) and
11334                                                                                      subsequently be added to W2
11335     0B6F       X     UBA  ADD       SPOA  CTF1  UBB  RH   LINK      SP3B        SPOA,SP3B := W2,W3 + ls (A0) + GCRY
11337     0B70             XR10 ADD       RA    F1TC       XR11 LINK      SP1B          ... and propogate carry to W0,W1
11339           *
11340           *  At this point W is contained in RA, SP1B, SPOA and SP4A
11341           *  with 8 guard bits in SP4A.  Now we figure out the sign
11342           *  and exponent of W and go off to normalize and finish.
11343           *
11344     0B71             XR1  ADD       X     CF1                ADD       CTR  CCTX       Restore X register, clear F1;
11346                                                                                      CTR := CTX := 0 (for #ENRM)
11347     0B72             XR0  ADD             F1HB  7FFF XR0  ANDL                        UBA := exp(V); UBB := exp(U)
11349     0B73       UBA   UBB  ADD                           XR0  ADD                     UBA := exp(W) (+512); UBB := sign.exp(U)
11351     0B74       UBB   XR0  XOR             HBF2  C080 UBA  ADDL                      F2 := sign(W) ( sign(V) XOR sign(U) );
11353                                                                                      UBB := (exp(W)+512) - (256/2) + extras
11354     0B75       UBB        ADD       XR0                      JSB  ENRM           UNC   XRA0 := exp(W), Go normalize it
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11357           ************************************************************************
11358           *                                                                      *
11359           *  EDIV -- W := U / V                                                  *
11360           *                                                                      *
11361           *  V1,4 = RA,RB,RC,RD                                                  *
11362           *  U1,4 = RE,RF,RG,RH                                                  *
11363           *                                                                      *
11364           ************************************************************************
11365           *
11366           *         first check if U or V is zero
11367           *
11368   0B76    EDIV    SP1B JSB  EZRO      ZERO       SP4A JSB  EDVZ           ZERO  Divide by zero trap if V = 0 else result is
11370                                                                                    zero if U = 0
11371           *         then shift V and U to left 8 bits
11372           *
11373   0B77        RC       ADD  RLZ            RD         ADD  LRZ
11374   0B78        UBA  UBB  IOR       RC        RB         ADD  RLZ  SP1B
11375   0B79        RC       ADD  LRZ            RB         ADD  LRZ  SP1B
11376   0B7A        UBA  SP1B IOR       RB        RA         ADD  RLZ
11377   0B7B        UBB  SP1B IOR       RA        RH         ADD  LRZ  SP1B
11378   0B7C        RD       ADD  RLZ  RD        RE    RA   SUB
11379   0B7D        RG       ADD  RLZ            UBB        CSL                LBF5       ; Set F5 if mantissa(U) < mantissa(V)
11381   0B7E        UBA  SP1B IOR       SPOA      RF         ADD  RLZ  SP1B
11382   0B7F        RG       ADD  LRZ            RF         ADD  LRZ  SP1B
11383   0B80        UBA  SP1B IOR       RF        RE         ADD  RLZ
11384   0B81        UBB  SP1B IOR       RE        001A       ADDL       CTR     NF5B  ; Set counter for divide
11386   0B82                 ADD        SP4A            XR0   ADD                ICTR
11387   0B83        UBB  XR0  XOR            HBF2  RH         ADD  RLZ  SP3B
11388           *
11389           *         do divide ( U1,U2,U3,U4 ) / ( V1,V2 )
11390           *
11391   0B84             RE   ADD            CF1        RF   REPC
11392   0B85        UBA  RA   DVSB                  UBB  RB   LINK           DCTR CTR0
11393   0B86        RREG      ADD  LSR  RG          RREG       LINK       SP1B        Q:= ( SP3B, SP4A ); R:= ( RG, SP1B )
11395   0B87             SP4A ADD       RF          SP3B       ADD        XR6         RF:= SP4A; XR6:= SP3B
11397   0B88        UBB       CAD       RE          UBA        LINK       SP2B        Q:= Q - 1 (IF REMAINDER < QUOTIENT*(V3,V4)
11399           *
11400           *         CALCULATE IF REMAINDER >= QUOTIENT*( V3, V4)
11401           *
11402           *
11403   0B89                 ADD        SPOA                   REPN                001F
11404   0B8A        RC   UBA  MPAD                  RD   UBB  LINK           DCTR CTR0
11405   0B8B             UBA  ADD       SPOA              UBB  ADD        SP3B
11406   0B8C        SPOA      RSUB      SPOA  CTF1  SP3B       LINK       SP3B
11407   0B8D        SPOA RG   RSUB      RH    F1TC  SP3B SP1B LINK       SP1B CF1
11408   0B8E        UBA       JSB  EDVL       POS   4000       ADDL       XR10        JSB IF REMAINDER >= QUOTIENT*( V3,V4 )
11410           *
11411           *         SINCE REMAINDER < QUOTIENT*( V3, V4), THEN
11412           *         THE CORRECT QUOTIENT := QUOTIENT - 1, &
11413           *         THE CORRECT REMAINDER := REMAINDER + ( V1,V2,V3,V4 ).
11414           *
11415           *
11416   0B8F        SPOA RC   ADD        SPOA  CTF1  SP3B RD   LINK       SP3B
11417   0B90        RH   RA   ADD        RH    F1TC  RB   SP1B LINK       SP1B
11418   0B91             SP2B ADD        RF                 RE   ADD        XR6
11419           *
```

```
 RECORD   C S          ********** ALU A **********  ************** ALU B **************
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11420           *         REPEAT THE SAME DIVIDE TO GET SECOND HALF QUOTIENT
11421           *         BY  ( CORRECT REMAINDER ) / ( V1, V2 )
11422           *
11423    0B92   EDVL RH        ADD             CF1        SP1B REPN              0020
11424    0B93        UBA  RA   DVSB                  UBB  RB   LINK       DCTR CTR0
11425    0B94        UBA       ADD  LSR  XR5         UBB       LINK  XR5
11426           *
11427           *         CALCULATE IF REMAINDER >= QUOTIENT * ( V3,V4 )
11428           *
11429    0B95             SP4A ADD       RG          SP3B      ADD       SP2B
11430    0B96                  ADD       SPOA                  REPN              001F
11431    0B97        RC   UBA  MPAD                    RD   UBB  LINK       DCTR CTR0
11432    0B98             UBA  ADD                         UBB  ADD
11433    0B99        UBA  XR5  RSUB                    UBB  XR5  LINK            CF1
11434    0B9A        UBA       JSB  EDLP       POS           INC                    JSB IF REMAINDER >= QUOTIENT*( V3,V4 )
11436    0B9B             SP2B RSUB                    UBB  RG   LINK       RG
11437    0B9C                  ADD                     UBA       ADD       SP2B        Q:= Q - 1
11439           *
11440           *         CALCULATE THE EXP(W) := EXP(U) - EXP(V) + 256     (NF5B),
11441           *                     OR         := EXP(U) - EXP(V) + 256 - 1 (F5B).
11442           *
11443    0B9D   EDLP RG        ADD       SP4A              XR6  ADD       RE
11444    0B9E        XR0  ADD                  F1HB  7FFF  XR0  ANDL                  ABS EXP(V); ABS EXP(U)
11446    0B9F        UBB  UBA  SUB                         SP4A ADD       SP3B CCTX F5B   EXP(W); SKIP IF F5
11448    0BA0             SP2B ADD       SPOA          UBA  XR10 ADD       SP2B       UNC   SPOA:= W3;
11450    0BA1                  ADD       SP4A          FFC0 UBB  ADDL      SP2B             ;SP2B:=EXP(W)
11452           *
11453           *         SHIFT W RIGHT 4 BITS
11454           *
11455    0BA2             RE   ADD                               RF   REPN              0003
11456    0BA3        UBA       QASR      RA            UBB       LINK      SP1B DCTR CTR0
11457    0BA4             SP2B ADD       XR0                     JSB  ENRM CTR       UNC   XRA0 := exp(W); CTR := 0, go normalize it
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
11460	0BA5	EDVZ		STA	ADD		RD	SF3A		SP1B	JSB	EZRO			ZERO	RD := status, set F3 (for #EZRO rtn;
11462																If U = 0 then write zero to W
11463	0BA6			XR3	ADD			WRD	003F	RE	ANDL					Write at (W); Strip leading one from U0
11465	0BA7		UBB	SP2B	ADD			DATA			ADD					Merge sign & exp into W0 and write it
11467																FIXED FOR CORRECT REG. (2635)
11468	0BA8			RF	ADD			DATA		UBA	ADD			CCA		Write W1; Set CCA on W0
11470	0BA9			RG	ADD			DATA			ADD					Write W2
11472	0BAA			RH	ADD			DATA			ADD					Write W3
11474	0BAB	VZRO	000A		ADDL		SPOA		RD	RD	ADD	LSL		CLO	NEG	SPOA := %12 (div by zero parm); Skip if user
11476																traps are enabled, clear overflow
11477	0BAC			XR3	ADD						ADD			SOV	NEXT	UBA := W abs adr; If user traps are not
11479																enabled then set overflow bit & NEXT
11480	0BAD				JSZ	UTRP		UNC	UBA	DB	SUB		RA	INSR		Go to user trap routine; Push W adr onto stk

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11483           ************************************************************************
11484           *                                                                      *
11485           *     ESUB -- W := U - V                                               *
11486           *                                                                      *
11487           ************************************************************************
11488           *
11489     0BAE  ESUB          ADD           CF3A  RA   XR8  ADD       RA               Clear F3; Complement the sign of V and do an
11491                                                                                    EADD
11492           ************************************************************************
11493           *                                                                      *
11494           *    EADD -- W := U + V                                                *
11495           *                                                                      *
11496           ************************************************************************
11497           *
11498           *      make sure neither U nor V is -0
11499           *
11500     0BAF  EADD          ADD                 SP3B SP4A IOR                        ; UBB := 0 iff both U and V are zero
11502     0BB0        SP4A    ADD       SPOA NZRO       UBB  JSB  EZRO          ZERO  Skip if V <> 0; If both are 0 result is 0
11504     0BB1                ADD       RA              SP1B ADD                NZRO  Make sure V is +0; Skip if U <> 0
11506     0BB2                ADD       SP4A                 ADD       RE            ; Make sure U is +0
11508           *
11509           *      calculate abs(V) - abs(U) to see if V > U
11510           *
11511     0BB3        RC  RG  SUB               CTF1  RD   RH   LINK                  Do ls(V) - ls(U), borrow to F1
11513     0BB4        RA  RA  ADD  LSR              RE   RE   ADD  LSR                Get rid of sign bits
11515     0BB5        UBA UBB SUB               F1TC  RB   RF   LINK                  Do ms(V) - ms(U)
11517           *
11518           *   If U was greater than V (V-U was negative) then swap V
11519           *   and U so that U is always the smaller operand
11520           *
11521     0BB6        FFFC    ADDL                    UBA  XR8  REPC            NCRY  UBA := -4; UBB := (V - U) + 8000
11523                                                                                  skip swapping if V - U >= 0
11524     0BB7            UBA INC               CRRY            ADD         DECN      Roll stack 4 times
11526           *
11527           *   Calculate exp(V) - exp(U); F2 := sign(V) XOR sign(U)
11528           *   Strip Exponents, insert leading 1's
11529           *
11530     0BB8                JSB EFC3          SF1   RA   RE   XOR         HBF2      Go split exp, man and insert leading 1's,
11532                                                                                    set F1 so we can get back;
11533                                                                                    F2 := sign (U) <> sign (V)
11534     0BB9            XRO ADD               F1HB  7FFF XRO  ANDL                  UBA := exp (V); UBB := exp (U)
11536     0BBA        UBA UBB SUB                                 ADD         CTR       UBA := exp(V) - exp(U); CTR := 0 for #ENRM
11538     0BBB        0E00    ADDL                    UBA  UBA  ADD  LSL  CTX           UBA.(2:8) := 56; CTX := exp diff (D.EXP)
11540           *
11541           *   Check for operands = 0 or difference in exponents > 56
11542           *
11543     0BBC            SP1B JSB  EVAN        ZERO  UBA  SREG SUB               CRRY  V' answer if orig U = 0; skip if exp diff is
11545                                                                                    <= 56
11546     0BBD        SPOA    JSB  EVAN         ZERO            JSB  EVAN          UNC   V' answer if original V = 0; V' answer if
11548                                                                                    difference in exponents > 56
11549           *
11550           *   If sign V <> sign U then U := -U
11551           *
11552     0BBE            RG  ADSB      SPOA CTF1       RH   LINK       SP3B          SPOA,SP3B := ls(U) (for following QASR)
11554     0BBF            RE  ADSB           F1TC       RF   LINK
11555           *
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11556            *   Shift U right to align with V
11557            *
11558    0BC0            UBA  ADD       RE            UBB  REPC      SP1B DCTX CTX0  RE,SP1B := ms(U); Skip if D.EXP = 0
11560    0BC1            UBA  QASR      RE            UBB  LINK      SP1B DCTX CTX0  Shift ms(U),ls(U) (RE,SP1B,SPOA,SP3B) right
11562                                                                                    and decrement D.EXP until D.EXP = 0
11563            *     now add'em up   (RA,SP1B,SPOA,SP3B) := man(V) + man(U)
11564            *
11565    0BC2       SPOA RC   ADD       SPOA CTF1  SP3B RD   LINK      SP3B            SPOA,SP3B := ls(U) + ls(V), carry to F1
11567    0BC3       RA   RE   ADD       RA   F1TC  RB   SP1B LINK      SP1B            RA,SP1B := ms(V) + ms(U) + previous carry
11569            *
11570            *   F2 := sign (W), check for 0 result, normalize
11571            *
11572    0BC4       SPOA UBA  IOR                  SP3B UBB  IOR            CCTX       Start 'OR'ing all parts of W;
11574                                                                                    CTX := 0 for #ENRM
11575    0BC5            XR0  ADD           HBF2  UBA  UBB  IOR                        F2 := sign(W) which will always = sign(V');
11577                                                                                    UBB := 0 iff result (W) = 0
11578    0BC6       UBA  XR0  ADD  LSR XR0            UBB  JSB  EZRO           ZERO  Exp(W) will always = exp(V), get rid of sign
11580                                                                                    bit for normalization routine;
11581                                                                                  goto #EZRO if W = 0 else fall thru to #ENRM
```

```
 RECORD  C.S          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11583          ********************************************************************
11584          *                                                                  *
11585          *      ENRM -- Normalize result of EADD, ESUB, EMPY and EDIV       *
11586          *              Merge sign, set OVF, UNF indicators if neccessary   *
11587          *              Write result back to memory                         *
11588          *              Set condition codes                                 *
11589          *                                                                  *
11590          *      Entered from EADD/ESUB/EMPY/EDIV with                       *
11591          *              F2 = sign of W                                      *
11592          *              RA,SP1B,SP0A,SP3B = W0,W1,W2,W3                     *
11593          *              XRA0 = exponent of W, SP4A = guard bits             *
11594          *              CTR = CTX = SR = 0                                  *
11595          *              XRA3 = absolute address to store result at          *
11596          *                                                                  *
11597          ********************************************************************
11598          *
11599          *    shift W left to BIT8, round, shift W right to BIT9
11600          *
11601  0BC7  ENRM      RA   ADD          BIT8       SP1B REPC                     UBA := W0, skip if already aligned to BIT8;
11602                                                                               UBB := W1
11603  0BC8            UBA  QASL   RA    BIT8       UBB  LINK      SP1B DCTX      Shift W0,W1,W2,W3,W4,W5,W6,W7 until BIT8;
11604                                                                               CTX := exp adj needed because of shifting
11605  0BC9      SP0A       INC          SP0A CTF1  SP3B      LINK      SP3B      Add 1 to W3,W4 to round, propagate carry
11606  0BCA          RA     ADD               F1TC       SP1B LINK                 into upper two words (W0,W1)
11607  0BCB          UBA    QASR   RA                    UBB  LINK      SP1B      and shift W0,W1,W2,W3 right to align to BIT9
11608          *
11609          *    add in exponent adjustment, merge sign, check OVF, UNF
11610          *
11611  0BCC          STA  ADD          RF                 CTRS ASR                RF := STATUS; Shift exp adjustment right
11612  0BCD      RA  XR0  ADD                             UBB  ASR                Add exp W into W0, (W0 will have a 1 in the
11613                                                                               exp bits if no carry from rounding, else
11614                                                                               it will have a 2.  Thus we add 1 to the
11615                                                                               exp for the previous shift right and add 1
11616                                                                               if rounding produced a carry into the exp
11617                                                                               bits.
11618                                                                            ; UBB := exp adjustment from the normalizing
11619                                                                               left shifts, this adjustment is now
11620                                                                               aligned with the exp field in W0.
11621  0BCE      UBA  UBB  CSL        SP4A HBF2  UBA  UBB  ADD       RA   F2HB      UBA := exp(W) + exp adj &csl(1), if UBA is
11622                                                                               odd then we have an OVF or UNF, F2 set for
11623                                                                               UNF.
11624                                                                            ; UBB := W0 + exp adj, store into W0, merge
11625                                                                               sign(W) into W0
11626          *   write W back to memory, set CC, check for W = 0
11627  0BCF          XR3  ADD       RH   WRD                ADD                  RH := W absolute address
11628  0BD0      RA       ADD            DATA       RA   ADD            CCA      Write W0; Set CCA on W0
11629  0BD1          SP1B ADD            DATA  SP3B       ADD       RG           Write W1; RG := W3
11630  0BD2      SP0A     ADD            DATA  SP3B SP1B IOR                      Write W2; UBB := W1 'LOR' W3
11631  0BD3          RG   ADD            DATA  UBA  UBB  IOR            DCC      Write W3; UBB := W1 lor W2 lor W3, set DCC
11632  0BD4      UBB  SP4A IOR            NZRO       SP4A JSB  EFOV       ODD  Skip if abs(W) <> 0; #EFOV if over/underflow
11633  0BD5                JSB  EFOV     UNC                ADD            SF2  NEXT  Underflow if W = 0 (that is not allowed),
11634                                                                               jump w/ F2 set; Else all done
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11651          ************************************************************************
11652          *                                                                      *
11653          *    Over/Underflow Trap                                               *
11654          *                                                                      *
11655          *    Entered with F2 = 1 for underflow, RF = status, SR = 0            *
11656          *                                                                      *
11657          *    If user traps are not enabled then set overflow status bit,       *
11658          *    else push W address back onto stack, push parameter onto          *
11659          *    stack, clear overflow bit and go to the user trap handler.        *
11660          *                                                                      *
11661          ************************************************************************
11662          *
11663   0BD6   EFOV 0008       ADDL       SPOA       RF   RF   ADD  LSL       CLO        SPOA := %10 (overflow parm); Clear overflow
11665                                                                                      bit.                          (2635)
11666   0BD7        UBA        ADD             NF2   UBB       ADD                 NEG   SKIP IF OVERFLOW; SKIP IF USER TRAPS (2635)
11668                                                                                      ARE ENABLED                   (2635)
11669   0BD8        UBA        ADD             CCA               ADD            SOV  NEXT  ; If user traps not enabled then NEXT with
11671                                                                                      overflow bit set AND CC SET TO CCG (2635)
11672   0BD9                   ADD             NF2   RH   DB     SUB       RA    INSR       Skip if overflow; Place W address (DB+ rel)
11674                                                                                      back on TOS
11675   0BDA        SPOA       INC        SPOA                        JSZ  UTRP            UNC   If underflow then parm := %11; Go to user
11677                                                                                        trap routine
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11679           ************************************************************************
11680           *                                                                      *
11681           *    ENEG -- V := -V                                                   *
11682           *                                                                      *
11683           *            If V <> 0 then complement sign of V and set CCG           *
11684           *            or CCL, else just set CCE.                                *
11685           *                                                                      *
11686           *            The algorithm is to fetch each Vx until a non-zero        *
11687           *            Vx is found.  Then complement the sign of VO and          *
11688           *            write it back to memory (or TOS register.)  Set CC        *
11689           *            based on new VO.  If all Vx equal zero then set CCE.      *
11690           *                                                                      *
11691           *    Entered with SPOA = 3                                             *
11692           *                                                                      *
11693           ************************************************************************
11694           *
11695    0BDB   ENEG RA       JSZ  PULM      SRZ             CAD       SP1B             UBA := RA, pull 1 TOS if SR = 0; SP1B := -1
11697    0BDC        UBA  DB  ADD       RA        SR   SM    ADD       SP3B             RA := abs adr of operand; SP3B := S
11698    0BDD                 ADD                                    ADD        EPOP     Wait for reg store; Decrement ESR and ENAMER
11701    0BDE   NEGL SPOA SP1B JSBS FINI      ZERO  RA   SP1B INC       RH   ROD        If x = 3 then V = 0; RH := E + x, read Vx
11703    0BDF        UBB  SM   CAD                  SP3B UBB  BNDE      CTR  TICB       If TOS registers if E > SM and E <= S;
11705                                                                                    BNDV if E > S
11706    0BE0        RH   DL   BNDE                 SP1B INC       SP1B       NZRO     BNDV if E < DL; Increment x, skip if x <> 0
11708    0BE1                 ADD                 8000 REGN ADDL    SP2B                ; SP2B := -(VO)
11710    0BE2        RA       ADD           WRD       SREG JSB  NEGL             ZERO  YREGA := VO abs adr; if Vx = 0 then examine
11712                                                                                    V(x+1), else fall thru to write negated
11713                                                                                    value back
11714    0BE3        RA   SM   CAD                  SP3B RA   SUB       CTR  TICB       CTR := 17 if not (E > SM and E <= S) else
11716                                                                                    TOS REGN#
11717    0BE4             SP2B ADD       REGN DATA  SR   SP2B IOR            CCA  NEXT   write -VO back to TOS or memory; set CCA on
11719                                                                                    VO (ior with SR to prevent CCE), NEXT
11720    0BE5   FINI          ADD                            ADD            CCA  NEXT   ; VO was zero so just set CCE, NEXT
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11723           ********************************************************************
11724           *                                                                  *
11725           *    Extended Compare (ECMP) Instruction                           *
11726           *                                                                  *
11727           *    U is compared to V and the condition code is set to CCG, CCL  *
11728           *    or CCE to indicate whether U is greater than, less than or    *
11729           *    equal to V.                                                   *
11730           *                                                                  *
11731           *    Entered with SPOA = 4 and SP1B = 0                            *
11732           *                 TOS   = V (DB-rel)                               *
11733           *                 TOS-1 = U (DB-rel)                               *
11734           *                                                                  *
11735           ********************************************************************
11736           *
11737    0BE6   ECMP       RB   JSZ  PULM      SRL2                JSZ  PUL2          SRZ   Pull 2 if SR = 0 else pull 1 if SR = 1
11739    0BE7        UBA   DB   ADD                     RA   DB    ADD       RA              UBA := abs(U); RA := abs(V)
11741    0BE8              UBA  JSB  EFCH      UNC         UBA   ADD       RB   EPP2        Fetch V0 & U0; RB := abs(U), pop 2
11743    0BE9        RH    CIR  IOR            CCA   RH    UBA   XOR                 POS   Set CCA on U0 (CIR prevents CCE);
11745                                                                                       Skip if signs are the same
11746    0BEA              RH   ADD            HBF2                ADD                 NEXT  Set F2 if U0 negative; NEXT if signs differ
11748    0BEB   CMPL SPOA  SP1B SUB       RH         RH    SP4A  SUB             CCZ   CRRY  RH := # of words left to compare; Set CCZ on
11750                                                                                       Ux - Vx (CCG or CCE), skip if U >= V
11751    0BEC              UBB  JSB  DONE      NZRO                CAD       SP1B CCA       Go finish up if result <> 0;
11753                                                                                       SP1B := -1 & set CCL if U < V
11754    0BED              RB   JSB  EFCH      UNC         RH    JSZ  NEXT            ZERO  NEXT if no more words left to compare, else
11756                                                                                       go get the next pair
11757    0BEE                   JSB  CMPL      UNC               ADD                       Go back to compare next pair
11759    0BEF   DONE            JSZ  NEXT      NF2               ADD       F2HB            NEXT if both were positive; Bit 0 := F2
11761    0BF0                   ADD                        UBB   SP1B XOR       CCA   NEXT  ; Flip CC if both were negative, NEXT
11763           *
11764           *                        Fetch Ux and Vx
11765           *
11766    0BF1   EFCH SR    SM   ADD       RH          UBA   SP1B ADD       SP3B ROD       RH := S; SP3B := abs Ux, read Ux
11768    0BF2        UBB   SM   CAD                    UBA   UBB  BNDE      CTR  TICB      TOS check:  (if (E > SM and E <= S) then
11770                                                                                       CTR := tos# else !F); BNDV if E > S
11771    0BF3        RA    SP1B ADD       RH    ROD   SP3B  DL   BNDE                    RH := abs Vx, read Vx; BNDV if abs Ux < DL
11773    0BF4        UBA   SM   CAD                    RH    UBA  BNDE      CTR  TICB      TOS check:  (if (E > SM and E <= S) then
11775                                                                                       CTR := tos# else !F); BNDV if E > S
11776    0BF5        RH    DL   BNDE                         REGN ADD       RH             BNDV if abs V < DL; RH := Ux
11778    0BF6              REGN ADD       SP4A                SP1B INC       SP1B      RSB  SP4A := UBA := Vx; Inc word ptr, RSB
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11781           ************************************************************************
11782           *                                                                      *
11783           *   EZRO -- Called when result (W) is zero, (from EADD/ESUB, EDIV,     *
11784           *           EMPY.)                                                     *
11785           *                                                                      *
11786           ************************************************************************
11787           *
11788   0BF7  EZRO      XR3  ADD            WRD             REPN              0003  Start write at (W); Repeat next line 4 times
11790   0BF8                 ADD            DATA            ADD          DCTR CTR0  Write zeroes; Set CCE
11792   0BF9                 JSB  VZRO      F3A             ADD          CCA  NEXT  Go back to EDVZ routine if F3; Else NEXT
11794                                                                                 with CCE set
11795           *
11796           ************************************************************************
11797           *                                                                      *
11798           *   EVAN -- Called when V will be the answer (from EADD/ESUB.)         *
11799           *                                                                      *
11800           ************************************************************************
11801           *
11802   0BFA  EVAN      XR3  ADD            WRD   003F RA   ANDL                      Write at (W); Strip leading one from W0
11804   0BFB      UBB   XR0  ADD            DATA            ADD                       Merge sign & exp into W0 and write it
11806   0BFC            RB   ADD            DATA      UBA   ADD          CCA          Write W1; Set CCA on W0
11808   0BFD            RC   ADD            DATA  RB  RC    IOR                       Write W2; Merge W2, W1
11810   0BFE            RD   ADD            DATA  RD  UBB   IOR          DCC  NEXT    Write W3; Set DCC on W1,W2,W3, NEXT
11812   0BFF                 ADD                            ADD                       NOP to keep system happy              (2555)
11814           %0C00
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11816            ***************************************************************
11817            *                                                             *
11818            *                  Look-up Table for I/O Instructions         *
11819            *                                                             *
11820            *                     All are two-word instructions           *
11821            *                                                             *
11822            *                  0                                  15      *
11823            *                  ---------------------------------------    *
11824            *       word(M)    |              %20302                  |    *
11825            *                  ---------------------------------------    *
11826            *       word(M+1)  |              I/O op code             |    *
11827            *                  ---------------------------------------    *
11828            *                                                             *
11829            ***************************************************************
11830            *
11831   0C00  IOBS           JSB  SHIO      SF1   01FF RB   ANDL      RG              0. SIOP; set F1 for SHIO; RG := MCDEV#
11833                                                                                 START I/O PROGRAM
11834   0C01                 JSB  SHIO      UNC   01FF RA   ANDL      RG              1. HIOP; RG := MCDEV#
11836                                                                                 HALT I/O PROGRAM
11837   0C02                 ADD                                JSZ  TRP        UNC   2. RIOC  --  unimplemented
11839                                                                                 READ I/O CHANNEL
11840   0C03                 ADD                                JSZ  TRP        UNC   3. WIOC  --  unimplemented
11842                                                                                 WRITE I/O CHANNEL
11843   0C04                 ADD                                JSZ  TRP        UNC   4. ROCL  --  unimplemented
11845                                                                                 ROLL-CALL COMMAND
11846   0C05                 ADD                                JSZ  TRP        UNC   5. IOCL  --  unimplemented
11848                                                                                 I/O CLEAR COMMAND
11849   0C06                 JSB  SHIO      SF3A  01F8 RA   ANDL      RG              6. INIT, set F3 for SHIO; RG := MOD, CHAN#
11851                                                                                 INITIALIZE I/O CHANNEL
11852   0C07                 ADD                                JSZ  TRP        UNC   7. MCS   --  unimplemented
11854                                                                                 MEMORY COMMAND and STATUS COMMAND
11855   0C08                 ADD                                JSZ  TRP        UNC  10. SEML  --  unimplemented
11857                                                                                 SEMAPHORE LOAD
11858   0C09       0001 XR15 IORL      XR15           RA   JSL  DMPX RH       UNC  11. STRT, set load bit, RH := cold ld mcdev#
11860                                                                                 WARMSTART COMMAND
11861   0C0A                 ADD                          RA   JSZ  DMPE RH       UNC  12. DUMP, RH := dump load mcdev #
11863                                                                                 MEMORY DUMP COMMAND
11864   0C0B                 JSB  RCMD      UNC           RA   ADD          HBF2       13. RIOA; Set F2 if global command
11866                                                                                 READ I/O ADAPTER
11867   0C0C                 JSB  WCMD      UNC           RA   INC                     14. WIOA; UBB := 1 (# TOS required - 2)
11869                                                                                 WRITE I/O ADAPTER
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11871           ********************************************************************
11872           *                                                                  *
11873           *               Start of 2-word I/O Instructions                   *
11874           *                                                                  *
11875           *                1.  Empty RG, go BUSY                             *
11876           *                2.  Calculate entry into jump table               *
11877           *                3.  Get one valid TOS, two if SIOP                *
11878           *                4.  Put DRTBANK, OFFSET into BKX3, XRA1           *
11879           *                5.  If invalid instruction then trap              *
11880           *                    else if interrupt pending then handle         *
11881           *                    else RSB to jump table entry                  *
11882           *                                                                  *
11883           *               Entered from #RSW with no TOS adjustments          *
11884           *               done and SP1B = 2.                                 *
11885           *                                                                  *
11886           ********************************************************************
11887           *
11888    0C0D   IOLT      STA  JSZ  TRP6      POS            ADD       BKX3           Mode violation?  BKX3 := 0
11890    0C0E       0C02      ADDL      RH             OPB  JSZ  PSHM SP2B      SR7    RH := go busy cmd (in case of #PSHM);
11892                                                                                    SP2B := NIR (opcode), empty RG if SR = 7
11893    0C0F       0007      ADDL      RH             UBA  ADD             BUSC        RH := 7 (for DRT & #SHIO); Go BUSY
11895    0C10       C001      ADDL      SP0A      IOBS SP2B ADDL LBL  SP1B             SP0A := !C001 (for #SHIO); SP1B := jump targ
11897    0C11            UBA  ADD            CCA        OPB  JSZ  PULM          SRZ    Set CCL; (OPB forces BUSC completion), pull
11899                                                                                    one TOS if SR = 0
11900    0C12            RH   INC       RG   ROB3       SP2B ADD                NZRO   Read DRTBANK @8; Skip if not SIOP instr
11902    0C13                 ADD                                JSZ  PULM          SRL2   ; Pull one more TOS if SR < 2 and SIOP
11904    0C14            RG   INC            ROX3  000D       ADDL      SP3B          Read DRTOFFSET @9; SP3B := # valid instr + 1
11906    0C15            SP1B ADD            WRX3       OPB   ADD       BKX3 CF2      YRGA := jump table address; BKX3 := DRTBANK,
11908                                                                                    clear F2 (for WIOA)
11909    0C16            OPA  ADD       XR1       4000       ADDL      SP2B          XRA1 := DRTOFFSET; SP4A := mask for DRT3
11911    0C17       RA        ADD  LSR  SP4A RAR   SP3B SP2B JSZC TRP                 SP4A := mcdev# &lsr(1); Trap if invalid inst
11913    0C18                 JSZ  IRDN      MSGI       RA   ADD       BKX5          Handle MSGINT; BKX5 := mcdev#
11915    0C19            SP4A ADD  LSR  SP4A RSB             ADD             SF4B   SP4A := mcdev# &lsr(2) for INIT instruction,
11917                                                                                    RSB to jump table entry; Set F4 (for STRT)
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11919           ***************************************************************
11920           *                                                             *
11921           *              Entry for SIOP, HIOP, INIT                     *
11922           *                                                             *
11923           *                  1.  Translate logical IMB# to physical MOD#*
11924           *                  2.  Place CDEV# into BKX5 for use by #IOMS *
11925           *                  3.  Read DRT for specified MCDEV#,         *
11926           *                      set F2 if abort bit DRT3.(2:1) is set  *
11927           *                  4.  Right justify CHNL# into SP1B for INIT *
11928           *                  5.  Exit to #SIOP, #HIOP or #INIT          *
11929           *                                                             *
11930           *              Entered with F1 set if SIOP, F3 set if INIT,   *
11931           *              RG = mcdev# (mc# if INIT), SP4A = mcdev# &lsr(2),*
11932           *              RH = 7, SPOA = !C001, XRA1 = DRTOFFSET         *
11933           *                                                             *
11934           ***************************************************************
11935           *
11936    0C1A   SHIO 001E SP4A ANDL      SP4A                JSL  LTOP BKX4      UNC   SP4A.(11:4) := channel #; BKX4 := 0,
11938                                                                                   set up physical module # in XRA9 for #IOMS
11939    0C1B        RG   RG   INC  LSL            RG   RH    JSZC NCHL           NCRY  UBA := DRT2 address; Trap if chan# = 0 (only
11941                                                                                   works for mod# = 0)
11942                                                                                  *
11943                                                                                   ***** CHNLO CK SB DELETED FOR PR IMBI *****
11944                                                                                  *
11945    0C1C        UBA  XR1  INC       SP4A       007F RG   ANDL      BKX5      F1    SP4A := DRT3 address;
11947                                                                                   BKX5 := cdev#, skip if SIOP
11948    0C1D                  JSB  INIT      F3A        SP4A ADD  LSR  SP1B SF5B       JSB if INIT instr; SP1B.(12:4) := channel#,
11950                                                                                   set F5
11951    0C1E   SIO1       SP4A ADD            ROX3       BKX3 ADD       BKX6           Read DRT3 (status) using semaphore (2319)
11953    0C1F               CAD              DATA            ADD                       Re-write semaphore as 'locked'
11955    0C20               OPA  JSBI BIDE     ZERO            ADD                       Loop until semaphore free for use
11957    0C21                    ADD                    UBA      CAD       BKX6          Save DRT3 in BKX6 while semaphore = -1
11959    0C22        SPOA OPA  AND       RH   HBF2            JSB  HIOP           NF1   RH := DRT3.(0:2,15:1), set F2 if bit 0 = 1;
11961                                                                                   HIOP if not SIOP
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11963           ***************************************************************************
11964           *                                                                         *
11965           *                    SIOP CMD--        0       6 7             15       *
11966           *                                      ----------------------------       *
11967           *                           S-1        |       |       MCDEV#    |       *
11968           *                                      ----------------------------       *
11969           *                           S          |     CHAN PGM POINTER    |       *
11970           *                                      ----------------------------       *
11971           *                                                                         *
11972           *                    1.  If DRT3.(2:1) = 1, exit with CCL.  (Set by       *
11973           *                        #IOLT.)                                          *
11974           *                    2.  If DRT3.(0:2) = 00, or DRT3.(0:2) = 01 with      *
11975           *                        DRT3.(15:1) = 1, set CCE.                        *
11976           *                    3.  Else set CCG.                                    *
11977           *                    4.  CCE only when the channel program is halted,     *
11978           *                        or if HIOP has been issued but not yet           *
11979           *                        serviced and the channel is in WAIT state.       *
11980           *                    5.  Then the channel program pointer is placed       *
11981           *                        in DRT0 and set DRT3 := !C000.  Send SIOP        *
11982           *                        command to the channel to request CSRQ on        *
11983           *                        behalf of the device.                            *
11984           *                                                                         *
11985           ***************************************************************************
11986           *
11987    0C23 . SIOP 2000 SREG ANDL                    5000      ADDL      SP3B            Test DRT3.(2:1); SP3B := SIOP command
11989    0C24        UBA       JSB  NEXS      NZRO  C000 RH   ANDL      SP1B            NEXT if DRT3.(2:1) = 1; Mask DRT3.(0:2)
11991    0C25        UBB  SP2B XOR            NZRO             ADD             CCA      Skip if DRT3.(0:2) <> 01; Set CCE
11993    0C26             OPA  JSB  SIOE      ODD        SP1B JSB  SIOE            ZERO  #SIOE if DRT3.(0:2) = 00 or
11995                                                                                     if DRT3.(0:2) = 01 and DRT3.(15:1) = 1
11996    0C27                  JSB  NEXS      UNC              INC             CCA      NEXT (no change); Set CCG
```

```
RECORD    C S.          ********** ALU A **********  ********** ALU B **********
  NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

11999             ********************************************************************
12000             *                                                                  *
12001             *                   HIOP CMD--       0          7          15     *
12002             *                                     ------------------------     *
12003             *                             S       |          |    MCDEV#  |     *
12004             *                                     ------------------------     *
12005             *                                                                  *
12006             *                   1.  Set CCG if DRT3.(0:2,15:1) = <010> or <100>. *
12007             *                       (Send HIOP command with DRT3 := !4000 only *
12008             *                       if <100>.)                                 *
12009             *                   2.  Else set CCE.  Also send HIOP command only *
12010             *                       for <101>, <110> & <111> with DRT3 := !4001 *
12011             *                   3.  CCE indicates that the channel program was *
12012             *                       halted or in a WAIT state when the         *
12013             *                       instruction was executed, or had been issued *
12014             *                       SIOP earlier but not yet serviced.         *
12015             *                   4.  Then set DRT3 := !4001 so it looks like an *
12016             *                       HIOP request from the WAIT state.  Then    *
12017             *                       issue HIOP command to the channel to request *
12018             *                       service on behalf of the device.           *
12019             *                                                                  *
12020             ********************************************************************
12021             *
12022   0C28  HIOP RH    SP2B JSBS NEXS      ZERO       SP2B ADD             CCA        NEXT (no change) if <010>; Set CCG
12024   0C29               ADD              CCA   UBB   UBB  JSBI NEXS SP1B      NF2  NEXT with CCE if <000>, <001> or <011>;
12026                                                                                    SP1B := !8001
12027   0C2A       000F OPA  ANDL                 7000       ADDL      SP3B            Mask DRT3.(12:4); SP3B := HIOP command
12029   0C2B       UBA  SP2B INC        SPOA      RH   SP1B JSBS SIOE          ZERO  !4000 + DRT3.(15:1);  SIOE if <101> with CCE
12031   0C2C       UBB       JSBI SIOE      ZERO                INC             CCA  SIOE if <100> with CCG
12033   0C2D       0002 SP2B ADDL       SPOA                   ADD             CCA  Send HIOP if <110> or <111> with CCE,
12035                                                                                  set up to set DRT3 to !4001
12036             *
12037             ********************************************************************
12038             *                                                                  *
12039             *         #SIOE is used by SIOP and HIOP to write DRT3 status      *
12040             *               (and DRT0 ptr if SIOP)                             *
12041             *                                                                  *
12042             ********************************************************************
12043             *
12044   0C2E  SIOE FFFD       ADDL                        ADD                        UBA := -3
12046   0C2F            SP4A ADD              WRX3  UBA   SP4A ADD             WRX3 NF1  Write to DRT3 addr;
12048                                                                                    Write to DRT0 addr, skip if HIOP
12049   0C30       0007 RG   ANDL       RG                RA   ADD             DATA      RG := dev#; Store chan pgm pointer to DRT0
12051   0C31                 ADD                                 JSL  IOMS          UNC  ; Send SIOP/HIOP command
12053   0C32       SPOA      CAD              DATA               ADD                     Write !C000/!4001 to <DRT3> (must be done
12055                                                                                    after #IOMS since this clears semaphore)
```

```
 RECORD   C S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12057           ************************************************************************
12058           *                                                                      *
12059           *          #NEXO is the common exit point for all I/O instructions     *
12060           *                except STRT and DUMP.                                 *
12061           *                                                                      *
12062           *                     1.  Increment P and read next instruction        *
12063           *                     2.  Pop 2 TOS elements if F5 is not set, else pop*
12064           *                         only 1 (WIOA does its own 1st pop)           *
12065           *                     3.  Go UNBUSY                                    *
12066           *                     4.  Do NEXT                                      *
12067           *                                                                      *
12068           ************************************************************************
12069           *
12070  0C33     NEXO           ADD             EPOP       P    INC       P    RONP F5B   EPOP; Set P and read instr after 2nd I/O
12072                                                                                       word, skip if from SIOP or RIOA
12073  0C34          0400      ADDL                                    ADD            EPOP       UBA := go UNBUSY command; Do 2nd EPOP
12075  0C35                    ADD                                UBA  ADD            BUSC       ; Send go UNBUSY message
12077  0C36                    JSZ  NEXT       UNC                     ADD                       NEXT in one clock
12079           *
12080           ************************************************************************
12081           *                                                                      *
12082           *          #NEXS is used to clear a semaphore of DRT3 when it is       *
12083           *                desired to change the contents.                       *
12084           *                                                                      *
12085           *                     1.  The contents of BKX6 are written to the      *
12086           *                         address contained in < BKX4.SP4A >.          *
12087           *                     2.  Control is then passed to #NEXO for          *
12088           *                         instruction termination.                     *
12089           *                                                                      *
12090           ************************************************************************
12091           *
12092  0C37     NEXS           ADD                             SP4A ADD            WRX3       Set up address...
12094  0C38                    JSB  NEXO       UNC             BKX6 ADD            DATA       ...and write back DRT3 unchanged
12096           *
12097           ************************************************************************
12098           *                                                                      *
12099           *          #BIDE is used to wait for a period of time.  This is        *
12100           *                used within the semaphore-wait loops of the           *
12101           *                SIOP and HIOP instructions, and prevents the CPU      *
12102           *                from accessing the DRT semaphore location so          *
12103           *                frequently that the channel program processor is      *
12104           *                unable to gain access to clear the semaphore.         *
12105           *                                                                      *
12106           ************************************************************************
12107           *
12108  0C39     BIDE           ADD                             REPN                0020   Waste just over 32 clocks
12110  0C3A                    ADD                             ADD            DCTR CTRO   Don't do anything except waste some time
12112  0C3B                    ADD                             JSB  SIO1           UNC    ; Return to check semaphore again
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12115          ********************************************************************************
12116          *                                                                              *
12117          *                    WIOA--        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5           *
12118          *                                 -------------------------------            *
12119          *                           S-2   |                 M M C C C C D D D|        *
12120          *                                 -------------------------------            *
12121          *                           S-1   |B B B R R R R                   |        *
12122          *                                 -------------------------------            *
12123          *                           S     |                 DATA            |        *
12124          *                                 -------------------------------            *
12125          *                                                                              *
12126          *                    1.  This instruction sends command and data to            *
12127          *                        channel(s).                                           *
12128          *                    2.  CCL if DRT3.(2:1) = 1, an ABORT bit set.              *
12129          *                    3.  CCE if instruction is completed OK.                   *
12130          *                    4.  If global command, skip the DRT3.(2:1) test.          *
12131          *                                                                              *
12132          *     Entered from #IOBS with UBB = 1, CCL already set, SR >= 1,               *
12133          *     F2 cleared                                                               *
12134          *                                                                              *
12135          ********************************************************************************
12136          *
12137   0C3C   WCMD       RC   JSZ  PUL2      SRL2  SR   UBB  JSZC PULM           ZERO  If SR = 2 then pull one more TOS else if
12139                                                                                      SR = 1 then pull two more TOS
12140   0C3D              RB   ADD            HBF2        UBA  JSL  LTOP RG        NF2   Set F2 if global cmd; Set up physical mod#
12142                                                                                      in XRA9 [#PULM & #PUL2 both return with
12143                                                                                      UBA = RC] ( med jump for RG)
12144   0C3E                   JSB  RWIO      EPOP  007F RC   ANDL      SP1B            Go read DRT3, pop 1; SP1B := cdev#
12146   0C3F   WIOC       SP2B JSB  NEXO      NZRO  RB   SP1B JSB  WIO4 SP3B      F2    NEXT if DRT3.(2:1) = 1; #WIO4 if global
12148                                                                                      SP3B := RB + cdev# (command)
12149   0C40   WIO4 RA         ADD       RG                      JSL  IOMS          ZERO  RG := the data; Send out the IMB command
12151   0C41                   JSB  NEXO      UNC                 ADD           CCA     NEXT; Set CCE
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12154           ********************************************************************
12155           *                                                                  *
12156           *                    RIOA--     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5   *
12157           *                               ---------------------------------  *
12158           *                         S-1   |               M M C C C C D D D|  *
12159           *                               ---------------------------------  *
12160           *                         S     |B B B B R R R R                 |  *
12161           *                               ---------------------------------  *
12162           *                                                                  *
12163           *                    1.  This instruction sends command from TOS to*
12164           *                        channel(s)                                *
12165           *                    2.  CCL if DRT3.(2:1) = 1, an ABORT bit set   *
12166           *                    3.  CCE if instruction is completed OK        *
12167           *                    4.  If global command, skip the DRT3.(2:1) test*
12168           *                                                                  *
12169           *   Entered from #IOBS with SR >= 1, CCL already set               *
12170           *                                                                  *
12171           ********************************************************************
12172           *
12173    0C42   RCMD       RB   ADD                              JSZ  PULM SP1B       SRL2  UBA:=RB (PULM WILL ALSO RTN W/UBA=RB)
12175                                                                                       ; IF SR<2 THEN PULL1, SP1B:=0 FOR RIOA
12176    0C43        007F UBA  ANDL       SP4A                 UBA  JSL  LTOP RG         SRNZ  SP4A := cdev#; XLATE logical IMB# to phys
12178    0C44   RWIO RG   RG   INC  LSL        F2    FFFB          ADDL       XR8        F2    DRT2 ADDR; XR8:=RETRY CT ;SKIP IF GLOBAL
12180    0C45        UBA  XR1  INC            ROB3  RG            JSZC NCHL             ZERO  READ DRT3; IF MCDEV#=0 THEN TRAP
12182    0C46             JSB  RWIX       F2    1000          ADDL       BKX5       F2    Skip test if F2; BKX5 := !1000 (for WIOA)
12184    0C47        2000 ADDL                                 OPB  REPN                 0020  Literal for #RWIXB; Waste 32 clocks...
12186    0C48        UBB  INC                 NZRO            OPB  ADD            DCTR CTRO  Skip if unlocked; ...if semaphore locked
12188    0C49        2000 ADDL                                 OPB  JSBI RWIO            ZERO  MASK; Loop if semaphore locked
12190    0C4A   RWIX      SP1B JSB  WIOC      NZRO  UBA  OPB  AND       SP2B             WIOC  IF WRITE CMD;
12192    0C4B        UBB       JSB  NEXO      NZRO  RA   SP4A JSB  RIO4 SP3B       F2    NEXO IF DRT3(2:1):=1; RIO4 IF GLOBAL CMD
12194                                                                                      SP3B:=RD CMD (RA + CDEV )
12195    0C4C   RIO4              ADD            CCA                 JSL  IOMS BKX5       ZERO  SET CCE;SEND THE IMB CMD
12197    0C4D                     ADD                            XR0  ADD               EVEN  ;SKIP IF NOT DNV
12199    0C4E             UBB JSB  RIO4      ODD             XR8  JSBI *+1  XR8        ZERO  READ DATA AGAIN IF NOT YET TIMEOUT
12201    0C4F                 JSB  NEXO      UNC             RG   ADD       RB   SF5B       JSB FOR NEXT INSTRUCTION; RB :=DATA
12203                                                                                        SF5B INHIBITS SECOND EPOP
```

```
 RECORD    C.S.          ********** ALU A **********  ********** ALU B **********
   NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12205              *********************************************************************
12206              *                                                                   *
12207              *                    INIT--          0         7 8 9      12     15  *
12208              *                                    ---------------------------------- *
12209              *                          S         |          M M | CHAN# |        | *
12210              *                                    ---------------------------------- *
12211              *                                                                   *
12212              *                    1.  Send INIT to specified IMB, channel.       *
12213              *                    2.  Clear channel's bit in interrupt mask words, *
12214              *                        (!01A-01D).                                *
12215              *                    3.  Clear the DRT3 entry for devices 0-7 on this *
12216              *                        channel.                                   *
12217              *                    4.  Initialize channel:                        *
12218              *                        Assert PPOLL response, go online to HPIB,  *
12219              *                        assert IFC and REN.  Clear all interrupt   *
12220              *                        mask bits.  Set CCE if it is CIC of HPIB,  *
12221              *                        else set CCG and take it offline again.    *
12222              *                                                                   *
12223              *    Entered from #SHIO with SP4A=DRT3 adr, RG=MCDEV#, SP1B=CHNL#   *
12224              *                                                                   *
12225              *********************************************************************
12226              *
12227              *         Reset channel's bit in mask, send IOCL
12228              *
12229  0C50  INIT RG   RG   ADD  LRZ            001A      ADDL                        UBA.(14:2):=IMB#;UBB:=START OF MASK LOCS.
12231  0C51       UBA  UBB  ADD            ROX4 002F SP1B SUBL      CTR               UBA:=MASK LOCATION,READ;CTR:=CHNL'S BITM#
12233  0C52                 ADD       RH                ADD                         RH:=0;
12235  0C53       3000      ADDL                       REGN CAD                      UBA:=INIT CMD. ;SRGB.(CHNL:1):=1, COMPLEMENT
12237  0C54       UBB  OPA  AND            DATA      UBA  JSL  IOMS SP3B      NZRO  RESET CHNL'S BIT IN MASK,WRITE; SEND IOCL
12239              *
12240              *   Clear DRT3 entries for devices 0-7 on this channel
12241              *
12242  0C55       0004      ADDL       RH        FFF8      ADDL      SP3B            RH:=ADDRESS INCREMENT; SP3B:=-8 (LOOPCOUNT)
12244  0C56  INIA RH   SP4A ADD        SP4A WRX3           ADD                       UBA:=DRT3 ADR FOR THIS CHANNEL, DEV
12246  0C57                 ADD             DATA  SP3B      JSBI INIA SP3B      NZRO  DRT3:=0 ;INIA IF NOT FINISH
12248              *
12249              *   Initialize channel
12250              *
12251  0C58       0008      ADDL       SPOA      0E00      ADDL      SP3B            MASK; REG# E
12253  0C59       1000      ADDL       SP4A                JSL  IOMS               UNC   MASK; READ REG# E
12255  0C5A       RG   SP4A AND            ZERO  FDFF      ADDL      SP2B            SKIP IF GIC-TYPE; TIMER FOR ASSERTING IFC
12257  0C5B                 JSB  NEXO       CCA   0100      ADDL      SP3B            DONE IF NOT GIC-TYPE CHANNEL; REG# 1
12259  0C5C       0100      ADDL       SP4A                JSL  IOMS               UNC   MASK;READ REG#1 FOR SYS CONTROLLER ??
12261  0C5D       SPOA RG   AND            NZRO  1600      ADDL      RH              SKIP IF SYS CONTROLLER;WRITE TO REG#6
12263  0C5E                 JSB  NEXO       UNC   UBB  SP4A ADD       SP1B CCA        SET CCG IF NOT SYS CONTROLLER & EXIT;REG#7
12265  0C5F       0008      ADDL       RG              RH   JSL  IOMS SP3B      NZRO  REG#6:=0008,PPOLL RESPONSE
12267  0C60                 ADD                        8080      ADDL                  ;ENABLE CRC AND GO ON-LINE
12269  0C61            UBB  ADD        RG                SP1B JSL  IOMS SP3B      NZRO  REG#7:=4080,GO ONLINE ON HP-IB
12271  0C62       0030      ADDL       RG                RH   JSL  IOMS SP3B      NZRO  REG#6:=0030,ASSERT IFC,REN ON HP-IB
12273  0C63  INIR 1200      ADDL                         SP2B JSBI INIR SP2B      NZRO  REG#2;ALLOW 100USEC TO ASSERT IFE & REN
12275  0C64       0020      ADDL       RG                UBA  JSL  IOMS SP2B      UNC   REG#6:=0020,REMOVE IFC,KEEP REN
12277  0C65                 CAD        RG                SP2B JSL  IOMS SP3B      NZRO  REG#2:=%FFFF,CLEAR ANY HP-IB INTERFACE ERROR
12279  0C66       0010      ADDL       RH                SP4A JSL  IOMS SP3B      NZRO  MASK;READ REG#1,FOR CIC OF HP-IB
12281  0C67       1700      ADDL                   RH     RG   AND       SP2B      NZRO  WRITE TO REG#7;SKIP IF HP-IB CONTROLLER
12283  0C68                 ADD        RG                UBA  JSL  IOMS SP3B      NZRO  REG#7:=0,TAKE OFFLINE AGAIN
12285  0C69                 JSB  NEXO       UNC   RH    SP2B SUB            CCA       EXIT & SET CCA
```

```
RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12288           *
12289           ************************************************************************
12290           *                                                                      *
12291           *  Enhanced (performance wise) LST/SST Commentary                      *
12292           *                                                                      *
12293           *  This set of instructions (2 to be exact) implements the enhanced    *
12294           *  version of LST/SST.  The instructions themselves are called         *
12295           *  LSTX/SSTX to distinguish them from their slower brothers (or is     *
12296           *  it sisters, I forget what the sex of microcode is).  On power-up    *
12297           *  or boot (etc.) the system defaults to use the older slower ones,    *
12298           *  but when the Operating System determines that it is okay to use     *
12299           *  the new instructions (ie. all the right stuff has been done),       *
12300           *  it will execute an ENPF instruction (enable performance) and the    *
12301           *  LUT entry will get changed to point to LSTX/SSTX.  See all the      *
12302           *  right places for more information (isn't that obnoxious?).          *
12303           *                                                                      *
12304           *  This enhancement (the switch) brought to you by me Eric B. Decker   *
12305           *  Microcode & Co.                                                     *
12306           *                                                                      *
12307           ************************************************************************
12308           *                                                                      *
12309           *  The new LST/SST instructions and support code have been             *
12310           *  incorporated into microcode sets CX2603A and later.  This code      *
12311           *  has been changed in order to increase the speed at which the        *
12312           *  LST and SST instructions execute.  This is done by caching the      *
12313           *  first %20 table pointers into extended CPU registers XRB102         *
12314           *  through XRB117.  This eliminates the memory references in the       *
12315           *  case where K in the instruction field is in the range %1-%17.       *
12316           *  The instructions were also optimized for these cases by             *
12317           *  eliminating unnecessary memory references that were done in the     *
12318           *  old code in order to economize on WCS space.                        *
12319           *                                                                      *
12320           *  The change in the LST/SST instructions also called for a way in     *
12321           *  which to initialize the extended registers.  Thus, the ISTR         *
12322           *  special hardware instruction was born!  It can be found             *
12323           *  immediately following the LST/SST code.                             *
12324           *                                                                      *
12325           *  8/15/85  MRG                                                        *
12326           *                                                                      *
12327           ************************************************************************
```

```
RECORD  C.S.                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12329           ****************************************************************
12330           *                                                              *
12331           *    LST (Load from System Table) Instruction -- Enhanced version   *
12332           *                                                              *
12333           *  This is the performance enhanced version of the LST instruction. *
12334           *  Please see the LST/SST commentary for a description of the      *
12335           *  enhancements.  The following is pseudocode for the new LST.     *
12336           *                                                              *
12337           *    begin                                                     *
12338           *      trap to mode violation if not privileged                *
12339           *      get table number (K) from opcode                        *
12340           *      if table number <> 0 then                               *
12341           *        read table pointer from XRB(102 + table number)       *
12342           *      else (take LST long path)                               *
12343           *        get table number from TOSA                            *
12344           *        read table pointer from MEMORY(%1000 + table number)  *
12345           *      extract table bank from table pointer (bits 11..15)     *
12346           *      extract table base from table ptr (bits 0..10 left justified)*
12347           *      add index (X) to base to point to requested table entry *
12348           *      read the table entry                                    *
12349           *      push data on TOS and set Condition Code                 *
12350           *    end                                                       *
12351           *                                                              *
12352           ****************************************************************
12353   0C6A  LSTX      DSPL ADD                          JSZ  TRP6 BKX3      NPRV  Get K From Instr / Set Bank=0, TRP If NoPriv
                                                                                     Changed label to LSTX
12355
12356   0C6B        X        ADD       RH          00E6 UBA  ADDL      CTR           Get X For Later / Set Up CTR To Read Ptr Reg
12358   0C6C             DSPL JSB  LSTL      ZERO  0200      ADDL      SP1B          Take Long Path If K=0 / Get %1000 Ready
12360   0C6D                  JSZ  PSHM      SR7   001F REGN ANDL      BKX3          Push One If TOS Full / Get Bank From Ptr Reg
12362   0C6E        0200 RH   ADDL                  FFE0 REGN ANDL                    Get %1000+X Ready / Get Table Addr From Ptr
12364   0C6F                  ADD            EPSH  UBB  UBA  ADD           ROX3      Push RH On TOS When Done / Do Table Read
12366   0C70                  ADD                             ADD                     DUM_DEE_DUM, WAITING FOR READ!
12368   0C71                  ADD                        OPB  ADD       RH   CCA  NEXT   / Put Result On TOS, Set CCA, NEXT
12370           *
12371           ****************************************************************
12372           *                                                              *
12373           *  LST long path                                               *
12374           *                                                              *
12375           *  If the LST instruction gets here, we know that K from the   *
12376           *  opcode is 0.  In this case, we will pull the top of stack   *
12377           *  into TOSA if necessary and then POP TOSA for the table      *
12378           *  number.  We then get the table pointer from MEMORY and      *
12379           *  use that pointer to read the table entry we want (indexed   *
12380           *  by X).  The performance of the LST 0 case was sacrificed    *
12381           *  in order to optimize the LST %1-%17 cases which account for *
12382           *  around 90% of all LST's performed.                          *
12383           *                                                              *
12384           ****************************************************************
12385   0C72  LSTL RA           ADD                       JSZ  PULM           SRZ   Get TOS -> UBA / If TOS Empty Get One
12387   0C73                    ADD                 UBA   SP1B ADD           ROX3     / Read Table Pointer From Memory (UBA=TOSA)
12389   0C74                    ADD                             ADD                   HOW BORING! WAITING FOR THE READ TO FINISH
12391   0C75                    ADD                 001F OPB  ANDL      BKX3          / Get Bank From Table Pointer
12393   0C76        0200 RH     ADDL                FFE0 OPB  ANDL                    Get %1000+X / Get Addr From Table Pointer
12395   0C77                    ADD                 UBB  UBA  ADD           ROX3      / Start Read Of (Table Addr)+%1000+X
12397   0C78                    ADD                             ADD                   OH BOY, WE'RE WAITING FOR A READ!
12399   0C79                    ADD                        OPB  ADD       RA   CCA  NEXT   / Put Result On TOS, Set CCA, NEXT
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12402           *
12403           ********************************************************************
12404           *                                                                  *
12405           *    SST (Store to System Table) instruction -- Enhanced version   *
12406           *                                                                  *
12407           *  This is the performance enhanced version of the SST instruction.*
12408           *  Please see the LST/SST commentary for a description of the      *
12409           *  enhancements.  The following is pseudocode for the new SST.     *
12410           *                                                                  *
12411           *    begin                                                         *
12412           *      trap to mode violation if not priviledged                   *
12413           *      get table number (K) from the opcode                        *
12414           *      if table number <> 0 then                                   *
12415           *        read table pointer from XRB(102 + table number)           *
12416           *      else (take SST long path)                                   *
12417           *        get table number from TOSA                                *
12418           *        read table pointer from MEMORY(%1000 + table number)      *
12419           *      extract table bank from table pointer (bits 11..15)         *
12420           *      extract table base from table ptr (bits 0..10 left justified)*
12421           *      add index (X) to base to point to requested table entry     *
12422           *      write data on TOS to requested table entry                  *
12423           *    end                                                           *
12424           *                                                                  *
12425           ********************************************************************
12426           *
12427  0C7A  SSTX      DSPL ADD        SP4A                    JSZ  TRP6 BKX3       NPRV  Get K / Set Bank=0, Trap If No Priv
12429                                                                                       changed label to SSTX
12430  0C7B       X         ADD        RH          00E6 UBA   ADDL      CTR              Get X Ready / Set Up To Read Table Ptr Reg
12432  0C7C           DSPL JSB  SSTL        ZERO   0200       ADDL      SP1B             Take Long Path If K=0 / Get %1000 Ready
12434  0C7D                ADD                    001F REGN  ANDL      BKX3              / Set Up Bank From Ptr Register
12436  0C7E       0200 RH   ADDL                   FFE0 REGN  ANDL                       Get %1000+X Ready / Get Addr From Ptr Reg
12438  0C7F                 ADD                    UBB  UBA   ADD             WRX3         / Start Write Of (Table Addr)+%1000+X
12440  0C80                 ADD                         RA    ADD             POP          / POP Data To Write Off TOS
12442  0C81                 ADD             NEXT        SREG  ADD             DATA         End Of Instr / Send Off The Write!
12444           *
12445           ********************************************************************
12446           *                                                                  *
12447           *  SST long path                                                   *
12448           *                                                                  *
12449           *  If the SST instruction gets here, we know that K from the opcode*
12450           *  is 0.  In this case, we will pull the table number from TOS.    *
12451           *  We will then go to MEMORY to get the table pointer.  (All pretty*
12452           *  similar to what we did with the LST 0 case!)                    *
12453           *                                                                  *
12454           ********************************************************************
12455           *
12456  0C82  SSTL RREG SREG ADD        RH          RA    SP1B ADD             ROX3       Get %1000+X Ready / Start Read Of Table Ptr
12458  0C83                 ADD             EPOP        ADD                              POP Table # Off TOS / WAIT FOR READ
12460  0C84                 JSZ  PULM       SRL2   001F OPB  ANDL      BKX3             We Need 2 On TOS /  Get Bank From Table Ptr
12462  0C85                 ADD                    FFE0 OPB  ANDL                        / Get Table Addr From Table Ptr
12464  0C86                 ADD                    UBB  RH    ADD             WRX3         / Start Write To (Table Addr)+%1000+X
12466  0C87                 ADD             EPOP        RB    ADD                          / POP Data To Write Off TOS
12468  0C88                 ADD             NEXT        SREG  ADD             DATA       End Of Instruction / Send Off The Write!
```

```
RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12471          ************************************************************************
12472          *                                                                      *
12473          *  ISTR  (Initialize System Table Registers) Instruction               *
12474          *                                                                      *
12475          *  OPCODE - %020104  ,  %000015                                        *
12476          *           Available in version CX2603A and beyond                    *
12477          *                                                                      *
12478          *  This instruction copies the contents of absolute memory             *
12479          *  locations %1000 through %1017 to extended CPU registers             *
12480          *  XRB102 through XRB117.  These CPU registers are then used as the    *
12481          *  table pointers for table numbers %0 through %17 by the LST and      *
12482          *  SST instructions.                                                   *
12483          *                                                                      *
12484          *  NOTE:  The last line of this instruction is a trap to segment 1.    *
12485          *         This line was necessary when microcode handled the           *
12486          *         re-initialization of the table pointer registers.  Now,      *
12487          *         however, on recovery from powerfail, the old LST/SST are     *
12488          *         used until an ENPF is executed, which automatically calls    *
12489          *         ISTR to insure that the table pointer registers are valid.   *
12490          *         Therefore, the last line of ISTR never gets executed and     *
12491          *         could be removed; however, I have grown rather fond of it    *
12492          *         and would be saddened to see it ruthlessly deleted           *
12493          *         without consideration for its feelings on the matter.        *
12494          *                                                                      *
12495          ************************************************************************
12496          *
12497   0C89   ISTR 000F      ADDL      RH                    ADD       BKX3           Get Index Var Ready / Set Bank = 0
12499   0C8A        0200      ADDL      SP4A                  ADD                      Get %1000 For Mem Reads
12501   0C8B                  ADD                             ADD                       Wait For SP4A To Become Valid
12503   0C8C   ISR1 FFFF RH   ADDL      RH          RH   SP4A ADD            ROX3      Dec Index Var / Read Table Pointer @ Index
12505   0C8D              SREG ADD                   00E6 RH   ADDL      CTR            Carry Index Along / Point CTR To Register
12507   0C8E              SREG JSB  ISR1      NZRO         OPB  ADD       REGN           Loop Until %0-%17 Done / Store Ptr In Reg
12509   0C8F                   ADD                             JSL  NEX1           F1     / End Of Instruction If Not Powerfail
12511   0C90                   ADD                             JSL  ENPI           F2     AND RETURN TO ENPF IF NEEDED.
12513   0C91                   ADD                             JSL  IR1            UNC    / Trap To Segment 1 If Recover From PFail
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12516           ************************************************************************
12517           *
12518           * LST/SST & Performance enhancement retrofit.
12519           * 10/25/85 ebd (Eric B. Decker) Microcode & Co.
12520           *          available in version CX2603A and beyond
12521           * This retrofit includes code to make the current level of microcode
12522           * fully compatible with all versions of the operating system (MPE)
12523           * back to but not including V/P (aka MPE IV with disc caching).
12524           *
12525           * When the system first powers up, all performance enhancements are
12526           * disabled.  The reason for this is primarily effected by LST/SST.
12527           * There are multiple registers (in the register file) that the
12528           * enhanced LST/SST uses that need to be set up prior to the first
12529           * invokation of LST/SST.  If this doesn't occur the system will die
12530           * horribly.  The new instruction ISTR is used by the operating system
12531           * to set these registers up when they are stable and prior to first
12532           * use of the instruction LST or SST.
12533           *
12534           * Other performance enhancements (namely the OS into microcode stuff)
12535           * also make use of the values in these registers to get a tad more
12536           * performance out of this tired old beast.  So it also imperative
12537           * that the LST/SST registers be set up prior to execution of any
12538           * of these instructions.  At this date of writing these additional
12539           * instructions are ERON, ERXT, XGDB, MSTA,TIMR, and TMRQ.
12540           *
12541           * Not all of the above instructions require LST/SST to be set up.
12542           * But to supply an encapsulation wall, we are disabling all of these
12543           * instructions until the ENPF, enable performance enhancements,
12544           * instruction is executed.  This instruction should not be executed
12545           * until all initilization needed is complete.  This includes the
12546           * execution of ISTR.
12547           *
12548           * ENPF currently does the following:
12549           *
12550           *         1) Changes the LUT entry for LST from the non-enhanced
12551           *            version to the enhanced version.
12552           *         2) changes the LUT entry for SST (same as LST)
12553           *         3) modifies the limit for doubleword instructions to enable
12554           *            performace (OS into microcode) instructions at %30 & up
12555           *         4) sets %1220 bit 13 to on.  This tells other routines that
12556           *            we are running Performance microcode.  Zoooooom
12557           *         5) Puts four words into the SysGlob Extension area, %21-%24
12558           *            The first three will be zero (reserved). The fourth word
12559           *            (%24) will have a bit set for every microcode routine
12560           *            (performance) that is implemented.  Currently we have:
12561           *
12562           *               LOCATION: SYSGLOB EXTENSION %24
12563           *                         ERON              bit 15
12564           *                         ERXT                  14
12565           *                         XGDB                  13
12566           *                         TIMR                  12
12567           *                         TMRQ                  11
12568           *                         MSTA                  10
12569           *                         Reserved             0-9
12570           *
12571           ************************************************************************
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12573    0C92  ENPF           ADD             SF2                JSL  ISTR           UNC   GO SET UP SYSGLOB CACHE FIRST
12575                                                                                      FLAG2 IS SET SO WE RETURN HERE
12576    0C93  ENP1           JSZ  PSM2       SRG5               ADD                      MAKE ROOM FOR TWO TOS ENTRIES IF FULL
12578                                                                                      THIS IS WHERE ISTR RETURNS TO
12579    0C94                 ADD                     003B       ADDL      CTR            POINT CTR AT LUT PARITY OFF
12581    0C95                 ADD       REGN               BNKP ADD       BKX3           DISABLE LUT PARITY
12583                                                                                      BKX3 <- BNKP NO WRP BUT WRX3
12584    0C96           P     ADD             WRX3  3000       ADDL                     GET BNKP AND P  (2ND WRD OF CUR INST)
12586                                                                                      PUT LST INSTR ON UBB FOR NEXT LINE
12587    0C97           UBB   ADD             DATA  0038       ADDL      CTR            STUFF OUT THE LST INST AT 2ND WORD
12589                                                                                      POINT REGN AT LUTO
12590    0C98     LSTX        ADDL LBL  SP4A             P    ADD             RONP       GET LSTX LABEL TO SP4A FOR GTPR CALL
12592                                                                                      GET LST INTO NIR TO ADDRESS LUT (@P)
12593    0C99                 JSB  GTPR       UNC                ADD                      CALL GTPR TO CALC PARITY
12595    0C9A     SPOA        ADD       RH                OPB  ADD                      STASH THE RESULTANT PARITY
12597                                                                                      AND MAKE SURE NIR IS VALID
12598    0C9B           LUTO  ADD       SP4A                     JSB  GTPR           UNC   GET THE FIRST HALF OF THE LUT FOR PAR
12600                                                                                      GO CALC PARITY ON IT.
12601    0C9C     SPOA  RH    XOR       RH                       ADD                      CALC COMPOSITE PARITY IF 1 CHANGE
12603    0C9D           LUT1  ADD       SP4A       LSTX          ADDL LBL                 GET THE PART WITH THE PARITY BIT
12605                                                                                      GET THE ADDRESS OF THE NEW PART
12606    0C9E           UBB   ADD       REGN WRX3                ADD             ICTR       STASH THE NEW LABEL IN THE LUT
12608                                                                                      BUMP CTR SO IT POINTS TO LUT1
12609    0C9F     RH          ADD             ZERO  0010       ADDL                     CHECK FOR ZERO IF SO NO PAR MOD
12611                                                                                      GET A WORD WITH BIT 27 SET (PARITY)
12612    0CA0     UBB   SP4A  XOR       REGN WRX3                ADD                      FLIP THE PARITY BIT AND STASH IN LUT
12614             *
12615             ************************************************************************
12616             *                                                                      *
12617             * At this point we have successfully set up the LUT to point at the   *
12618             * enhanced version of LST (LSTX).  Now let's do it for SST.            *
12619             *                                                                      *
12620             ************************************************************************
12621             *
12622    0CA1           P     ADD             WRX3  30D0       ADDL                     POINT AT THE MEM CELL BEING USED
12624                                                                                      PUT A SST 0 INSTR ON UBB FOR NXT LNE
12625    0CA2           UBB   ADD             DATA  0038       ADDL      CTR            WRITE THE SST 0 INSTR OUT
12627                                                                                      RESET CTR BACK TO LUTO
12628    0CA3     SSTX        ADDL LBL  SP4A             P    ADD             RONP       PUT NEW LUT VALUE IN PARM FOR GTPR
12630                                                                                      MAKE NIR PNT AT THE SST LUT ENTRY
12631                                                                                      GET FETCH NIR FOR LUT POINTER
12632    0CA4                 JSB  GTPR       UNC                ADD                      GO CALC PARITY ON NEW LUT ENTRY
12634    0CA5     SPOA        ADD       RH                OPB  ADD                      STASH RESULT PARITY IN SAFE PLACE
12636                                                                                      ALSO MAKE SURE NIR IS VALID
12637    0CA6           LUTO  ADD       SP4A                     JSB  GTPR           UNC   GET THE OLD LUT ENTRY
12639                                                                                      GO CALC PARITY ON IT
12640    0CA7     SPOA  RH    XOR       RH                       ADD                      IF 0 NO PAR CHANGE, 1 YEP CHANGE PAR
12642    0CA8           LUT1  ADD       SP4A       SSTX          ADDL LBL                 GET THE LUT ENTRY WITH PARITY SAFE
12644                                                                                      GET WHAT TO MAKE THE FIRST LUT ENTRY
12645    0CA9           UBB   ADD       REGN WRX3                ADD             ICTR       STASH THE NEW VALUE INTO THE LUT
12647                                                                                      AND BUMP CTR TO LUT1 FOR NXT IF PAR
12648    0CAA     RH          ADD             ZERO  0010       ADDL                     CHECK FOR ZERO IF SO NO PARITY MOD
12650                                                                                      GET THE BIT TO CHANGE FOR PARITY SW.
12651    0CAB     UBB   SP4A  XOR       REGN WRX3  0027       ADDL      CTR            SWITCH PARITY AND PUT BACK INTO LUT
12653                                                                                      POINT CTR AT WCS PARITY DISABLE
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12655           ********************************************************************************
12656           *                                                                              *
12657           * At this point both the LUT entries for LST and SST have been                 *
12658           * modified.  Now let's go enable the double word performance inst.             *
12659           * by changing the jump table index check at CLKX in CLKI.                      *
12660           *                                                                              *
12661           ********************************************************************************
12662           *
12663  0CAC           P    ADD            WRX3  000E      ADDL                               POINT AT WHERE TO RESTORE THE INST
12665                                                                                         THE DATA TO PUT BACK  (ENPF INST.)
12666  0CAD      UBB       ADD            DATA  CLKX      ADDL LBL                           REPLACE THE ORIGINAL INST. (@P)
12668                                                                                         PUT THE ADDR OF WHERE TO CHANGE OUT
12669  0CAE      UBB       ADD            WRX3            ADD                                STICK YREGA WITH THE ADDRESS
12671  0CAF                ADD       REGN RAR   003B      ADDL      CTR                      KICK WCS PARITY OFF ON *+3
12673                                                                                         POINT CTR (REGN) AT WCS3 FOR READS
12674  0CB0                ADD                            ADD                                WAIT FOR A BIT.  NEW CTR VALUE RESTS
12676  0CB1                JSB  *+2       MEDJ      REGN  ADD                      RSB       BEATS ME.  A STRANGE INCANTATION TO
12678                                                                                         GET THE WCS READ SET UP
12679  0CB2                ADD                            ADD                                THIS IS WHERE THE READ STUFF GOES.
12681  0CB3                ADD                       SREG ADD       RH                       HOLD THE DATA AND GRAB IT INTO A REG
12683                                                                                         WE CAN USE.
12684  0CB4                ADD                       UBB  ADD  LSR                           GET RID OF PARITY.  WE WANT THE
12686                                                                                         CONSTANT FIELD BY ITSELF.
12687  0CB5      1FFF UBB  ANDL       SP4A                 JSB  GTPR           UNC          STRIP TOP TWO BITS OF WCS3 AND STASH
12689                                                                                         FOR GTPR.  THEN CALL THE BEAST
12690  0CB6      SP0A      ADD        RH          C001 RH   ANDL      SP1B                   STASH THE PARITY RESULT IN SAFE PLACE
12692                                                                                         ALSO GET THE BITS THAT STAY
12693  0CB7      001D      ADDL       SP4A                 JSB  GTPR           UNC          CALC PARITY ON NEW VALUE
12695                                                                                         AND OF COURSE MUST CALL THE ROUTINE
12696  0CB8      SP0A RH   XOR        RH          001D     ADDL                              PRODUCE WHETHER PARITY NEEDS CHANGING
12698                                                                                         GET THE NEW VALUE TO STICK OUT
12699  0CB9      UBB       ADD  LSL             0023     ADDL      CTR                      POSITION DATUM PROPERLY
12701                                                                                         POINT AT WCS3 FOR WRITES
12702  0CBA      CLKX      ADDL LBL   SP4A      UBA  SP1B IOR       SP1B                   GET ADDRESS OF WCS TO CHANGE
12704                                                                                         COMBINE THE DATA TO GET REAL STUFF
12705  0CBB      0001 UBB  XORL                 RH        ADD                      ZERO      BUILD FLIPPED PARITY VERSION IN CASE
12707                                                                                         CHECK TO SEE IF WE SHOULD FLIP PARITY
12708  0CBC           SP4A ADD            WRX3  UBA       ADD       SP1B                   STASH THE ADDRESS IN YREGA
12710                                                                                         IF NOT KILLED BY PREV LINE STASH
12711  0CBD                ADD            RAR             ADD       RH   EPSH             NOW GET IT INTO RAR
12713                                                                                         STACK WILL HAVE ONE MORE ITEM
12714                                                                                         ZERO WHAT WILL BE TOS-1
12715                                                                                         WAIT FOR THINGS TO SETTLE DOWN
12716  0CBE           SP1B ADD       REGN WRX3            ADD                                AND STASH OUT IN WCS
12718  0CBF                ADD                            ADD                                WAIT FOR THINGS TO SETTLE DOWN
12720  0CC0                JSB  *+1       MEDJ            ADD                      RSB       JSB OVERRIDES RSB.
12722                                                                                         RSB FORCES RAR ON VBUS DURING RANK 1
12723                                                                                         THIS SURE IS RANK ISN'T IT.
```

```
 RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12725            *
12726            * WCS SHOULD NOW BE WRITTEN.  LETS MODIFY %1220 TO TELL THE S/W
12727            * THAT WE ARE RUNNING PERFORMANCE MICROCODE.  WHEN THAT IS DONE SET
12728            * UP THE SYSGLOB EXTENSION AREA TO TELL WHICH ROUTINES ARE PRESENT
12729            *
12730   0CC1          0290      ADDL                            ADD       BKX3 EPSH       PREPARE TO ADDRESS BANK 0, ABS %1220
12732                                                                                      ZERO BKX3 TO GET BANK 0
12733                                                                                      AT NEXT HAVE TWO MORE WORDS ON STACK
12734   0CC2          UBA       ADD             ROX3       P     INC       P    RONP      GO GET THE MONSTER (%1220)
12736                                                                                      BUMP P TO POINT CORRECT AND GET NIR
12737   0CC3          0004      ADDL                  003A       ADDL      CTR             GET A WORD WITH JUST BIT 13 SET
12739                                                                                      AND WAIT FOR THE READ TO HAPPEN
12740                                                                                      POINT AT 3A TO TURN LUT PARITY ON
12741   0CC4          UBA  OPA  IOR             DATA  02FF       ADDL                    AND SEND THE WORD WITH BIT 13 BACK
12743                                                                                      UBA <- PTR TO SYSGLOB EXT AREA
12744   0CC5                    ADD       REGN             UBB  ADD             ROX3      TURN LUT PARITY BACK ON!
12746                                                                                      READ SYSGLOB EXT BASE INTO OPB
12747   0CC6                    ADD                  0211       ADDL                    UBB <- OFFSET TO XP'FEATURE'MASK WD1
12749   0CC7                    ADD                  UBB  OPB   ADD             WRX3      SET UP TO WRITE INTO XP'FEATURE'MASK
12751   0CC8                    ADD                             ADD             DATA      CLEAR WORD 1
12753   0CC9                    ADD                             ADD             DATA      CLEAR WORD 2
12755            *
12756            * The A side of the next line determines what we tell the software
12757            * we have implemented.  Currently set to 3F says we have set up
12758            * ERON,ERXT,XGDB,TIMR,TMRQ, and MSTA.  If more routines are added,
12759            * this number will have to be changed accordingly.
12760            *
12761   0CCA          003F      ADDL                            ADD             DATA      UBA <- MASK THAT ENABLES ERON,ERXT,XGDB
12763                                                                                        TIMR,TMRQ and MSTA
12764                                                                                      CLEAR WORD 3
12765   0CCB                    ADD                       UBA  ADD             DATA      WORD 4 <- MASK
12767   0CCC                    ADD                            ADD                  NEXT   GO TO THE NEXT INSTRUCTION
```

```
 RECORD   C S                  ********** ALU A **********  ************** ALU B **************
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12770           ************************************************************************
12771           *                                                                      *
12772           *    LST (Load from system table) Instruction                          *
12773           *                                                                      *
12774           *    Slow version - non-enhanced                                       *
12775           *    This is the original version of LST/SST.  It is the non-perf.     *
12776           *    enhanced version.  It is here for backward compatibility.  See    *
12777           *    ENPF (LSTRETRO LST retrofit), LSTX, and SSTX for details.         *
12778           *                                                                      *
12779           *    A system that does not do the right things will fail if the new   *
12780           *    LST/SST instructions are executed (note other performance inst.   *
12781           *    also make use of the LST/SST registers).  To avoid this the old   *
12782           *    version of the instruction is executed until the Operating Sys    *
12783           *    tells the microcode that it is okay.  This makes all this stuff   *
12784           *    backward compatibile.  The older versions of the Operating Sys    *
12785           *    don't know about the new instruction ENPF (which tells us it      *
12786           *    is okay) so it can't tell us it is okay.                          *
12787           *                                                                      *
12788           ************************************************************************
12789           *
12790           $LUT INSTR=LST:0 011 000 000 00,DSPL=4,SR=0,ENTRY=LST
12791           *
12792    OCCD   LST       DSPL ADD       SP4A NZRO            JSZ  TRP6 BKX3       NPRV  Skip if K <> 0; BKX3 := 0, trap if NPRV
12794    OCCE            RA   JSZ  PULM      SRZ   0200      ADDL      SP3B             UBA := (S), pull TOS if K = 0 and SR = 0;
12796                                                                                     SP3B := UBB := %1000
12797    OCCF       X    UBB  ADD       RH        UBB  UBA   ADD            ROX3       RH := X + %1000; Read ((S) + %1000)
12799    OCD0            DSPL ADD            ZERO  RREG SP4A  ADD            ROX3                  ;READ (K+%1000)
12801    OCD1                 JSB  LST5      UNC   001F OPB   ANDL      BKX3             JMP IF K<>0    ;BKX3=TABLE BANK          (CSTX)
12803    OCD2            DSPL ADD            NZRO  FFE0 SREG  ANDL                       Skip if K <> 0; Mask to address on UBB(CSTX)
12805    OCD3                 JSB  LST0      UNC   RH   UBB   ADD            ROA3       JSB if K = 0;                             (CSTX)
12807                                                                                      Read (X + %1000 + ((S) + %1000))
12808    OCD4   LST5          ADD                       001F OPB   ANDL      BKX3                  ;MASK TO BANK IN BKX3     (CSTX)
12810    OCD5                 ADD                       FFE0 SREG  ANDL                  ;MASK TO ADDRESS ON UBB                (CSTX)
12812    OCD6                 JSZ  PSHM      SR7   RH   UBB   ADD            ROA3       JSB if SR = 7;                            (CSTX)
12814                                                                                      Read (X + %1000 + (K + %1000))
12815    OCD7                 ADD            EPSH            JSL  LWP2            UNC   Push; JSB (@ #LWP2;  RH := OPA, CCA)
12817    OCD8   LST0      OPA  ADD       RA    CCA           ADD                  NEXT  (S) := OPA, CCA; NEXT
```

```
 RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12820           ****************************************************************
12821           *                                                              *
12822           *    SST (Store into system table) Instruction                 *
12823           *                                                              *
12824           *    SST non-enhanced (performance wise).  See LST, LSTX, SSTX,*
12825           *    ENPF for more info.                                       *
12826           *                                                              *
12827           ****************************************************************
12828           *
12829           $LUT INSTR=SST:0 011 000 011 01x xxx, DSPL=4, SR=1, ENTRY=SST
12830           *
12831    0CD9   SST  0200       ADDL       SP4A                 JSZ  TRP6 BKX3       NPRV  SP4A := %1000; BKX3 := 0, trap if NPRV
12833    0CDA             DSPL ADD                      RA   UBA  ADD            ROX3       UBA=K          ; Read ((S) + %1000)
12835    0CDB                  ADD                      UBA  SP4A ADD            ROX3                      ; Read (K + %1000)
12837    0CDC             DSPL ADD             ZERO     001F OPB  ANDL      BKX3            SKP IF K=0; MASK TO BANK & PUT IN BKX3(CSTX)
12839    0CDD        X    SP4A ADD             UNC      FFE0 SREG ANDL                      UBA=X+%1000;MASK TO ADDRESS ON UBB    (CSTX)
12841    0CDE                  JSB  SST8       EPOP     UBA  UBB  ADD            WRX3       Pop 1 and JSB if K = 0;               (CSTX)
12843                                                                                        WRITE (X + %1000 + ((S) + %1000))
12844    0CDF                  ADD                      001F OPB  ANDL      BKX3                      ;MASK TO BANK & PUT IN BKX3(CSTX)
12846    0CE0        X    SP4A ADD                      FFE0 SREG ANDL                      UBA=X+%1000  ;MASK TO ADDRESS ON UBB  (CSTX)
12848    0CE1             RA   JSB  SST9       EPOP     UBA  UBB  ADD            WRX3       UBA := (S), pop 1, JSB;               (CSTX)
12850                                                                                          Read X + %1000 + M(K + %1000)
12851    0CE2   SST8      RB   ADD             EPOP                JSZ  PULM            SRL2  UBA := (S-1); Pull 1 TOS if SR < 2
12853    0CE3   SST9           ADD             NEXT          UBA  ADD            DATA       NEXT; Write (S) or (S-1) if K = 0
12855
12856
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12858           ************************************************************************
12859           *                                                                      *
12860           * Get Parity - GTPR                                                    *
12861           *                                                                      *
12862           * This routine takes a 16 bit input and determines if it is even       *
12863           * or odd parity.  If even it returns 0, if odd 1.                      *
12864           *                                                                      *
12865           * Input:  SP4A - word to be worked on.                                 *
12866           * Output: SP0A - parity indication 0 - even, 1 - odd                   *
12867           *                                                                      *
12868           ************************************************************************
12869           *
12870    0CE4   GTPR            ADD       SP0A                ADD             PSHR        ZERO THE PARITY ACCUMULATOR
12872                                                                                     INCREMENT RAC SO WE REMEMBER WHR FRM
12873    0CE5             SP4A JSB  GTPX       ZERO            ADD                         IF WE ARE ZERO QUIT
12875    0CE6   GTP1      SP4A ADD             EVEN  0001      ADDL                        CHECK FOR SP4A(15:1) BEING SET
12877                                                                                     GET THE 0001 TO XOR WITH RESULT
12878    0CE7       SP0A UBB  XOR       SP0A            SP4A ADD  LSR                    IF SP4A(15:1) IS SET SWITCH THE SENSE
12880                                                                                     SHIFT SP4A (INPUT) ONE MORE RIGHT    (2502)
12881    0CE8             UBB  ADD       SP4A            UBB  JSB  GTP1          NZRO    REMEMBER THE SHIFTED VALUE
12883                                                                                     AND TRY AGAIN BUT ONLY IF NEEDED
12884    0CE9   GTPX            ADD            NZRO            ADD             POPR        SLOW DOWN THE RETURN SO POPR CAN TAKE
12886                                                                                     POPR SO WE POINT AT CORRECT RETURN
12887    0CEA                   ADD            RSB             ADD                         AND RETURN TO CALLER
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
12890	OCEB	CYCL	RA		JSZ	PULM		SRZ	003C		ADDL		CTR			GET THE VALUE TO PUT THERE. GET IT
12892																IF NEEDED. PNT CTR AT PERF REGISTER
12893	OCEC		UBA		ADD			WRX3		P	INC		P	RONP		GET NEW PERF VALUE INTO YREGA
12895																BUMP P AND GET NEXT INST. IN NIR
12896	OCED				ADD		REGN	EPOP	003D		ADDL		CTR			OKAY STICK IT OUT INTO THE PERF REG
12898																REMOVE THE DATA FROM THE STACK
12899																POINT CTR AT THE DISPLAY PERF LOC
12900	OCEE				ADD		REGN				ADD				NEXT	MAKE SURE DISPLAYING PERF REG
12902		*														
12903		*	DISABLE PERF DISPLAY													
12904		*														
12905	OCEF	NCYC			ADD				003E		ADDL		CTR			POINT AT DISP CIR
12907	OCF0				ADD		REGN				JSL	NEX1			UNC	KICK SO CIR DISPLAYS
12909	OCF1				ADD						ADD					NOP to keep system happy
12911	OCF2				ADD						ADD					NOP to keep system happy

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12916           %0D00
12917           ************************************************************************
12918           *    Look up routine for machine dependent instructions                *
12919           *                                                                      *
12920           *    Entered from #MBLA with privileged mode already checked.          *
12921           *                                                                      *
12922           * WARNING: If you add instructions at the end of the table be sure     *
12923           * to change ENPF so it enables them.  Otherwise you can not execute    *
12924           * them.  The constant embeded at CLKX should not be changed.           *
12925           *                                                                      *
12926           ************************************************************************
12927           *
12928   0D00  CLKI 0015      ADDL      SP4A       CTAB OPB  ADDL LBL                  SP4A := %21, (addr of LR);
12930                                                                                    UBB := indexed entry into jump table
12931   0D01           UBB  ADD             WRX3  0019      ADDL      CTR             YRGA := table entry;
12933                                                                                    CTR := DCUSTORE (for CNTL 'B' instr)
12934   0D02                ADD       SP0A RAR               ADD             CRF       SP0A := parameter := 0, RAR := YRGA
12936                                                                                  Clear RFLAG for regn usage in eron    (2553)
12937   0D03  CLKX    SP4A ADD             ROX3  0010 OPB  SUBL                CRRY    Read LR; Skip if 2nd word is <= %20   (2553)
12939                                                                                  ADDED LABEL CLKX FOR ENPR MOD         (2553)
12940                                                                                  NOTE: 20 GETS CHANGED BY ENPF         (2553)
12941   0D04       UBA       CAD       RH    RSB               JSZ  TRAP           UNC   RH := %20, vector through jump table;
12943                                                                                   Unimplemented instructions > %57     (2553)
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

12945           ***************************************************************
12946           *                                                               *
12947           *          Jump table for machine dependent instructions        *
12948           *                                                               *
12949           ***************************************************************
12950           *
12951     0D05  CTAB       OPA  JSB  RCCR      UNC   UBA       CAD           ROX3       0.) READ CLOCK instruction, UBA := LR;
12953                                                                                       Read TR
12954     0D06             OPA  JSB  SCLR      UNC   UBA       CAD           ROX3       1.) SET CLOCK instruction, UBA := LR;
12956                                                                                       Read TR
12957     0D07       FF7F XR15 ANDL      XR15                  JSB  NEX1           UNC  2.) TOFF instruction, clear bit 8;
12959                                                                                       Read next instruction
12960     0D08                 JSB  TON       UNC        P     INC       P     RONP       3.) TON instruction;
12962                                                                                       Read next instruction
12963     0D09       0003      ADDL      SP4A                  JSL  MCMD           UNC  4.) MESSAGE COMMAND instruction, SP4A := 3
12965                                                                                       (number of TOS required)
12966     0D0A       1000      ADDL      SP4A                  JSL  FLSH BKX5      UNC  5.) FLUSH instruction, SP4A := !1000,
12968                                                                                       BKX5 := 0
12969     0D0B                 ADD                             JSL  MLOG           UNC  6.) Microcode Logging instruction
12971     0D0C                 ADD                             JSL  CCPP           UNC  7.) Communicate w/Channel Program Processor
12973     0D0D                 ADD                       P     JSBI SINC P         UNC  10.) SINC instruction, P := P + 1
12975     0D0E                 ADD                             JSL  SBF            UNC  11.) Set Bounds Flag (PRIV bounds checking)
12977     0D0F                 ADD                             JSL  SBL            UNC  12.) Set Bounds Limit (PRIV bounds checking)
12979     0D10                 ADD                             JSL  DDE            UNC  13.) Enable/Disable CNTL 'B'
12981     0D11                 ADD                             JSL  RDCU           UNC  14.) Read DCU log instruction
12983     0D12                 ADD             SF1             JSL  ISTR           UNC  15.) Initialize System Table Registers(2553)
12985     0D13                 ADD             CF1             JSL  ENPF           UNC  16.) Enable Performance ...          (2553)
12987     0D14                 ADD                             JSL  CYCL           UNC  17.) Cylon - Cycle Perf Reg.   (RTOC on 37)
12989     0D15                 ADD                             JSL  NCYC           UNC  20.) Turn off Cylon.             (WTOC on 37)
12991     0D16                 ADD                             JSZ  TRP            UNC  21.) Unimplemented instruction (PFL on 37)
12993     0D17                 ADD                             JSZ  TRP            UNC  22.) Unimplemented instr. (FIRMVER on 37)
12995     0D18                 ADD                             JSZ  TRP            UNC  23.) Unimplemented instruction (OSSIG on 37)
12997     0D19                 ADD                             JSZ  TRP            UNC  24.) Reserved for non-68 expansion...
12999     0D1A                 ADD                             JSZ  TRP            UNC  25.) Reserved for non-68 expansion...
13001     0D1B                 ADD                             JSZ  TRP            UNC  26.) Reserved for non-68 expansion...
13003     0D1C                 ADD                             JSZ  TRP            UNC  27.) Reserved for non-68 expansion...
13005     0D1D                 ADD                             JSL  ERON           UNC  30.) Erroron routine for MPE          (2553)
13007     0D1E                 ADD                             JSL  ERXT           UNC  31.) Errorexit routine for MPE        (2553)
13009     0D1F       0005      ADDL      SPOA                  JSL  XGDB BKX3      UNC  32.) ExchangeDB routine for MPE       (2553)
13011     0D20                 ADD                             JSL  TIMR           UNC  33.) TIMER routine for MPE            (2553)
13013     0D21       0004      ADDL      RH                    JSL  TMRQ           UNC  34.) TIMEREQ routine for MPE          (2553)
13015     0D22                 INC             CCA             JSL  MSTA BKX3      UNC  35.) MMStat routine from MPE          (2553)
13017     0D23                 ADD                             ADD                      Nop to keep the cpu happy
13019     0D24                 ADD                             ADD                      Nop to keep the cpu happy
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13024           %0D40
13025           ************************************************************************
13026           *    Clock Instructions                                                *
13027           ************************************************************************
13028           *
13029   0D40    TON  0080 XR15 IORL      XR15                ADD                       SET TON;
13031   0D41              STA  ADD  LSL       POS            ADD                       SKIP IF INT IS OFF
13033   0D42              XR15 JSZ  CSNC      NEG            ADD                       SERVICE INTERRUPTS
13035   0D43                   ADD            NEXT           ADD                       NEXT
13037   0D41    RCCR           ADD                      UBA  XR31 JSZ  PSHM SP3B      SR7   ; SP3B:=UBB:=LR + CR, EMPTY RG IF SR = 7
13039   0D45              JSB  NEX1      UNC        UBB  OPB  ADD       RH   EPSH            JSB: NEW (S):=(LR + CR) + TR, EPSH
13041   0D46    SCLR           ADD                      UBA  XR31 JSZ  PULM SP3B      SRZ   ; SP3B:=UBB:=LR + CR, FILL RA IF SR = 0
13043   0D47              SP4A ADD            WRX3       UBB  OPB  ADD             DATA      ADDR OF LR; TR:=(LR + CR) + TR
13045   0D48              RA   JSB  NEX1      DATA           RA   SUB       XR31 EPOP        JSB, LR:=(S); CR:=-(S), EPOP
13047   0D49    NEX1           JSZ  NOP       UNC            P    INC       P    RONP        WAIT TWO CLOCKS FOR NEXT;
13049                                                                                        P:=P + 1, READ NEXT INSTRUCTION
13050   0D4A    SINC      UBB  ADD            RONP       SP4A INC             ROA3        READ NEXT INSTRUCTION; READ STAT
13052   0D4B         0007      ADDL      SP4A            8000 ADDL      RH                SP4A:=MASK TO SET DINTFF;
13054                                                                                        RH:=UBB:=BIT 0 MASK
13055   0D4C              STA  ADD  LSL       POS            UBB  OPB  IOR             DATA      SKIP IF INTERRUPTS ARE OFF;
13057                                                                                          (STAT) := !8000 IOR (STAT)
13058   0D4D                   JSZ  INT9      SF2            8C01      ADDL      SP1B            SERVICE INTERRUPT IF INTERUPTS ON, SF2 FOR
13060                                                                                          #INT9;
13061                                                                                        SP1B:=EXTERNAL LABEL
13062   0D4E         RH   XR15 IOR       XR15 NEXT           SP4A ADD             CCPX        SET SINCBIT, NEXT; SET DINTFF
13064   0D4F                   ADD                                 ADD                       NOP to keep system happy
13066   0D50                   ADD                                 ADD                       NOP to keep system happy
```

```
 RECORD    C S.      ********** ALU A **********  ********** ALU B **********
   NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13071           %0D80
13072           !*********************************************************************
13073           !                                                                     *
13074           !       ERON - REPLACEMENT FOR THE MPE PROCEDURE ERRORON              *
13075           !                                                                     *
13076           !            Macro-assembly code: %020104,%000030  (doubleword op)    *
13077           !                available in version CX2603A and beyond              *
13078           !                                                                     *
13079           !     This microcode routine replaces the MPE procedure ERRORON,      *
13080           !     which turns off traps, and if the caller is not a system        *
13081           !     routine increments the procedure's error level count, clearing   *
13082           !     that error level and possibly the one above it.  If the count    *
13083           !     exceeds six the counter is still incremented every call, but     *
13084           !     that's all - the (non-existent) levels above 6 are not cleared.  *
13085           !                                                                     *
13086           !       This routine hard-codes the offsets of all the locations it   *
13087           !     uses as they are defined in U-MIT, specifically:                *
13088           !                                                                     *
13089           !          Current PCB number in absolute(4)                          *
13090           !          Sysglob entry for start of PCBs in LST 3                   *
13091           !          Pointer to the PXFIXED area in (DL-2)                      *
13092           !          Errorlevel, followed by 6 error words, at PXFIXED+35       *
13093           !          System code bit for process at offset PCB+3(1:1)           *
13094           !          Process type bit for process at offset PCB+9(6:1)          *
13095           !                                                                     *
13096           !   +---------------**********  WARNING  **********---------------+   *
13097           !   |     IF ANY OF THE OFFSETS OR BIT POSITIONS USED BY THIS CODE |   *
13098           !   |     SHOULD CHANGE IN FUTURE VERSIONS OF MPE, THIS CODE MUST  |   *
13099           !   |     BE MODIFIED OR IT WILL CEASE TO WORK PROPERLY.           |   *
13100           !   +--------------------------------------------------------------+   *
13101           !                                                                     *
13102           !       This routine also makes use of the enhanced LST/SST code,     *
13103           !     assuming the value for LST 3 to be in XRB105 (!E9).             *
13104           !                                                                     *
13105           !       The original source in SPL which this is taken from is        *
13106           !     defined in the module CHECKER, and is as follows:               *
13107           !                                                                     *
13108           !               <<RUN TIME (ABORT) ERRORS.                            *
13109           !                    - INCREMENT ERROR LEVEL                          *
13110           !                    - CLEAR ERROR WORD        >>                     *
13111           !                 << >>                                               *
13112           !          PROCEDURE ERRORON;                                         *
13113           !            OPTION  PRIVILEGED,UNCALLABLE;                           *
13114           !            BEGIN                                                    *
13115           !                    EQUATE    CPCB=4,                              *
13116           !                              SYXL=9,                              *
13117           !                              SYSL=3;                              *
13118           !                    DEFINE    SYSF=(11:1)#;                        *
13119           !                    DEFINE    SYXF=(6:1)#;                         *
13120           !                    DEFINE    TRAPFLD=(2:1)#;                      *
13121           !                    DEFINE    PCBPT = CURPRC#;                     *
13122           !                    INTEGER PXFIXEDLOC;                            *
13123           !                    LOGICAL STATUS=Q-1;                            *
13124           !                    INTEGER INDEX;                                 *
13125           !                    << >>                                          *
13126           !                    STATUS.TRAPFLD:=0;        <<TURN OFF TRAPS>>   *
```

```
RECORD  C S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13127           *                   IF PROCSTATE.SYSTEMPROCFLAG THEN RETURN;                *
13128           *                   PXFIXED;                                                *
13129           *                   TOS:=PXFXERRLEVEL;                                      *
13130           *                   IF = THEN STKINFO.INSYSTEMFLAG := 1;                    *
13131           *                   TOS := TOS+1;                                           *
13132           *                   ASSEMBLE(DUP,DUP);                                      *
13133           *                   PXFXERRLEVEL:=TOS;                                      *
13134           *                   INDEX:=TOS;                                             *
13135           *                   TOS:=TOS-6;                                             *
13136           *                   IF > THEN RETURN;                                       *
13137           *                   IF < THEN                                               *
13138           *                        BEGIN PXFXINTRERR:=0;     <<uses index>>           *
13139           *                              INDEX:=INDEX+1;                             *
13140           *                        END;                                              *
13141           *                   PXFXINTRERR:=0;                                         *
13142           *             END;                                                          *
13143           *                                                                           *
13144           *    Where the previously defined objects are:                              *
13145           *          LOGICAL POINTER                                                  *
13146           *             LPCB=3;               !SYSGLOB ENTRY # OF START OF PCBS       *
13147           *          DEFINE                                                           *
13148           *             CURRPRC=ABSOLUTE(4)#; !FIXED LOWMEM# OF CURRENT PCB           *
13149           *          EQUATE                                                           *
13150           *             STKINFOWORDNUM=3;     !STACK DST NUMBER                       *
13151           *          DEFINE                                                           *
13152           *             STKINFO=LPCB(PCBPT+STKINFOWORDNUM)#;                          *
13153           *          DEFINE                                                           *
13154           *             INSYSTEMFLAG=(1:1)#; !IF EXECUTING SYSTEM CODE                *
13155           *          EQUATE                                                           *
13156           *             PROCSTATEWORDNUM=9;                                           *
13157           *          DEFINE                                                           *
13158           *             PROCSTATE=LPCB(PCBPT+PROCSTATEWORDNUM)#;                      *
13159           *          DEFINE                                                           *
13160           *             SYSTEMPROCFLAG=(6:1)#; !SYSTEM PROCESS TYPE                   *
13161           *          INTEGER                                                          *
13162           *             S0=S-0;                                                       *
13163           *          ARRAY                                                            *
13164           *             QARRRAY(*) = Q+0;                                             *
13165           *          INTEGER ARRAY                                                    *
13166           *             QAM2(*) = Q-2;                                                *
13167           *          ARRAY                                                            *
13168           *             QP35(*) = QARRAY+35;                                          *
13169           *          DEFINE                                                           *
13170           *             PXFIXED=PUSH(Q,DL);                                           *
13171           *                << TOS = DL - Q >>                                         *
13172           *             PXFIXEDLOC := TOS-QAM2(S0)#;                                  *
13173           *                << (DL-Q) - contents of (DL-2) >>                          *
13174           *          DEFINE                                                           *
13175           *             PXFXERRLEVEL=QP35(PXFIXEDLOC)#,                               *
13176           *             PXFXINTRERR=QP35(PXFIXEDLOC+INDEX)#;                          *
13177           *                          <<INDEX must be defined in process>>             *
13178           *                          <<INDEX must start at 1>>                        *
13179           *                                                                           *
13180           *****************************************************************************
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	SPEC	SKIP	COMMENT
13182	0D80	ERON		P	INC			RONP	00E9		ADDL		CTR			Read next instruction into NIR
13184																Set up for LST 3
13185	0D81		0004		ADDL						ADD		BKX3			Set up for CPCB read (absolute 4)
13187																Use bank zero for absolute read
13188	0D82		UBA		ADD			ROX3	FFE0	REGN	ANDL					Read CPCB (absolute 4) into OPA
13190																Partial offset for LST 3
13191	0D83		0209	UBB	ADDL				001F	REGN	ANDL		BKX3			Get final offset for LST 3
13193																Get the bank number from the LST 3
13194	0D84		UBA	OPA	ADD		RH	ROB3	FFFF	Q	ADDL					Get PROCSTATE=(*(LST 3)+ *(abs4)+!209)
13196																Get pointer to status (Q-1) to mask
13197	0D85		UBB		ADD			ROS		P	INC		P			Read the status to mask user traps
13199																Increment P so it points to next inst.
13200	0D86		DFFF		ADDL				0200	OPB	ANDL				ZERO	Mask for user trap enable status off
13202																Test PROCSTATE.SYSTEMPROCFLAG bit
13203	0D87		UBA	OPA	AND			DATA	FFFE	DL	ADDL				NEXT	Write masked status back out
13205																Get DL-2, return if system process
13206		*** The next 6 lines are very dependent on each other and should ***														
13207		*** NOT be modified without careful study, especially the reads. ***														
13208	0D88		FFFA		ADDL				UBB		ADD		ROAS			Get (-6) for STKINFO address calcs
13210																Read (DL-2) for ERRORLEVEL computation
13211	0D89		0023	DL	ADDL				RH	UBA	ADD		ROX3			Get DL+!23 as part of ERRORLEVEL addr
13213																Read STKINFO into OPB for flag set
13214	0D8A		UBA	OPA	SUB		RH	ROBS			ADD					Read ERRORLEVEL into OPB, ptr in RH,
13216																where ERRORLEVEL= (DL- *(DL-2)+ !23)
13217	0D8B				ADD				4000	OPB	IORL		SP1B			Set STKINFO.INSYSTEMFLAG,but we will
13219																not write it til we check ERRORLEVEL
13220	0D8C				ADD					OPB	ADD		SP3B		NZRO	Test ERRORLEVEL for zero,as well as
13222																saving it in SP3B for later checks
13223	0D8D			UBB	INC			DATA		SP1B	ADD			DATA		Increment ERRORLEVEL and write it out
13225																Write modified STKINFO if ERRORLEVEL=0
13226		*** end of heavily dependent code...													***	
13227	0D8E		FFF9	UBA	ADDL				RH	UBA	ADD			WRS		Compute (orig ERRORLEVEL)-6 for tests
13229																Set up for PXFXINTRERR writes-indexed
13230	0D8F		UBA		ADD			NEG		UBA	INC				POS	Test ERRORLEVEL>6, skip next if false
13232																Test ERRORLEVEL<6 so skip extra clear
13233	0D90				ADD			NEXT			ADD			DATA		Return if > 6, thus not clearing any
13235																Conditional clear PXFXINTRERR, inc ptr
13236	0D91				ADD						ADD			DATA	NEXT	Always clear one PXFXINTRERR if <= 6,
13238																and then return from routine

```
 RECORD   C.S.      ********** ALU A **********  ************** ALU B **************
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13240           ************************************************************************
13241           *                                                                      *
13242           *      ERXT - REPLACEMENT FOR THE MPE PROCEDURE ERROREXIT              *
13243           *                                                                      *
13244           *           Macro-assembly code: %020104,%000031  (doubleword op)      *
13245           *               available in verion CX2603A and beyond                 *
13246           *                                                                      *
13247           *          This microcode routine replaces the MPE procedure           *
13248           *    ERROREXIT.  All the offsets to the locations used by this         *
13249           *    routine, and the different bit positions accessed by it, are      *
13250           *    hard coded as they are defined in U-MIT.                          *
13251           *          The routine is built on top of the new LST/SST instruc-     *
13252           *    tions which store the first 16 table pointers in the extended     *
13253           *    registers !E6 through !F5.                                        *
13254           *                                                                      *
13255           *  +---------------**********  WARNING  **********---------------+     *
13256           *  |    IF ANY OF THE OFFSETS OR BIT POSITIONS USED BY THIS CODE |     *
13257           *  |    SHOULD CHANGE IN FUTURE VERSIONS OF MPE, THIS CODE MUST  |     *
13258           *  |    BE MODIFIED OR IT WILL CEASE TO WORK PROPERLY.           |     *
13259           *  +-------------------------------------------------------------+     *
13260           *                                                                      *
13261           *          The original SPL source code, as defined in the module      *
13262           *    CHECKER is as follows :                                           *
13263           *                                                                      *
13264           *          <<RUN TIME (ABORT) ERRORS.                                  *
13265           *               IF STOV FLAG SET THEN ABORT                            *
13266           *                    IF ERROR LEVEL <=1                                *
13267           *               IF INTRINSIC #=99 THEN IGNORE CARRY AND NO ABORT       *
13268           *               IF ERRWORD<>0 THEN (ERROR)                             *
13269           *                    -INSERT ERRWORD                                   *
13270           *                    -IF INTRINSIC #=0 DO NOT ABORT                    *
13271           *                    -IF ERROR LEVEL <=0 THEN ABORT                    *
13272           *               RESET Q :=Q-DELTAQ                                     *
13273           *               CARRY BIT IN STATUS:= 1 IF ERROR                       *
13274           *                                  := 0 IF NO ERROR                    *
13275           *               DECREMENT ERROR LEVEL                                  *
13276           *               EXIT THROUGH THIS PREVIOUS STACK MARKER                *
13277           *                    USING PARAMETER COUNT (N)                         *
13278           *                                                                      *
13279           *               INTRINEXIT.(0:10) = INTRINSIC ERROR #                  *
13280           *                         .(10:6) = PARAMETER COUNT (N)                *
13281           *               ERRWORD.(0:8)     = PARAMETER                          *
13282           *                      .(8:8)     = ERROR BYTE                         *
13283           *               PARAM = ADDITIONAL INFO IF ABORT OCCURS   >>           *
13284           *                                                                      *
13285           *PROCEDURE ERROREXIT (INTRINEXIT,ERRWORD,PARAM);                       *
13286           *    VALUE INTRINEXIT,ERRWORD,PARAM;                                   *
13287           *    LOGICAL INTRINEXIT,ERRWORD,PARAM;                                 *
13288           *    OPTION PRIVILEGED,UNCALLABLE;                                     *
13289           *    BEGIN                                                             *
13290           *         EQUATE TYPE=1, MARK=2, MODE=[8/MARK,8/TYPE];                 *
13291           *         EQUATE SOTYPE=0, SOCODE=20, SOMODE=[8/MARK,8/SOTYPE];        *
13292           *         EQUATE CPCB=4,                                               *
13293           *                SYSL=3,                                               *
13294           *                SYXL=9,                                               *
13295           *                STOVL=9;                                              *
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13296            *          DEFINE STOVF=(5:1)#;                                      *
13297            *          DEFINE SYXF=(6:1)#;                                       *
13298            *          DEFINE SYSF=(11:1)#;                                      *
13299            *          DEFINE IFLD=(0:10)#,                                      *
13300            *                 NFLD=(10:6)#,                                      *
13301            *                 CFLD=(5:1)#;                                       *
13302            *          ARRAY QARRAY(*)=Q+0;                                      *
13303            *          INTEGER PXFIXEDLOC;                                       *
13304            *          INTEGER INDEX;                                            *
13305            *          INTEGER PCBPT;                                            *
13306            *          INTEGER KARRY=0;                                          *
13307            *          LOGICAL DELTAQ=Q+0, STATUS=Q-1;                           *
13308            *          << >>                                                     *
13309            *          PCBPT := CURRPRC;                                         *
13310            *          IF PROCSTATE.SYSTEMPROCFLAG THEN GOTO CONT;               *
13311            *          PXFIXED:                                                  *
13312            *          TOS:=PXFXERRLEVEL;          << DECR. ERROR LEVEL >>       *
13313            *          IF<= THEN                                                 *
13314            *               BEGIN TOS:=TOS+1;                                    *
13315            *                     INDEX:=0;                                      *
13316            *                     PXFXINTERR:=0;                                 *
13317            *                     INDEX:=1;                                      *
13318            *                     PXFXINTERR:=0;                                 *
13319            *               END;                                                 *
13320            *          TOS:=TOS-1;                                               *
13321            *          ASSEMBLE(DUP,DUP);                                        *
13322            *          PXFXERRLEVEL:=TOS;                                        *
13323            *          ASSEMBLE(TEST);                                           *
13324            *          IF > THEN GOTO CONTX;          <<STACK OVFLW CHECK>>      *
13325            *          STKINFO.INSYSTEMFLAG:=0;                                  *
13326            *          IF PROCSTATE.STOVFLAG                                     *
13327            *               AND NOT RESABORTINFO.CRITFLAG                        *
13328            *               AND NOT RESABORTINFO.HASSIRFLAG                      *
13329            *          THEN                                                      *
13330            *               BEGIN                                                *
13331            *               COMMENT: LET KERNEL ROUTINES KNOW PROCESS IS         *
13332            *                    ABORTING. NOTE THAT ERROREXIT DOESN'T CHECK     *
13333            *                    STOVABORT BIT SINCE CURRENTLY STOV->ABORT->     *
13334            *                    DEBUG->ERROREXIT MUST RESULT IN ANOTHER CALL    *
13335            *                    TO ABORT. STOVABORT BIT IS LEFT ON TO ALLOW     *
13336            *                    YOU TO GET INTO DEBUG. WE COUNT ON LAST CALL    *
13337            *                    TO ERROREXIT TO CATCH STOV AND GO TO ABORT;     *
13338            *               DISABLE;                                             *
13339            *               PROCSTATE.STOVFLAG:=0;  <<MUST TURN THIS OFF>>       *
13340            *               RESABORTINFO.STOVABORTFLAG:=1;                       *
13341            *               ENABLE;                                              *
13342            *               ABORT(SOMODE,SOCODE,0);                              *
13343            *               END;                                                 *
13344            *  CONTX: TOS:=ERRWORD;          <<ERROR WORD>>                       *
13345            *          IF = THEN GOTO CONT;   <<NO ERROR>>                       *
13346            *          KARRY:=1;                                                 *
13347            *          ASSEMBLE(XCH,DUP);                                        *
13348            *          IF TOS>=6 THEN                                            *
13349            *               BEGIN DEL;                                           *
13350            *                     TOS:=5;                                        *
13351            *               END;                                                 *
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13352           *         INDEX:=TOS+1;                                              *
13353           *         PXFXINTERR:=TOS;        <<PUT ERROR WORD>>                 *
13354           *         ASSEMBLE(TEST);                                            *
13355           *         IF > THEN GOTO CONT;    <<NESTED ERROR>>                   *
13356           *         IF INTRINEXIT.IFLD=0 THEN GOTO CONT;                       *
13357           *         ABORT(MODE,INTRINEXIT,PARAM);                              *
13358           *  CONT:  TOS:=%031400;           <<EXIT INSTRUCTION>>                *
13359           *         TOS:=TOS LOR INTRINEXIT.NFLD;                              *
13360           *         TOS:=KARRY;                                                *
13361           *         TOS:=INTRINEXIT.IFLD;                                      *
13362           *         ASSEMBLE(SED 0);                                           *
13363           *         PUSH(Q);                                                   *
13364           *         TOS:=DELTAQ;                                               *
13365           *         TOS:=TOS-TOS;                                              *
13366           *         SET(Q);                                                    *
13367           *         IF TOS=99 THEN DEL ELSE STATUS.CFLD:=TOS;                  *
13368           *         ASSEMBLE(XEQ 0);        << EXIT N >>                       *
13369           *         HELP;                   <<DEBUG LINKING CALL>>             *
13370           *  END;                                                              *
13371           *                                                                    *
13372           *    where :                                                         *
13373           *                                                                    *
13374           *    Current PCB number in absolute(4)                               *
13375           *    Sysglob entry for start of PCB's in LST 3                       *
13376           *    Pointer to PXFIXED in (DL-2)                                    *
13377           *    Errorlevel at PXFIXED+35                                        *
13378           *    Stack information for process at offset PCB+3                   *
13379           *    Process state information for process at offset PCB+9           *
13380           *    Abort information for process at offset PCB+0                   *
13381           *                                                                    *
13382           *    IDENTIFIER      CLASS         TYPE   ADDRESS                    *
13383           *                                                                    *
13384           * CFLD               DEFINE                          (5:1)         *
13385           * CONT               LABEL                 PB+165                  *
13386           * CONTX              LABEL                 PB+134                  *
13387           * CPCB               EQUATE                        VALUE=%4        *
13388           * DELTAQ             SIMP.VAR  LOGICAL     Q+000                   *
13389           * ERRWORD            SIMP.VAR  LOGICAL     Q-005                   *
13390           * IFLD               DEFINE                          (0:10)        *
13391           * INDEX              SIMP.VAR  INTEGER     Q+002                   *
13392           * INTRINEXIT         SIMP.VAR  LOGICAL     Q-006                   *
13393           * KARRY              SIMP.VAR  INTEGER     Q+004                   *
13394           * MARK               EQUATE                        VALUE=%2        *
13395           * MODE               EQUATE                        VALUE=%1001     *
13396           * NFLD               DEFINE                          (10:6)        *
13397           * PARAM              SIMP.VAR  LOGICAL     Q-004                   *
13398           * PCBPT              SIMP.VAR  INTEGER     Q+003                   *
13399           * PXFIXEDLOC         SIMP.VAR  INTEGER     Q+001                   *
13400           * QARRAY             ARRAY     LOGICAL     Q+000                   *
13401           * SOCODE             EQUATE                        VALUE=%24       *
13402           * SOMODE             EQUATE                        VALUE=%1000     *
13403           * SOTYPE             EQUATE                        VALUE=%0        *
13404           * STATUS             SIMP.VAR  LOGICAL     Q-001                   *
13405           * STOVF              DEFINE                          (5:1)         *
13406           * STOVL              EQUATE                        VALUE=%11       *
13407           * SYSF               DEFINE                          (11:1)        *
```

```
 RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13408          * SYSL              EQUATE                          VALUE=%3          *
13409          * SYXF              DEFINE                            (6:1)           *
13410          * SYXL              EQUATE                          VALUE=%11         *
13411          * TYPE              EQUATE                          VALUE=%1          *
13412          *                                                                     *
13413          *                                                                     *
13414          *    NOTE : Procedure ERROREXIT requires three parameters, i.e        *
13415          *    INTRINEXIT, ERRWORD, and PARAM. This microcode version           *
13416          *    assumes that these parameters are pushed onto the stack          *
13417          *    previous to calling the routine, and that they will reside       *
13418          *    in TOS registers RC,RB, and RA respectively.                     *
13419          *         When we exit the microcode routine, the condition codes     *
13420          *    will be set to pattern CCA, where CCE means an exit thru         *
13421          *    the previous stack marker (taken care of in the define),         *
13422          *    and CCG means that we have to go to ABORT. If the latter is      *
13423          *    true, the necessary parameters for the ABORT procedure will      *
13424          *    be pushed onto the stack by ERXT.                                *
13425          *                                                                     *
13426          ***********************************************************************
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	SPEC	SKIP	COMMENT
13428	0D92	ERXT			ADD					P	INC		P	RONP		Increment P and store new instr. in NIR
13430	0D93				JSZ	PUL2		SRL2			ADD		BKX3			jump to PUL2 if we have less than 2 valid TOS registers
13432																
13433																read from BANK 0
13434	0D94				ADD			SRG2	0004		ADDL					Check for at least 3 valid TOS registers
13436																UBB gets 4 (for abs(4) read)
13437	0D95				JSZ	PULM		UNC	UBB		ADD			ROX3		Pull one item from memory to TOS regs.
13439																read abs(4)
13440	0D96				ADD				00E9		ADDL		CTR			set up for LST 3
13442	0D97				ADD						ADD					waiting for the store in CTR
13444	0D98				ADD				FFE0	REGN	ANDL					get address bits for LST 3
13446	0D99		0209	UBB	ADDL				001F	REGN	ANDL		BKX3			get partial address of PROCSTATE
13448																set the correct bank for future reads
13449	0D9A		FFFE	DL	ADDL				UBA	OPB	ADD		RH	ROX3		compute DL-2 for future use
13451																read PROCSTATE, store pointer in RH
13452	0D9B		UBA		ADD			ROS	FFFA	UBB	ADDL		SP1B			read from (DL-2) into OPA
13454																SP1B gets the address of STKINFO
13455	0D9C		0023	DL	ADDL				0200	OPB	ANDL				ZERO	compute DL+!23 for future use
13457																test if SYSTEMPROCFLAG is on
13458	0D9D		UBA	OPA	SUB		SP0A	ROS			JSB	CNT0			UNC	read ERRORLEVEL, store pointer in SP0A
13460																if PROCSTATE.SYSTEMPROCFLAG on,jmp to CNT0
13461	0D9E				ADD					OPB	ADD		SP2B			store PROCSTATE in SP2B
13463	0D9F			OPA	ADD		SP4A			SP1B	ADD			ROX3		store ERRORLEVEL in SP4A
13465																read STKINFO into OPB
13466	0DA0		UBA		ADD			ZERO			ADD					if ERRORLEVEL = 0, skip the jump
13468	0DA1				JSB	LBL1		POS	BFFF	OPB	ANDL		SP1B			If ERRORLEVEL >0 go to LBL1
13470																Prepare to clear STKINFO.INSYSTEMFLAG
13471	0DA2			SP4A	INC		SP4A				ADD					else, increment ERRORLEVEL
13473	0DA3		SP0A		INC			WRS			ADD					increment the pointer to ERRORLEVEL, and set up for the write
13475																
13476	0DA4				ADD			DATA			ADD					clear (original ERRORLEVEL location) +1
13478	0DA5	LBL1	SP0A		ADD			WRS	FFFF	SP4A	ADDL		SP3B			set up to write into (orig. ERRORLEVEL loc.)
13480																ERRORLEVEL gets ERRORLEVEL-1, store new value in SP3B for future use
13481																
13482	0DA6			UBB	ADD			DATA	UBB		ADD				ZERO	write new data into original location
13484																if new ERRORLEVEL = 0, skip the jump
13485	0DA7		FFF7	RH	ADDL						JSB	CONX			POS	get address of RESABORTINFO
13487																if new ERRORLEVEL > 0, go to CONX
13488	0DA8			UBA	ADD			ROX3		SP1B	ADD			DATA		read RESABORTINFO into OPA
13490																if new ERRORLEVEL<=0, clear INSYSTEMFLAG
13491	0DA9				ADD				FBFF	SP2B	ANDL		SP2B			Prepare to clear PROCSTATE.STOVFLAG (2604)
13493	0DAA		3000	OPA	ANDL				0400	SP2B	ANDL					test both, RESABORTINFO.CRITFLAG and RESABORTINFO.HASSIRFLAG
13495																
13496																get PROCSTATE.STOVFLAG
13497																**this line MUST immediately follow the previous one for SP2B to be correct***
13498																
13499	0DAB			UBA	ADD			ZERO	UBB		JSB	CONX			ZERO	if either CRITFLAG or HASSIRFLAG are set then jump to CONX
13501																
13502																if PROCSTATE.STOVFLAG=0 then jump to CONX
13503	0DAC				JSB	CONX		UNC		RH	ADD			WRX3		jump to CONX
13505																set up for the write into PROCSTATE (2604)
13506	0DAD		0002	OPA	IORL					SP2B	ADD			DATA		prepare to set RESABORTINFO.STOVABORTFLAG
13508																clear PROCSTATE.STOVFLAG
13509	0DAE			UBA	ADD			DATA			ADD					set RESABORTINFO.STOVABORTFLAG
13511	0DAF				JSB	ABRT		UNC			ADD					Jump to ABORT
13513	0DB0	CONX			ADD				FFF9	SP4A	ADDL		RH			Get (new ERRORLEVEL)- 6 into UBB and RH

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
13515	0DB1			UBB	ADD			NEG	0006		ADDL					If (new ERRORLEVEL)<6, skip
13517																Put 6 into UBB
13518	0DB2		SP0A	UBB	ADD			WRS		RB	ADD				ZERO	else, set up to write PXFXINTERR location where INDEX=6
13520																
13521																if ERRWORD(in RB)=0, take jump to CNT0
13522	0DB3			RH	ADD			POS	0400		ADDL		RH		EVEN	skip next line if INDEX = 6
13524																use RH as a mask depending on entry point to CNT1/CNT0
13525																
13526	0DB4		SP0A	SP4A	ADD			WRS			JSB	CNT0			UNC	set up to write PXFXINTERR location where INDEX= ERRORLEVEL
13528																
13529																go to CNT0
13530	0DB5			RB	ADD			DATA	SP3B		ADD				ZERO	Write ERRWORD into correct location
13532																Test ERRORLEVEL-1
13533	0DB6				ADD						JSB	CNT1			UNC	If entry flag = 0, go to CNT1
13535	0DB7				ADD				FFC0	RC	ANDL				NZRO	If INTRINEXIT(in RC).(0:10)<>0, go to ABORT
13537	0DB8				ADD						JSB	CNT1			UNC	else, go to CNT1
13539	0DB9				ADD						JSB	ABT1			UNC	go to ABORT
13541	0DBA	CNT0			ADD				FBFF		ADDL		RH			set up mask, depending on entry point. store it in RH
13543																
13544	0DBB	CNT1		Q	ADD			ROBS	003F	RC	ANDL		SP1B			Read delta Q into OPB
13546																these bits determine how many items to pop off the stack when we exit
13547																
13548	0DBC		FFC0	RC	ANDL					UBA	ADD					UBA <- INTRINEXIT(0:10)
13550																UBB <- Q
13551	0DBD		E740	UBA	ADDL				UBB	OPB	SUB		Q	-		UBA <- INTRINEXIT(0:10) - 99
13553																Q <- Q - DELTAQ
13554	0DBE			UBA	JSB	EXT		ZERO	FFFF	UBB	ADDL					If INTRINEXIT(0:10) - 99 = 0, go to EXT
13556																UBB <- Q -1 (STATUS)
13557	0DBF				ADD				UBB		ADD			ROS		Read STATUS into OPB
13559	0DC0				ADD				RH		ADD				EVEN	If error, skip next line
13561	0DC1				ADD				RH	OPB	AND			DATA		Clear STA.CFLD
13563	0DC2				ADD				RH		ADD				ODD	If no error, skip next line
13565	0DC3				ADD				RH	OPB	IOR			DATA		Set STA.CFLD
13567	0DC4	EXT			ADD				3300	SP1B	IORL		RA			TOS <- exit (N) instruction
13569	0DC5				ADD						ADD			CCA	NEXT	EXIT and set condition code to CCE
13571	0DC6	ABRT			ADD				0014		ADDL		RB			push SOCODE onto TOS-1
13573	0DC7				ADD				0200		ADDL		RC			push SOMODE onto TOS-2
13575	0DC8				ADD		RA			UBB	ADD			CCA	NEXT	push Q onto TOS
13577																set CCA to CCG and EXIT
13578	0DC9	ABT1			ADD				0201		ADDL		RC			push MODE onto TOS-2
13580	0DCA				ADD					RC	ADD		RB			push INTRINEXIT onto TOS-1
13582	0DCB				ADD					RC	ADD			CCA	NEXT	

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13584           ************************************************************************
13585           *                                                                      *
13586           *       XGDB - REPLACEMENT FOR THE MPE PROCEDURE EXCHANGEDB            *
13587           *                                                                      *
13588           *            Macro-assembly code: %020104,%000032  (doubleword op)     *
13589           *                 available in version CX2603A and beyond              *
13590           *                                                                      *
13591           *    This microcode routine replaces the MPE procedure EXCHANGEDB,     *
13592           *    which switches the DB and DBank software registers to point to    *
13593           *    a specific data segment (passed in as the DST# of the segment)    *
13594           *    or to the program's own stack segment (passed as 0).  The         *
13595           *    microcode handles all cases except absent data segments, which    *
13596           *    are detected and passed back to the software to handle.  This     *
13597           *    is flagged by the condition codes: CCE means the microcode was    *
13598           *    able to handle it, CCG means the software must.                   *
13599           *                                                                      *
13600           *  +--------------**********  WARNING  **********---------------+      *
13601           *  |    IF ANY OF THE OFFSETS OR BIT POSITIONS USED BY THIS CODE |      *
13602           *  |    SHOULD CHANGE IN FUTURE VERSIONS OF MPE, THIS CODE MUST  |      *
13603           *  |    BE MODIFIED OR IT WILL CEASE TO WORK PROPERLY.           |      *
13604           *  +-------------------------------------------------------------+      *
13605           *                                                                      *
13606           *    The SPL source used to create this routine is as follows:         *
13607           *                                                                      *
13608           *   LOGICAL PROCEDURE EXCHANGEDB(WHERE);                               *
13609           *   VALUE WHERE;                                                       *
13610           *   INTEGER WHERE;                                                     *
13611           *   OPTION PRIVILEGED,UNCALLABLE;                                      *
13612           *                                                                      *
13613           *                                                                      *
13614           *   COMMENT                                                            *
13615           *                                                                      *
13616           *   EXCHANGEDB IS CALLED TO PUT DB AT THE BASE OF A DATA SEGMENT       *
13617           *   OR TO RETURN DB TO THE CALLER'S STACK DB.  THE DESTINATION DATA    *
13618           *   SEGMENT NUMBER IS SUPPLIED AS PARAMETER IF NOT RETURNING TO THE    *
13619           *   STACK.  IF RETURNING TO THE STACK, SUPPLY 0 AS PARAMETER.          *
13620           *                                                                      *
13621           *   EXCHANGEDB RETURNS THE DST NUMBER OF WHERE DB WAS (0 IF STACK).    *
13622           *   THIS VALUE MAY BE SAVED AND RETURNED ON THE NEXT CALL TO EXCHAN    *
13623           *   TO RESTORE THE PREVIOUS ENVIRONMENT.                               *
13624           *                                                                      *
13625           *   ;                                                                  *
13626           *                                                                      *
13627           *   BEGIN                                                              *
13628           *                                                                      *
13629           *   LOGICAL DELTAP=Q-2;                                                *
13630           *   LOGICAL STATUS=Q-1;                                                *
13631           *   INTEGER PROCINX,                                                   *
13632           *           PCBPT;                                                     *
13633           *   DOUBLE OBJ := 0D;                                                  *
13634           *   LOGICAL ARRAY OBJIDENT(*)=OBJ;                                     *
13635           *   INTEGER DESCSTINX;                                                 *
13636           *                                                                      *
13637           *   TURNOFFTRAPS;                                                      *
13638           *                                                                      *
13639           *   PDISABLE;                                                          *
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13640           *    <<MMSTAT'(MMSTATEXCHDB,WHERE,DELTAP,STATUS,0,0,0);>>                *
13641           *    IF PMBCFIRMWARE                                                      *
13642           *       THEN UPDATE'ICS'XDSEGBNKCELL;                                     *
13643           *    TOS:=%10000D;                                                        *
13644           *    ASMB(XCHD);   <<FOR FAST ADDRESSING>>                                *
13645           *    PROCINX := PCBPT := CURPRC;                                          *
13646           *    TOS:=DBXDSINFO;                                                      *
13647           *    IF < THEN SUDDENDEATH(611); <<MUST RESET FIRST>>                     *
13648           *    EXCHANGEDB:=TOS.XDSDSTFIELD;                                         *
13649           *    TOS:=WHERE;                                                          *
13650           *    TOS.(0:1):=0;   <<DISCARD CLEAN BIT>>                                *
13651           *    DBXDSINFO.XDSDSTFIELD := S0;                                         *
13652           *    OBJIDENT(ObjIDnumField):=S0;                                         *
13653           *    IF S0=0 THEN                                                         *
13654           *       BEGIN  <<DB GOES TO STK>>                                         *
13655           *       TOS:=ICS(-ICS'STKBANKCELL);                                       *
13656           *       TOS:=ICS(X:=X+1);                                                 *
13657           *       ASMB(XCHD);                                                       *
13658           *       END                                                               *
13659           *    ELSE                                                                 *
13660           *       BEGIN <<GOING TO AN XDS>>                                         *
13661           *          IF DST(0) < S0 THEN SUDDENDEATH(611); <<OUT OF RANGE>>         *
13662           *       DISABLE;                                                          *
13663           *       DESCSTINX:=TOS & LSL(2);                                          *
13664           *       IF DST(DESCSTINX)=%100000 THEN                                    *
13665           *          SUDDENDEATH(625);                                              *
13666           *       IF NOT LOGICAL(DST(DESCSTINX)).ABSENTFLAG THEN                    *
13667           *          BEGIN  <<NEW DB SEG IS PRESENT>>                               *
13668           *          IF PMBCFIRMWARE                                                *
13669           *                                                                         *
13670           *                                                                         *
13671           *            THEN BEGIN                                                   *
13672           *            X := DESCSTINX;                                              *
13673           *            GET'XDSEG'LIMITS;                                            *
13674           *            XFER'XDSEG'LIMITS;                                           *
13675           *            END;                                                         *
13676           *                                                                         *
13677           *          X := DESCSTINX;                                                *
13678           *          TOS:=DST(X:=X+2);                                              *
13679           *          TOS:=DST(X:=X+1);                                              *
13680           *          ASMB(XCHD);                                                    *
13681           *          END                                                            *
13682           *       ELSE                                                              *
13683           *          BEGIN <<NOT PRESENT>>                                          *
13684           *          ENABLE;                                                        *
13685           *          IF LOGICAL(DST(X:=X+1)).ROCFLAG THEN                           *
13686           *             BEGIN                                                       *
13687           *             IF GCLASSENABLEDMASK.CLASS0 THEN                            *
13688           *                BEGIN  <<MEASURE RECOVERY OF DATA SEG BY PROCESS>>       *
13689           *                TOS:=MEASSTATXDSBANK;                                    *
13690           *                TOS:=MEASSTATXDSBASE;                                    *
13691           *                TOS:=TOS+COSUB0'SEGRELOFF+C'DATARECOVERY;                *
13692           *                ASMB(LSEA);                                              *
13693           *                TOS:=TOS+1;                                              *
13694           *                ASMB(SSEA;DDEL);                                         *
13695           *                END;                                                     *
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13696           *            RECOVEROC(OBJ,DESCSTINX,0D);                          *
13697           *            END                                                   *
13698           *         ELSE                                                     *
13699           *            BEGIN  <<REALLY ABSENT>>                              *
13700           *            QueueOnObject(OBJ);                                   *
13701           *            PDISABLE;                                             *
13702           *            END;                                                  *
13703           *                                                                  *
13704           *                                                                  *
13705           *            IF PMBCFIRMWARE  THEN                                 *
13706           *                                                                  *
13707           *              BEGIN                                               *
13708           *              X := DESCSTINX;  << SET X = TO DST ENTRY >>          *
13709           *              GET'XDSEG'LIMITS;                                   *
13710           *              XFER'XDSEG'LIMITS;                                  *
13711           *              END;                                                *
13712           *                                                                  *
13713           *         X := DESCSTINX;                                          *
13714           *         TOS:= DST(X:= X+2);                                      *
13715           *         TOS:=DST(X:=X+1);                                        *
13716           *         ASMB(XCHD);                                              *
13717           *         END;                                                     *
13718           *      DST(DESCSTINX).REFERENCEDFLAG:=1;                           *
13719           *      END;                                                        *
13720           *   PENABLE;                                                       *
13721           *   END  <<PROCEDURE EXCHANGEDB>>;                                 *
13722           *                                                                  *
13723           *   And now for the microcode procedure...                         *
13724           *                                                                  *
13725           ********************************************************************
```

RECORD NO	C S ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
13727	0DCC	XGDB	UBA		CAD			ROX3	00E9		ADDL		CTR			Read CURPRC = absolute(4)
13729																Set up for LST 3 (DBXDSINFO)
13730	0DCD				JSB	XDIE		SR7	UBA	Q	RSUB			ROS		Check to make sure RG is scratchpad
13732																Read WHERE from address (Q-4)
13733	0DCE		0202	OPA	ADDL				FFE0	REGN	ANDL					More LST 3 calculations (SYSGLOB + 2)
13735																Mask LST 3 for offset
13736	0DCF		UBA	UBB	ADD		SP4A	ROX4	001F	REGN	ANDL		BKX4			Full LST 3 read of DBXDSINFO
13738																Mask LST 3 for bank and put in BKX4
13739	0DD0		SPOA	Q	RSUB			WRS	3FFF		ADDL		RH			Set up return value address (Q - 5)
13741																Set up mask for XDSDSTFIELD (2:14)
13742	0DD1			OPA	JSB	XDIE		NEG		P	INC		P	RONP		Die if LST 3 (DBXDSINFO) < 0
13744																Move P up for doubleword instruction
13745	0DD2		RH	OPA	AND			DATA	7FFF	OPB	ANDL		SP1B			Mask and write return value to (Q-5)
13747																Mask high bit of WHERE, store in SP1B
13748	0DD3		C000	OPA	ANDL		SPOA		UBB	UBB	JSB	GOXD	SP2B		NZRO	Mask off low bits of DBXDSINFO for DPF
13750																Jump off to handle XDS if WHERE <> 0
13751	0DD4		01FC		ADDL		RG		00ED		ADDL		CTR			Offset of LST 7: ICS(-ICS_ABSTKDBCELL)
13753																Set up for LST 7 access to REGN
13754	0DD5	XFIN		SP4A	ADD			WRX4	RH	SP1B	AND					Set up for SST 3 write (DBXDSINFO)
13756																Mask WHERE for SST 3 deposit (2:14)
13757	0DD6		SPOA	UBB	IOR			DATA	FFE0	REGN	ANDL					Write SST 3 (DBXDSINFO.XDSDSTFIELD)
13759																Mask for offset of LST 7/2 (ICS/DST)
13760	0DD7		UBB	RG	ADD			ROB3	001F	REGN	ANDL		BKX3			First full LST read (ICS or DST)
13762																Mask LST for bank to read from
13763	0DD8		UBA		CAD			ROB3			ADD			CCA		Second LST read of ICS or DST
13765																Clear condition code for good return
13766	0DD9				ADD					OPB	ADD		DB			Put results of first read into DB
13768	0DDA				ADD					OPB	ADD		BNKD			Put results of second read into DBANK
13770	0DDB		RH		INC				UBB	BNKS	XOR				ZERO	Set up !4000 for possible FSS set
13772																Check BNKS = BNKD for split stack mode
13773	0DDC		1000	UBA	IORL				UBA		ADD			CCPX	UNC	Set up !5000 for possible FSS clear
13775																Set FSS (split stack) and skip clear
13776	0DDD				ADD			NEXT	UBA		ADD			CCPX		Return to caller , but not before...
13778																Clear FSS (split stack) if not skipped
13779	0DDE	GOXD			ADD				00E8		ADDL		CTR			Set up for LST 2 read (DST)
13781	0DDF			SP2B	ADD	LSL			0200		ADDL		SP2B			Get (WHERE << 2) for LST 2 (DST)
13783																Get %1000 for SYSGLOB relative address
13784	0DE0		UBA	UBB	ADD				FFE0	REGN	ANDL		SP3B			Add %1000 to (WHERE << 2) for SYSGLOB
13786																Mask offset for LST 2 (DST)
13787	0DE1		UBB	UBA	ADD			ROX3	001F	REGN	ANDL		BKX3			Full LST 2 (DST.DESCSTINX) read
13789																Mask LST 2 for bank of read (DST)
13790	0DE2		0003	SREG	ADDL		RG		SP3B	SP2B	ADD			ROX3		Set up for LST 2 of XCHD,save in XRO
13792																Read LST 2 (DST(0)) for cmp to WHERE
13793	0DE3			OPA	JSB	XDIE		NEG	FFFF	SP1B	ADDL					Die if bit 0 of DST.DESCSTINX is on
13795																Use WHERE-1 for compare to LST 2
13796	0DE4		2000		ADDL				UBB	OPB	JSBS	XDIE			POS	Bit set for SST 2: DST(DESCSTINX).REF
13798																Die if LST 2 < WHERE
13799	0DE5		UBA	OPA	IOR			DATA			JSB	XFIN			UNC	Write SST 2 (DST(DESCSTINX).REFERENC
13801																Finish up with code above...
13802	0DE6	XDIE			ADD						INC			CCA		Set condition code to show not handle
13804	0DE7				ADD						ADD				NEXT	Go to next macro instruction.
13806																set CCA to CCG and EXIT
13807	0DE8				ADD						ADD					NOP to keep system happy
13809	0DE9				ADD						ADD					NOP to keep system happy

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13812           ************************************************************************
13813           *                                                                      *
13814           *       TIMR - REPLACEMENT FOR THE MPE DOUBLE PROCEDURE TIMER          *
13815           *                                                                      *
13816           *            Macro-assembly code: %020104,%000033  (doubleword op)     *
13817           *                 available in version CX2603A and beyond              *
13818           *                                                                      *
13819           *    This microcode routine replaces the MPE procedure TIMER only      *
13820           *  for the SERIES 64 machines.  All the offsets to the locations       *
13821           *  used by this routine and the different bit positions accessed       *
13822           *  by it are hard coded as they are defined in U-MIT.                  *
13823           *                                                                      *
13824           *    This routine is built on top of the new LST/SST instructions      *
13825           *  which store the first 16 table pointers in the extended regs.       *
13826           *  !E6 through !F5.                                                    *
13827           *                                                                      *
13828           *  +---------------**********  WARNING  **********----------------+    *
13829           *  |    IF ANY OF THE OFFSETS OR BIT POSITIONS USED BY THIS CODE |    *
13830           *  |    SHOULD CHANGE IN FUTURE VERSIONS OF MPE, THIS CODE MUST  |    *
13831           *  |    BE MODIFIED OR IT WILL CEASE TO WORK PROPERLY.           |    *
13832           *  +-------------------------------------------------------------+    *
13833           *                                                                      *
13834           *    This routine returns an unsigned 31 bit double word.              *
13835           *  This quantity represents the number of milliseconds since the      *
13836           *  midnight preceeding the last cold load.  This quantity will be     *
13837           *  reset to zero on 24-day intervals at exactly 12 o'clock midnight   *
13838           *  Detection and correction of this case between two calls to TIMR     *
13839           *  (less than 24 days apart) can be done as follows :                  *
13840           *                                                                      *
13841           *    If the current returned TIMR value is equal or less than the      *
13842           *  last TIMR value, add 2073600000 (the number of MS in 24 days)       *
13843           *  to the result.                                                      *
13844           *                                                                      *
13845           *    The millisecond count since the midnight preceeding cold load     *
13846           *  is computed in number of clock ticks by adding the count regis-     *
13847           *  ter (XR31) of the system clock with the overflow counter (TRL(5)    *
13848           *  and TRL(6)).  The procedure TICK will reset the count in TRL        *
13849           *  every 24 hours and the number of days since cold load is stored     *
13850           *  in TRL(3).  TRL(3) is reset every 24 days and the base Julian       *
13851           *  date is updated accordingly.                                        *
13852           *                                                                      *
13853           *    This routine places the LSW of the result in Q-4 and the MSW of   *
13854           *  the result in Q-5(1:15).                                            *
13855           *                                                                      *
13856           *    The SPL code is as follows :                                      *
13857           *                                                                      *
13858           * DOUBLE PROCEDURE TIMER;                                              *
13859           * OPTION PRIVILEGED;                                                   *
13860           *                                                                      *
13861           *                                                                      *
13862           * BEGIN                                                                *
13863           *                                                                      *
13864           *    LOGICAL S33 := TRUE;                                              *
13865           *    INTEGER NUM'OF'DAYS, TICK'IN'INT  = 1100,                         *
13866           *            MSW = Q-5, LSW = Q-4;       << RETURN VALUE >>            *
13867           *    DOUBLE  MS, NUM'OF'TICKS, NUM'TICK'PER'DAY  = 9447100000D;        *
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13868           *    REAL    TICK'TO'MS := 9.14566375E-2;<<CONVERT TICK TO MS>>      *
13869           *                                                                     *
13870           *    DISABLE;                                                         *
13871           *    PUSH(STATUS); TOS.(2:1):=0; SET(STATUS);                         *
13872           *    ASMB(PCN);  << GET CPU # >>                                      *
13873           *    IF TOS <> SERIES'33  THEN                                        *
13874           *    BEGIN                                                            *
13875           *       S33 := FALSE;                                                 *
13876           *       NUM'TICK'PER'DAY := MS'PER'DAY;                               *
13877           *    END;                                                             *
13878           *    TOS := 0;                                                        *
13879           *    ASMB(RCCR);                              << COUNT REG VALUE >>   *
13880           *    IF S33  THEN                                                     *
13881           *    BEGIN                                                            *
13882           *       TOS := F(TEMPLR);  << LAST UPDATED COUNT REG VALUE >>         *
13883           *       ASMB(LSUB);                                                   *
13884           *    END;                                                             *
13885           *     TOS := TRLDTIME1;                                               *
13886           *     TOS := TRLDTIME2;                                               *
13887           *    ASMB(DADD, DDUP);                                                *
13888           *     NUM'OF'DAYS := TRLNUMDAYS;                                      *
13889           *    TOS := NUM'TICK'PER'DAY;  << CHECK IF OVER A DAY >>              *
13890           *    ASMB(DSUB, DDEL);                                                *
13891           *    IF >=  THEN                                                      *
13892           *    BEGIN                                                            *
13893           *       NUM'OF'TICKS := TOS - NUM'TICK'PER'DAY;                       *
13894           *       NUM'OF'DAYS := IF NUM'OF'DAYS = 23  THEN  0  ELSE             *
13895           *                      NUM'OF'DAYS + 1;                               *
13896           *    END  ELSE  NUM'OF'TICKS := TOS;                                  *
13897           *    MS := IF S33  THEN  FIXR(REAL(NUM'OF'TICKS)*TICK'TO'MS)          *
13898           *          ELSE  NUM'OF'TICKS;                                        *
13899           *                                                                     *
13900           *    TOS := DOUBLE(NUM'OF'DAYS)*MS'PER'DAY + MS;                      *
13901           *    LSW := TOS;                                                      *
13902           *    MSW := TOS.(1:15);                                               *
13903           * END;    << TIMER >>                                                 *
13904           *                                                                     *
13905           *     This routine is not meant to be called by the users, but if a   *
13906           *   user calls it, he/she needs to make sure that SR = 0.             *
13907           *                                                                     *
13908           ***********************************************************************
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13912         %E00
13913   0E00  TIMR                 ADD       SP0A        00F0       ADDL      CTR              SP0A <- 0 for the multiplication
13915                                                                                           Set up for LST %12
13916   0E01        0015           ADDL                                 ADD       BKX3             Set up for reading TR; Read from bank 0
13918   0E02        0013           ADDL                  UBA            ADD            ROA3      Set up for reading LR; Read TR into OPA
13920   0E03        UBA            ADD             ROB3  FFE0 REGN ANDL                         Read LR into OPB
13922                                                                                           Partial offset for LST %12
13923   0E04              OPA      ADD                   0206 UBB  ADDL      SP2B             UBA <- TR
13925                                                                                           UBB <- offset for TRLDTIME2
13926   0E05        FFFF  UBB      ADDL      RH          UBA  XR31 ADD                        UBA <- offset for TRLDTIME1
13928                                                                                           UBB <- TR+CR, where CR is loaded into XR31
13929                                                                                             by the TCLK routine.
13930   0E06        FFFE  UBA      ADDL      SP4A        UBB  OPB  ADD       RA               SP4A <- offset for TRLNUMDAYS
13932                                                                                           RA <- SYS CLK CTR = TR+LR+CR
13933   0E07                       ADD                   001F REGN ANDL      BKX3             Get bank number for LST %12
13935   0E08                       ADD                         RH  ADD            ROX3       Read TRLDTIME1 into OPB
13937   0E09              SP4A     ADD             ROX3                ADD       RB               Read TRLNUMDAYS into OPA
13939                                                                                           RB <- 0
13940   0E0A              SP2B     ADD             ROX3                ADD       SP2B             Read TRLDTIME2 into OPA
13942                                                                                           SP2B <- 0, for the multiply
13943   0E0B              OPA      ADD       RH                  P     INC       P     RONP       RH <- TRLNUMDAYS
13945                                                                                           Increment P and put the new value in NIR.
13946   0E0C              OPA      ADD                           OPB   ADD                        UBA <- TRLDTIME2
13948                                                                                           UBB <- TRLDTIME1
13949   0E0D        UBB   RB       ADD                   UBA   RA    LINK      RC               Perform a double add, where
13951                                                                                             Y = 0,SYS CLK CTR + TRLDTIME1,TRLDTIME2
13952                                                                                           RC <- Y(LSW)
13953   0E0E              UBA      ADD       RD          5C00        ADDL                       RD <- Y(MSW)
13955                                                                                           UBB <- MS'PER'DAY(LSW)
13956   0E0F        UBB             ADD       SP4A        0526        ADDL      SP3B             UBB,UBA = SP3B,SP4A <- MS'PER'DAY
13958   0E10        RD    UBB      SUB                   RC    UBA   LINK      RA               Perform a double subtract, where
13960                                                                                             Z = Y - MS'PER'DAY
13961                                                                                           RA <- Z(LSW)
13962   0E11              UBA      ADD       RB                        ADD                      RB <- Z(MSW)
13964   0E12              UBA      JSB   TIM1        NEG   FFE9        ADDL                     If Z < 0, jump to TIM1
13966                                                                                           UBB <- -23
13967   0E13        RH    UBB      ADD               ZERO        RA    ADD       RC               If NUM'OF'DAYS = 23, skip next line
13969                                                                                           RC <- RA
13970   0E14              RH       INC       RH      NZRO              ADD                      NUM'OF'DAYS <- NUM'OF'DAYS + 1
13972   0E15                       ADD       RH                        ADD                      NUM'OF'DAYS <- 0
13974   0E16                       ADD                         RB    ADD       RD               RD <- RB
13976   0E17  TIM1  FFFB  Q        ADDL                  FFFC  Q     ADDL                       UBA <- POINTER TO Q-5
13978                                                                                           UBB <- POINTER TO Q-4
13979   0E18              UBA      ADD               WRS         UBB   ADD             WRS      Set up to write at Q-5
13981                                                                                           Set up to write at Q-4
13982   0E19                       ADD                                 REPN                001F  The next 3 lines perform a multiplication
13984                                                                                             NUM'OF'DAYS*MS'PER'DAY
13985   0E1A              UBA      MPAD                  RH    UBB   LINK            DCTR CTR0
13986   0E1B              UBA      ADD                         UBB   ADD
13987   0E1C              UBA      ADD                   SP3B        ADD       RA               RA <- NUM'OF'DAYS*MS'PER'DAY(LSW)
13989   0E1D        SP0A            ADD                               ADD                      UBA <- NUM'OF'DAYS*MS'PER'DAY(MSW)
13991   0E1E        RD    UBA      ADD                   RC    RA    LINK            DATA       Perform a double add, where RESULT =
13993                                                                                           (NUM'OF'DAYS*MS'PER'DAY)+MS, where MS=RD,RC
13994                                                                                           Q-4 <- RESULT(LSW)
13995   0E1F              UBA      ADD               DATA                ADD                NEXT  Q-5 <- RESULT(MSW); Exit!
```

```
 RECORD    C.S.      ********** ALU A **********  ********** ALU B **********
   NO      ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

13998            ************************************************************************
13999            *                                                                      *
14000            *      TMRQ - REPLACEMENT FOR THE MPE INTEGER PROCEDURE TIMEREQ        *
14001            *                                                                      *
14002            *           Macro-assembly code: %020104,%000034  (doubleword op)      *
14003            *                available in version CX2603A and beyond               *
14004            *                                                                      *
14005            *     This microcode routine replaces the MPE procedure TIMEREQ        *
14006            *  only for the series 64 machines.All the offsets to the locations    *
14007            *  used by this routine and the different bit positions accessed       *
14008            *  by it are hard coded as they are defined in U-MIT.                  *
14009            *                                                                      *
14010            *     This routine is built on top of the new LST/SST instructions     *
14011            *  which store the first 16 table pointers in the extended regs.       *
14012            *  !E6 through !F5.                                                    *
14013            *                                                                      *
14014            *  +---------------**********  WARNING  **********----------------+    *
14015            *  |     IF ANY OF THE OFFSETS OR BIT POSITIONS USED BY THIS CODE |    *
14016            *  |     SHOULD CHANGE IN FUTURE VERSIONS OF MPE, THIS CODE MUST  |    *
14017            *  |     BE MODIFIED OR IT WILL CEASE TO WORK PROPERLY.           |    *
14018            *  +-------------------------------------------------------------+    *
14019            *                                                                      *
14020            *     This routine sets up a time request. It accepts time up to       *
14021            *  2**32-1.  The most prioritary request is always first in the        *
14022            *  list.  All header entries are table word offsets. Entry # in        *
14023            *  entry is the number of the next entry.                              *
14024            *     This routine also returns an integer value to location Q-%10     *
14025            *  Upon exit, it sets the condition codes in the following manner :    *
14026            *  CCG = go to SUDDENDEATH (N), where N is already placed in TOS       *
14027            *  CCL = go to MPE'TABLE'FULL (N), N already in TOS                    *
14028            *  CCE = regular exit.                                                 *
14029            *                                                                      *
14030            *  NOTE : FOR PERFORMANCE REASONS, TRLENTRYSIZE (=4), WAS HARD         *
14031            *         CODED.  IF IT WERE TO CHANGE, THIS ROUTINE SHOULD CHANGE     *
14032            *         TO REFLECT THE NEW VALUE.                                    *
14033            *                                                                      *
14034            *  The SPL code is as follows :                                        *
14035            *                                                                      *
14036            *INTEGER PROCEDURE TIMEREQ(CODE,REQ,TIME);                            *
14037            *VALUE CODE,REQ,TIME;                                                 *
14038            *DOUBLE TIME;                                                         *
14039            *INTEGER CODE,REQ;                                                    *
14040            *OPTION UNCALLABLE,PRIVILEGED;                                        *
14041            *                                                                     *
14042            *                                                                     *
14043            *BEGIN                                                                *
14044            *      INTEGER  TRLX, S, T := 8;                                      *
14045            *      INTEGER  S0 = S-0, S1 = S-1;                                   *
14046            *      INTEGER  MST = Q-5, LST = Q-4;   << TIME >>                    *
14047            *   INTEGER TRLPTR;                                                   *
14048            *                                                                     *
14049            *      TIME := TIME+100D;                 << FIGURE TICKS >>       *
14050            *      TOS := 0;  TOS := MST;                                         *
14051            *      TOS := 100;                                                    *
14052            *      ASMB( LDIV );                                                  *
14053            *      TOS := LST;                                                    *
```

```
RECORD  C.S.      ************ ALU A ************  ************** ALU B **************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14054           *       TOS := 100;                                                  *
14055           *       ASMB( LDIV,DEL );                                            *
14056           *       DISABLE;                                                     *
14057           *       TRLX := TRLFREELISTP;                                        *
14058           *       IF  =  THEN  SUDDENDEATH(3);                                 *
14059           *       IF NOT(%14 <= TRLX <= TRLNUMENTRIES * TRLENTRYSIZE) THEN     *
14060           *                                                                    *
14061           *          SUDDENDEATH(25);                                          *
14062           *       << if this is last TRL entry, report TABLE'FULL >>           *
14063           *       TRLPTR := TRLX;                                              *
14064           *       TOS := TRLPTRNEXT;                                           *
14065           *       if = then                                                    *
14066           *          MPE'TABLE'FULL(4);  << 4 is timer req. list >>            *
14067           *       TRLFREELISTP := TOS;  << UPDATE NEXT POINTER>>               *
14068           *       TIMEREQ := TRLX&ASR(2);                                      *
14069           *       DO                                                           *
14070           *          BEGIN   << FIND POSITION IN LINE >>                       *
14071           *          TRLPTR := T;                                              *
14072           *          T := TRLPTRNEXT;                                          *
14073           *          TOS := TRLSERVTIME1;                                      *
14074           *          TOS := TRLSERVTIME2;                                      *
14075           *          ASMB( DSUB );                                             *
14076           *          END                                                       *
14077           *       UNTIL  T = 0  OR  S1 < TRL(T+2)  OR  =  AND  LOGICAL(S0) <   *
14078           *                 LOGICAL(TRL(X:=X+1));                              *
14079           *S := TRLPTR;                                                        *
14080           *                                                                    *
14081           *       << BUILD ENTRY >>                                            *
14082           *       IF REQ=0 THEN SUDDENDEATH(26);                               *
14083           *       TIME := TOS;                                                 *
14084           *  TRL(TRLX) := %100000+CODE&LSL(10)+(T/TRLENTRYSIZE);               *
14085           *   TRLPTR := X;                                                     *
14086           *   TRLREQUEST := REQ;                                               *
14087           *   TRLSERVTIME1 := MST;                                             *
14088           *   TRLSERVTIME2 := LST;                                             *
14089           *   TRLPTR := S;                                                     *
14090           *   TRLLINK := TRLX&ASR(2);    <<  CONVERT TO ENTRY # >>             *
14091           *                                                                    *
14092           *       << CHECK FOR TIME ADJUST IN T >>                             *
14093           *       IF  T <> 0  THEN                                             *
14094           *           BEGIN                                                    *
14095           *   TRLPTR := T;                                                     *
14096           *   TOS := TRLSERVTIME1;                                             *
14097           *   TOS := TRLSERVTIME2;                                             *
14098           *           TOS := TOS-TIME;                                         *
14099           *   TRLSERVTIME2 := TOS;                                             *
14100           *   TRLSERVTIME1 := TOS;                                             *
14101           *           END;                                                     *
14102           *                                                                    *
14103           *                                                                    *
14104           *       << IF SIO TIMEOUT, DIT8.((CODE LAND %17):1) := 0 >>          *
14105           *       TOS := CODE;                                                 *
14106           *       TOS.(11:1) := 0;                                             *
14107           *       IF = THEN DEL                                                *
14108           *       ELSE                                                         *
14109           *          BEGIN << SIO TIMEOUT >>                                   *
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14110           *          TOS := F(%1010+REQ);                                    *
14111           *          ASMB(XBX;                                               *
14112           *               TRBC 0,X;                                          *
14113           *               XCH,STAX;);                                        *
14114           *          F(X) := TOS;                                            *
14115           *          END;                                                    *
14116           *                                                                  *
14117           *                                                                  *
14118           *                                                                  *
14119           *    TRLTRACEWORD := %10000 + TRLX;                                *
14120           *                                                                  *
14121           *                                                                  *
14122           *                                                                  *
14123           *END;  << T I M E R E Q >>                                         *
14124           *                                                                  *
14125           *      This routine is not meant to be called by the users, but in *
14126           *    case it is, he/she needs to make sure that SR = 4.            *
14127           *                                                                  *
14128           ************************************************************************
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
14130	0E20	TMRQ	SR	UBA	SUB			POS			ADD					Check if SR = 4
14132	0E21	XXX			JSZ	PULM		UNC			ADD					if SR-4 < 0, go to PULM
14134	0E22		SR	RH	JSBS	XXX		NEG	0064		ADDL					Repeat until SR = 4
14136																UBB <- 100D
14137	0E23			RB	ADD				UBB	RA	LINK		RA			TIME <- TIME + 100D, where TIME is in RB,RA
14139																RA <- LST
14140	0E24		UBA		ADD		RB		UBB		ADD					RB <- MST
14142	0E25			UBA	ADD		SPOA		0064		ADDL		RH			The next 4 lines perform the division
14144																0,MST/100D. RB <- quotient, RG <- rem.
14145	0E26				ADD					RB	REPN				0010	
14146	0E27		UBA	RH	DVSB				UBB		LINK			DCTR	CTR0	
14147	0E28		UBA		ADD	LSR	RG				ADD					
14148	0E29			SP4A	ADD		RB				ADD					
14149	0E2A			RA	ADD		SPOA				ADD					The next 4 lines perform the division
14151																RG,LST/100D. RA <- quotient, the remainder
14152																is ignored.
14153	0E2B			RG	ADD					RA	REPN				0010	
14154	0E2C		UBA	RH	DVSB				UBB		LINK			DCTR	CTR0	
14155	0E2D		UBA		ADD	LSR					ADD					
14156	0E2E				ADD				00F0		ADDL		CTR			CTR <- F0 for LST %12
14158	0E2F			SP4A	ADD		RA			P	INC		P	RONP		RA <- quotient
14160																inc P and store new instr. in NIR
14161	0E30		0200		ADDL				FFE0	REGN	ANDL					UBA <- offset to TRLNUMENTRIES
14163																UBB <- address bits for LST %12
14164	0E31		UBA	UBB	ADD		RF	ROX3	001F	REGN	ANDL		BKX3			RF <- address of TRLNUMENTRIES;read into OPA
14166																BKX3 <- correct bank for LST %12
14167	0E32		UBA		INC						ADD					
14168	0E33			OPA	ASL				UBA		INC			ROX3		UBA <- TRLNUMENTRIES*2
14170																Read TRLFREELISTP into OPB
14171	0E34			UBA	ASL						ADD					UBA <- TRLNUMENTRIES*TRLENTRYSIZE
14173	0E35			UBA	ADD					OPB	JSB	DY03	SP2B		ZERO	SP2B <- TRLFREELISTP; if 0, go to DY03 which
14175																sets up things for SUDDENDEATH(3)
14176	0E36		000B		ADDL				UBA	OPB	JSBS	DY25			NEG	UBA <- %13 for range check later
14178																if TRLFREELISTP > TRLNUMENTRIES*TRLENTRYSIZE
14179																then go to DY25
14180	0E37			RF	ADD				UBA	OPB	JSBS	DY25			POS	UBA <- address of LST %12
14182																if TRLFREELISTP < %14 go to DY25
14183	0E38		0002	UBA	ADDL				UBA	OPB	ADD			ROX3		UBA <- address of TRLFREELISTP
14185																Read TRLLINK into OPB
14186	0E39		UBA		ADD			WRX3	0008		ADDL		SP1B			Set up to write into TRLFREELISTP
14188																SP1B <- initial T = 8
14189	0E3A			SP2B	ASR				03FF	OPB	ANDL		RG			UBA <- TRLX&ASR
14191																RG <- TRLLINK
14192	0E3B			UBA	ASR		RH			UBB	ASL					RH <- TRLX&ASR(2)
14194																UBB <- TRLLINK*2
14195	0E3C			UBB	ASL			DATA	FFF8	Q	ADDL					TRLFREELISTP <- TRLPTRNEXT
14197																UBB <- Q - %10
14198	0E3D			UBB	ADD			WRS		UBA	ADD				NZRO	Set up to write return value at Q-%10
14200																IF TRLPTRNEXT <> 0, SKIP NEXT LINE
14201	0E3E		RH		ADD			DATA			ADD			SF1		Write return value
14203																SET F1 TO GO TO TBFL ON EXIT
14204	0E3F	LUP1		SP1B	ADD		RE				ADD					RE <- TRLPTR
14206	0E40		RF	UBA	ADD			ROX3	0002		ADDL					Read TRL(TRLPTR) into OPA
14208																UBB <- 2
14209	0E41				ADD				UBA	UBB	ADD			ROA3		Read TRLSERVICETIME1 into OPA
14211	0E42		03FF	OPA	ANDL		RG		UBB		INC			ROX3		RG <- TRLLINK

RECORD NO	C S ADDR	LABL	********** ALU A ********** RREG	SREG	FUNC	SFNC	STOR	SPSK	********** ALU B ********** RREG	SREG	FUNC	SFNC	STOR	SPEC	SKIP	COMMENT
14213																Read TRLSERVICETIME2 into OPB
14214	0E43			UBA	ASL						ADD					UBA <- TRLLINK*2
14216	0E44				ADD				UBA		ASL		SP1B			SP1B (T) <- TRLPTRNEXT
14218	0E45		RB	OPA	SUB				RA	OPB	LINK		RA			RB,RA <- TIME - TRLSERVICETIME1,2
14220	0E46			UBA	ADD		RB			UBB	ADD					
14221	0E47			SP1B	JSB	L1		ZERO	0002	SP1B	ADDL					If T = 0 go to L1
14223																UBB <- T+2
14224	0E48		RF	UBB	ADD			ROX3			ADD					Read TRL(T+2) into OPA
14226	0E49				ADD				UBA		INC		ROX3			Read TRL(X:=X+1) into OPB
14228	0E4A				ADD						ADD					
14229	0E4B		RB	OPA	SUB				RA	OPB	LINK					If S-0,S-1 >= TRL(T+2),TRL(X:=X+1) go to LUP1.
14231																
14232	0E4C		UBA		JSB	LUP1		POS			ADD					
14233	0E4D	L1		RC	JSB	DY26		ZERO			ADD					If REQ = 0 go to DY26
14235	0E4E			RD	ADD						REPN				0009	UBA <- CODE
14237																Set up for CODE&ASL(10)
14238	0E4F				ADD				UBA	UBB	ASL		DCTR	CTR0		UBB <- CODE&ASL(10)
14240	0E50				ADD				8000	UBB	ADDL		RG			RG <- %100000 + CODE&ASL(10)
14242	0E51			SP1B	ASR						ADD					UBA <- T/2
14244	0E52		RF	SP2B	ADD			WRX3		UBA	ASR					Set up to write into TRL(TRLX)
14246																UBB <- T/TRLENTRYSIZE
14247	0E53		UBB	RG	ADD			DATA	RE	RF	ADD		ROA3			TRL(TRLX) <- %100000+CODE&ASL(10)+T/TRLENTRY
14249																Read from TRL(TRLPTR) into OPA
14250	0E54		RC		ADD			DATA		SP2B	ASR					TRLREQUEST <- REQ
14252																UBB <- TRLX&ASR
14253	0E55		RB		ADD			DATA		UBB	ASR					TRLSERVICETIME1 <- MST
14255																UBB <- TRLX&ASR(2)
14256	0E56		RA		ADD			DATA	03FF	UBB	ANDL					TRLSERVICETIME2 <- LST
14258																UBB <- TRLX&ASR(2).(6:10)
14259	0E57		FC00	OPA	ANDL					UBB	ADD					UBA <- TRL(TRLPTR).(0:6)
14261																UBB <- TRLX&ASR(2).(6:10)
14262	0E58			SP1B	JSB	LUP2		ZERO	UBA	UBB	IOR		DATA			If T = 0 go to LUP2
14264																TRLLINK <- TRLX&ASR(2).(6:10)
14265	0E59				ADD				0002	SP1B	ADDL					UBB <- TRLPTR + 2
14267	0E5A		RF	UBB	ADD			ROX3			ADD					Read TRLSERVICETIME1 into OPA
14269	0E5B				ADD				UBA		INC		ROX3			Read TRLSERVICETIME2 into OPB
14271	0E5C				ADD						ADD					waiting for the reads to complete
14273	0E5D			OPA	ADD					OPB	ADD					UBA <- TRLSERVICETIME1
14275																UBB <- TRLSERVICETIME2
14276	0E5E		UBA	RB	SUB				UBB	RA	LINK		DATA			TRLSERVICETIME1,2 <- TRLSERVICETIME1,2-TIME
14278	0E5F		UBA		ADD			DATA			ADD					
14279	0E60	LUP2	0010	RD	ANDL						ADD		BKX4			UBA <- CODE.(11:1)
14281																BKX4 <- 0
14282	0E61		0208	RC	ADDL				UBA		JSB	LUP3			ZERO	UBA <- REQ + %1010
14284																If CODE.(11:1) = 0 go to LUP3
14285	0E62		UBA		ADD			ROX4	000F	RD	ANDL					Read abs(REQ+%1010) into OPA
14287																UBB <- CODE.(11:1) := 0
14288	0E63		FFFE		ADDL				000F	UBB	SUBL				NZRO	The next 4 lines will set up the mask to
14290																clear the correct bit in the TRBC 0,X
14291																instruction.
14292	0E64		UBA		ADD				UBB		JSB	NMSK			UNC	The mask is already set up, so go to NMSK
14294	0E65			UBA	ADD					UBB	ADD					
14295	0E66	MSK	UBA		ADD				UBB		CAD					
14296	0E67		UBA		CSL				UBB		JSB	MSK			NZRO	
14297	0E68		UBA		ADD				UBB		ADD					
14298	0E69		UBA		ADD				UBB		ADD					

RECORD NO	C.S. ADDR	********** ALU A ********** LABL	RREG	SREG	FUNC	SFNC	STOR	SPSK	********** ALU B ********** RREG	SREG	FUNC	SFNC	STOR	SPEC	SKIP	COMMENT
14299	0E6A	NMSK	UBA	OPA	AND			DATA			ADD					Clear bit, write it back to abs(REQ+%1010)
14301	0E6B	LUP3	0009	RF	ADDL				1000	SP2B	ADDL		RH			UBA <- address of TRLTRACEWORD
14303																UBB <- TRLX + %10000
14304	0E6C		UBA		ADD			WRX3			ADD					Set up to write into TRLTRACEWORD
14306	0E6D		RH		ADD			DATA			JSB	TBFL			F1	TRLTRACEWORD <- TRLX + %10000
14308																IF F1, GO TO TBFL
14309	0E6E				ADD						ADD			CCA	NEXT	SET CC TO CCE AND EXIT
14311	0E6F	DY03			ADD				0003		ADDL		RA			Store 3 in TOS
14313	0E70				ADD				UBB		ADD			CCA	NEXT	Set CC to CCG and exit
14315	0E71	DY25			ADD				0019		ADDL		RA			Store 25 in TOS
14317	0E72				ADD				UBB		ADD			CCA	NEXT	Set CC to CCG and exit
14319	0E73	DY26			ADD				001A		ADDL		RA			Store 26 in TOS
14321	0E74				ADD				UBB		ADD			CCA	NEXT	Set CC to CCG and exit
14323	0E75	TBFL	FFFF		ADDL				0004		ADDL		RA			Store 4 in TOS
14325	0E76				ADD				UBA		ADD			CCA	NEXT	Set CC to CCL and exit

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14328           ************************************************************************
14329           *                                                                      *
14330           *      MSTA - REPLACEMENT FOR THE MPE PROCEDURE MMSTAT                 *
14331           *                                                                      *
14332           *           Macro-assembly code: %020104,%000035  (doubleword op)      *
14333           *                available in version CX2603A and beyond               *
14334           *                                                                      *
14335           *   This routine replaces the MPE procedure MMSTAT, which logs         *
14336           *   low-level system events into a 128-entry circular queue.  Each     *
14337           *   event consists of the current process number, an event number,     *
14338           *   and six parameters.  Every time the queue wraps around, the        *
14339           *   time is saved, along with the previous wrap time (from a call      *
14340           *   to TIMER, also microcoded into this routine).                      *
14341           *                                                                      *
14342           *   The routine assumes itself to be inside a procedure call, so       *
14343           *   all the parameters passed to it are in the Q- area (Q-4 thru       *
14344           *   Q-%12).  The monitor buffer is located by a pointer in LST %11,    *
14345           *   and the current index is stored in SysGlob %34.                    *
14346           *                                                                      *
14347           *   This routine, as with some of the others in this group, uses       *
14348           *   the condition codes to return status of the instruction.  If       *
14349           *   it was able to handle the call, it returns with the condition      *
14350           *   code set to CCE.  If, however, it could not (as in when tape       *
14351           *   logging is enabled [only in debugging], or if there are not        *
14352           *   enough TOS registers [not possible in the way it is currently      *
14353           *   called]), then it returns with a condition code of CCG. (whew!)    *
14354           *                                                                      *
14355           *   If the routine is called with a negative event (those that only    *
14356           *   get put into the tape buffer), and tape logging is disabled        *
14357           *   (the normal case), then the routine just returns - finished.       *
14358           *   If, however, the event is positive, the event number along with    *
14359           *   the current process number and the six parameters are placed       *
14360           *   into the circular queue.  When a wrap occurs, the time is saved    *
14361           *   as well as the previous time in entry 0 (each is 2 words long).    *
14362           *                                                                      *
14363           *  +---------------********** WARNING **********---------------+      *
14364           *  |    IF ANY OF THE OFFSETS OR BIT POSITIONS USED BY THIS CODE |      *
14365           *  |    SHOULD CHANGE IN FUTURE VERSIONS OF MPE, THIS CODE MUST  |      *
14366           *  |    BE MODIFIED OR IT WILL CEASE TO WORK PROPERLY.           |      *
14367           *  +-------------------------------------------------------------+      *
14368           *                                                                      *
14369           *   The SPL source for this routine, from the module KERNELC, is:      *
14370           *                                                                      *
14371           *<<************************************************************    *
14372           *                                                                      *
14373           *   COMMENT  THIS FUNCTION IS CALLED TO MONITOR AND/OR COLLECT         *
14374           *            PERFORMANCE MEASUREMENT DATA;                             *
14375           *   PROCEDURE MMSTAT'(EVENT,P1,P2,P3,P4,P5,P6);                        *
14376           *   VALUE EVENT,P1,P2,P3,P4,P5,P6;                                     *
14377           *   INTEGER EVENT,P1,P2,P3,P4,P5,P6;                                   *
14378           *   OPTION          PRIVILEGED,UNCALLABLE;                             *
14379           *                                                                      *
14380           *                                                                      *
14381           *   BEGIN <<MMSTAT>>                                                   *
14382           *                                                                      *
14383           *   INTEGER                                                            *
```

```
 RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14384           *       PCBPT,                                                        *
14385           *       OFFSET;                                                       *
14386           *    DOUBLE                                                           *
14387           *       BASE,                                                         *
14388           *       SAVE'DB;                                                      *
14389           *                                                                     *
14390           *    INTRINSIC         TIMER;                    <<TIME FUNCTION>>    *
14391           *    $PAGE "SYSTEM TABLE MANAGEMENT : GET SYSTAB ENTRY"                *
14392           *                                                                      *
14393           *                                                                      *
14394           *                                                                      *
14395           *    DOUBLE SUBROUTINE CONVERT'POINTER'TO'DOUBLE'ADDRESS(PTR);         *
14396           *    VALUE PTR;                                                        *
14397           *    LOGICAL PTR;                                                      *
14398           *    BEGIN                                                             *
14399           *                                                                      *
14400           *    TOS := ABS(SYSBASE + PTR).(11:5); << extract bank >>              *
14401           *    TOS := ABS(X) LAND %177740; << extract address >>                 *
14402           *    BASE := TOS;                                                      *
14403           *    CONVERT'POINTER'TO'DOUBLE'ADDRESS := BASE;                        *
14404           *                                                                      *
14405           *    END; << subroutine CONVERT'POINTER'TO'DOUBLE'ADDRESS >>           *
14406           *                                                                      *
14407           *    << IF A PROCESS IS EXECUTING, GET ITS PIN >>                      *
14408           *                                                                      *
14409           *    PCBPT := CURPRC;                                                  *
14410           *                                                                      *
14411           *                                                                      *
14412           *    << IF THE EVENT NUMBER IS POSITIVE THEN >>                        *
14413           *    <<                                      >>                        *
14414           *    <<       CHECK EVENT VALIDITY           >>                        *
14415           *    <<       DISABLE INTERRUPTS             >>                        *
14416           *    <<       XCHD TO SYSDB                  >>                        *
14417           *    <<       LOG THE EVENT TO MONBUF        >>                        *
14418           *    <<                                      >>                        *
14419           *    << IF THE EVENT NUMBER IS NEGATIVE THEN >>                        *
14420           *    <<                                      >>                        *
14421           *    <<       MAKE IT POSITIVE               >>                        *
14422           *    <<       CHECK ITS VALIDITY             >>                        *
14423           *    <<       DISABLE INTERRUPTS             >>                        *
14424           *    <<       XCHD TO SYSDB                  >>                        *
14425           *                                                                      *
14426           *                                                                      *
14427           *                                                                      *
14428           *                                                                      *
14429           *    <<  Turn off traps >>                                             *
14430           *    PUSH(STATUS);                                                     *
14431           *    TOS.(2:1) := 0;                                                   *
14432           *    SET(STATUS);                                                      *
14433           *                                                                      *
14434           *    TOS := EVENT;                                                     *
14435           *                                                                      *
14436           *    IF >= THEN                                                        *
14437           *                                                                      *
14438           *      BEGIN <<POSITIVE EVENT>>                                        *
14439           *                                                                      *
```

```
RECORD   C.S.      ************ ALU A ************  ************** ALU B ***************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14440            *      IF EVENT > MAXEVENTNUMBER THEN                                *
14441            *         RETURN;                                                    *
14442            *                                                                    *
14443            *      DISABLE;                                                      *
14444            *      push(z); push(s); assemble(sub);                              *
14445            *                                                                    *
14446            *      if TOS<34 then return; <<MMstat cant stand a stack overflow>> *
14447            *                                                                    *
14448            *      TOS := SYSBASE D;                                             *
14449            *                                                                    *
14450            *      ASSEMBLE(XCHD);                                               *
14451            *      SAVE'DB := TOS;                                               *
14452            *                                                                    *
14453            *                                                                    *
14454            *                                                                    *
14455            *    << Save procinx, event, and parms into MONBUF >>                *
14456            *                                                                    *
14457            *      SMONBUF(SMONDIX) := PCBPT;                                    *
14458            *      SMONBUF(X := X + 1) := EVENT;                                 *
14459            *      SMONBUF(X:=X+1)  := P1;                                       *
14460            *      SMONBUF(X:=X+1)  := P2;                                       *
14461            *      SMONBUF(X:=X+1)  := P3;                                       *
14462            *      SMONBUF(X := X + 1) := P4;                                    *
14463            *      SMONBUF(X := X + 1) := P5;                                    *
14464            *      SMONBUF(X := X + 1) := P6;                                    *
14465            *                                                                    *
14466            *                                                                    *
14467            *                                                                    *
14468            *                                                                    *
14469            *    << IF THE END OF THE BUFFER HAS BEEN REACHED, >>                *
14470            *    << SET THE BUFFER INDEX TO THE START OF THE   >>                *
14471            *    << BUFFER AND RESET THE TIME STAMPS IN THE    >>                *
14472            *    << FIRST ENTRY.  THIS ENTRY CONTAINS TWO TIME >>                *
14473            *    << TIME STAMPS INDICATING THE LAST TWO TIMES  >>                *
14474            *    << WRAP AROUND HAS OCCURED.  TIME STAMPS ARE  >>                *
14475            *    << TWO WORDS IN LENGTH.                       >>                *
14476            *                                                                    *
14477            *                                                                    *
14478            *       IF (SMONDIX := SMONDIX + SMON'ENTRY'SIZE) >=                 *
14479            *           SMONBUFSIZE THEN                                         *
14480            *                                                                    *
14481            *         BEGIN <<MONBUF WRAP AROUND>>                               *
14482            *                                                                    *
14483            *         SMONDIX := SMON'ENTRY'SIZE;                                *
14484            *         TOS := TIMER;              <<GET NEW TIME STAMP>>          *
14485            *         TOS := SMONBUF(0);         <<GET OLD TIME STAMP>>          *
14486            *         TOS := SMONBUF(X:=X+1);                                    *
14487            *         SMONBUF(X:=X+2) := TOS;  <<SAVE OLD TIME STAMP>>           *
14488            *         SMONBUF(X:=X-1) := TOS;                                    *
14489            *         SMONBUF(X:=X-1) := TOS;  <<SAVE NEW TIME STAMP>>           *
14490            *         SMONBUF(X:=X-1) := TOS;                                    *
14491            *                                                                    *
14492            *         END; <<MONBUF WRAP AROUND>>                                *
14493            *                                                                    *
14494            *      END <<POSITIVE EVENT>>                                        *
14495            *                                                                    *
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14496           *   ELSE                                                          *
14497           *                                                                 *
14498           *     BEGIN <<NEGATIVE EVENT>>                                    *
14499           *                                                                 *
14500           *     TOS := -EVENT;                                              *
14501           *                                                                 *
14502           *     IF (EVENT := TOS) > MAXEVENTNUMBER THEN                     *
14503           *        RETURN;                                                  *
14504           *                                                                 *
14505           *     DISABLE;                                                    *
14506           *     TOS := SYSBASE D;                                           *
14507           *     ASSEMBLE(XCHD);                                             *
14508           *     SAVE'DB := TOS;                                             *
14509           *                                                                 *
14510           *     END; <<NEGATIVE EVENT>>                                     *
14511           *                                                                 *
14512           *                                                                 *
14513           *   << IF A MONITOR RUN IS REQUESTED THEN LOG THE EVENT >>        *
14514           *   << TO THE MEASBUF DOUBLE BUFFER SET AND CALL MEASIO >>        *
14515           *   << WHEN A BUFFER IS FILLED.                          >>       *
14516           *                                                                 *
14517           *                                                                 *
14518           *   IF (TOS:=MEASFLAG) THEN                                       *
14519           *                                                                 *
14520           *     BEGIN <<MEASUREMENT ENABLED>>                               *
14521           *                                                                 *
14522           *                                                                 *
14523           *   << COMPUTE THE GROUP ID (MODULO 10) AND TEST ITS >>           *
14524           *   << CORRESPONDING MEASUREMENT ENABLE FLAG.        >>           *
14525           *                                                                 *
14526           *                                                                 *
14527           *     TOS := EVENT/10;                                            *
14528           *     ASSEMBLE(TRBC 11);                                          *
14529           *                                                                 *
14530           *     TOS := IF = THEN MEAS'MSK0 ELSE MEAS'MSK1;                  *
14531           *                                                                 *
14532           *     ASSEMBLE(STBX; TBC 0,X; DDEL);                              *
14533           *                                                                 *
14534           *                                                                 *
14535           *                                                                 *
14536           *   << IF THE EVENT IS ENABLED FOR MONTIORING, DUMP IT   >>       *
14537           *   << TO THE CURRENT MEASUREMENT BUFFER.  IF THE BUFFER >>       *
14538           *   << IS FILLED, CALL "MEASIO" TO WRITE THE BUFFER TO   >>       *
14539           *   << TAPE AND SWITCH TO THE OTHER BUFFER.              >>       *
14540           *                                                                 *
14541           *                                                                 *
14542           *                                                                 *
14543           *     IF <> THEN                              <<TEST ENABLE FLAG>>*
14544           *                                                                 *
14545           *        BEGIN <<EVENT ENABLED>>                                  *
14546           *                                                                 *
14547           *        ASSEMBLE(TBC 14);                    <<TEST GROUP FLAG>>*
14548           *        OFFSET := IF = THEN 0 ELSE MEASBUFSIZE;                  *
14549           *                                                                 *
14550           *        TOS := TIMER;                                            *
14551           *        ASSEMBLE(XCH);                                           *
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14552          *        MEASBUF(INTEGER(MEAS'IDX) + OFFSET) := TOS;                  *
14553          *        MEASBUF(X:=X+1)   := TOS;                                    *
14554          *        MEASBUF(X := X + 1) := PCBPT;                                *
14555          *        MEASBUF(X:=X+1)   := EVENT;                                  *
14556          *        MEASBUF(X:=X+1)   := P1;                                     *
14557          *        MEASBUF(X:=X+1)   := P2;                                     *
14558          *        MEASBUF(X:=X+1)   := P3;                                     *
14559          *        MEASBUF(X := X + 1) := P4;                                   *
14560          *        MEASBUF(X := X + 1) := P5;                                   *
14561          *        MEASBUF(X := X + 1) := P6;                                   *
14562          *                                                                     *
14563          *        IF (MEAS'IDX := MEAS'IDX + MEAS'ENTRY'SIZE) >=               *
14564          *            MEASBUFSIZE THEN                                         *
14565          *                                                                     *
14566          *           BEGIN <<WRITE BUFFER>>                                    *
14567          *                                                                     *
14568          *           MEAS'IDX := 0;                                            *
14569          *           ASSEMBLE(TCBC 14; DUP);     <<SWITCH BUFFER FLAG>>        *
14570          *           MEASFLAG := TOS;            <<SAVE IN CASE INTERRUPTED>>  *
14571          *           TOS := 0;                   <<RESULT>>                    *
14572          *           TOS := MEAS'LDEV;           <<TAPE LDEV>>                 *
14573          *           TOS := 1;                   <<MEASIO WRITE REQUEST>>      *
14574          *           TOS := CONVERT'POINTER'TO'DOUBLE'ADDRESS(MEASBUFPTR);     *
14575          *           TOS := TOS + OFFSET;                                      *
14576          *           TOS := MEASBUFSIZE;         <<MEASBUF SIZE>>              *
14577          *           TOS := MEAS'PLAB;           <<MEASIO PLABEL>>             *
14578          *                                                                     *
14579          *           TOS := SYSBASE.D;                                         *
14580          *           ASSEMBLE(XCHD;DDEL);                                      *
14581          *                                                                     *
14582          *           ASSEMBLE(PCAL 0);           <<CALL MEASIO>>               *
14583          *                                                                     *
14584          *                                                                     *
14585          *                                                                     *
14586          *   << ON RETURN FROM "MEASIO", CHECK THE TAPE STATUS  >>             *
14587          *   << AND MARK THE APPROPRIATE MEASFLAG BIT:          >>             *
14588          *   <<          MEASFLAG(12:1) - TAPE ERROR            >>             *
14589          *   <<          MEASFLAG(13:1) - END OF TAPE           >>             *
14590          *   <<          MEASFLAG(14:1) - BUFFER SELECT         >>             *
14591          *   <<          MEASFLAG(15:1) - MEASIO ENABLE         >>             *
14592          *                                                                     *
14593          *           DEL;                    <<DELETE TAPE STATUS WORD>>       *
14594          *                                                                     *
14595          *           IF <> THEN              <<TEST TAPE STATUS>>              *
14596          *                                                                     *
14597          *              BEGIN <<TAPE STATUS>>                                  *
14598          *                                                                     *
14599          *              IF > THEN TOS.(13:1) := 1    <<END OF TAPE>>           *
14600          *                   ELSE TOS.(12:1) := 1;   <<TAPE ERROR>>            *
14601          *                                                                     *
14602          *              TOS.(15:1) := 0;             <<CLEAR MEASIO ENABLE>>  *
14603          *                                                                     *
14604          *              END; <<TAPE STATUS>>                                   *
14605          *                                                                     *
14606          *           END; <<WRITE BUFFER>>                                     *
14607          *                                                                     *
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14608            *        END; <<EVENT ENABLED>>                                    *
14609            *                                                                  *
14610            *     MEASFLAG := TOS;                 <<RESTORE MEASFLAG>>        *
14611            *                                                                  *
14612            *     END <<MEASUREMENT ENABLED>>                                  *
14613            *                                                                  *
14614            *                                                                  *
14615            *   ELSE                                                           *
14616            *                                                                  *
14617            *     DEL;                             <<DELETE MEASFLAG>>         *
14618            *                                                                  *
14619            *   TOS := SAVE'DB;                                                *
14620            *   ASSEMBLE(XCHD);                                                *
14621            *   DDEL;                                                          *
14622            *                                                                  *
14623            *   END; <<MMSTAT>>                                                *
14624            *                                                                  *
14625            ************************************************************************
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14627   0E77  MSTA           JSB  MMXT      SRG2        P    INC       P    RONP       Make sure that we can use the last five TOS
14629                                                                                  Increment and read from P - doubleword instr
14630   0E78        02B7       ADDL                 000A       ADDL                    Get address of MEASFLAGS to see if tape on
14632                                                                                  Get negative offset of EVENT for read
14633   0E79             UBA   ADD             ROX3 UBB  Q    RSUB           ROS      Read MEASFLAGS for check of tape logging
14635                                                                                  Read EVENT (Q-%12) for use later
14636   0E7A        021C       ADDL      RH         00EF       ADDL      CTR          Get address of SMON INDEX (SysGlob %34)
14638                                                                                  Get pointer to LST %11 (SMON BASE)
14639   0E7B             OPA   JSB  MMXT      NZRO 0009       ADDL      RG           Leave if MEASFLAGS is on (tape logging)
14641                                                                                  Get pointer to first parameter for later
14642   0E7C                   ADD             CCA        OPB  JSB  MMXT            NEG Set condition code to CCE for good returns
14644                                                                                  Leave if EVENT is negative - all done!
14645   0E7D             UBB   ADD       SPOA       0004       ADDL      SP1B         Store EVENT in SPOA for safekeeping
14647                                                                                  Get address of CURRENT PROCESS for later
14648   0E7E             RH    ADD             ROX3 FFE0 REGN ANDL                    Read SMON INDEX (SysGlob %34)
14650                                                                                  Mask offset of LST %11 (SMON BASE)
14651   0E7F        200  UBB   ADDL      RD         001F REGN ANDL      BKX4         Compute partial address of next queue entry
14653                                                                                  Set up bank register for queue address
14654   0E80             OPA   ADD                          SP1B ADD           ROA3   Get SMON INDEX for complete address compute
14656                                                                                  Read CURRENT PROCESS NUMBER for entry
14657   0E81             UBA   RD    ADD       WRX4 0008 UBA  ADDL      SP1B         Set up write address for queue entry
14659                                                                                  Store new SMON INDEX in SP1B temporarily
14660   0E82             OPA   ADD             DATA RG   Q    RSUB           ROAS     Write CURRENT PROCESS NUMBER to entry
14662                                                                                  Read first parameter for entry
14663   0E83        SPOA       ADD             DATA       UBB  INC            ROAS     Write event to entry
14665                                                                                  Read second parameter for entry
14666   0E84             OPA   ADD             DATA       UBB  INC            ROAS     Write first parameter to entry
14668                                                                                  Read third parameter for entry
14669   0E85             OPA   ADD             DATA       UBB  INC            ROAS     Write second parameter to entry
14671                                                                                  Read fourth parameter for entry
14672   0E86             OPA   ADD             DATA       UBB  INC            ROAS     Write third parameter to entry
14674                                                                                  Read fifth parameter for entry
14675   0E87             OPA   ADD             DATA       UBB  INC            ROAS     Write fourth parameter to entry
14677                                                                                  Read sixth parameter for entry
14678   0E88             OPA   ADD             DATA 3FF        ADDL                    Write fifth parameter to entry
14680                                                                                  Set up for queue wrap check
14681   0E89             OPA   ADD             DATA UBB  SP1B JSBS MMDN           POS  Write sixth parameter to entry
14683                                                                                  Check for queue wrap (INDEX >= %2000)
14684           *
14685           ************************************************
14686           *                                              *
14687           *     The following code duplicates the TIMER  *
14688           *     routine (back at %E00) except its inline *
14689           *     The result is final in line %E9E, and is *
14690           *     written out to queue entry 0.            *
14691           *                                              *
14692           ************************************************
14693           *
14694   0E8A        14         ADDL                 00F0       ADDL      CTR          Set up for reads of LR and TR
14696                                                                                  Set up for LST %12 (TRL BASE)
14697   0E8B        UBA        INC             ROX3       BKX4 ADD       BKX5         Read TR (Time since last software timer int)
14699                                                                                  Save queue bank register for later
14700   0E8C        RREG       CAD             ROB3 001F REGN ANDL      BKX4         Read LR (System clock limit register)
14702                                                                                  Set up bank register for LST %12 (TRL BASE)
14703   0E8D             OPA   ADD                    FFE0 REGN ANDL                  Get TR from memory
14705                                                                                  Get offset of LST %12 (TRL BASE)
```

RECORD NO	C S. ADDR	LABL	********** ALU A ********** RREG	SREG	FUNC	SFNC	STOR	SPSK	********** ALU B ********** RREG	SREG	FUNC	SFNC	STOR	SPEC	SKIP	COMMENT
14706	0E8E		0205	UBB	ADDL				UBA	OPB	ADD					Compute offset of TRLDTIME1
14708																Partial SYS_CLK_CTR = TR + LR
14709	0E8F			UBA	ADD			ROX4	UBB	XR31	ADD		SP3B			Read TRLDTIME1 into OPA
14711																Full SYS_CLK_CTR = TR + LR + CR (from XR31)
14712	0E90		FFFE	UBA	ADDL						UBA	INC			ROX4	Compute offset of TRLNUMDAYS
14714																Read TRLDTIME2 into OPB
14715	0E91		2E00		ADDL		SP4A				UBA	ADD			ROX4	LSW of MS_PER_DAY; shifted right one bit
14717																Read TRLNUMDAYS into OPB
14718	0E92			OPA	ADD		SP0A		SP3B	OPB	LINK		RG			Perform doubleword add, where NUM_TICKS =
14720																SP0A,RG = 0,SYS_CLK_CTR+TRLDTIME1,TRLDTIME2
14721	0E93			SP4A	ADD	LSL	SP4A		0526		ADDL		SP3B			Get LSW of MS_PER_DAY (!5C00) into SP4A
14723																Get MSW of MS_PER_DAY into SP3B
14724	0E94		SP0A	UBB	SUB				RG	UBA	LINK		RF			Perform doubleword subtract, where Z =
14726																UBA,RF = NUM_TICKS - MS_PER_DAY
14727	0E95			UBA	JSB	MMTM		NEG		OPB	ADD		RF			If Z < 0, less than a day has passed - skip
14729																Put TRLNUMDAYS into RF
14730	0E96			UBA	ADD		SP0A			RF	ADD		RG			Set NUM_TICKS = Z so under a day's ticks
14732																for both words involved (SP0A,RG)
14733	0E97				ADD				FFE9	OPB	ADDL		RF		ZERO	NUM_OF_DAYS= TRLNUMDAYS - 23, skip next line
14735																if NUM_OF_DAYS is zero (0)
14736	0E98				ADD					OPB	INC		RF			Otherwise (TRLNUMDAYS-23 <> 0) increment it,
14738																since we have subtracted a day's ticks
14739	0E99	MMTM			ADD		SP0A			BKX5	ADD		BKX4			Clear out SP0A for the upcoming multiply
14741																Restore the queue's bank register
14742	0E9A		SP0A		ADD		RE		RD		ADD				ROX4	Put MSW of NUM_TICKS into RE for safekeeping
14744																Read old time stamp MSW from entry0(0)
14745	0E9B				ADD						REPN				001F	Set up for multiply - 32 bits wide
14747																using RREGA,RREGB * SP0A,SP3B,SP4A
14748	0E9C			UBA	MPAD				RF	UBB	LINK				DCTR CTR0	Multiply NUM_OF_DAYS * MS_PER_DAY to giving
14750																total number of ms since load (mod 24 days)
14751	0E9D			UBA	ADD					UBB	ADD					Need this line for the multiply to work
14753																correctly - pipeline effects
14754	0E9E		SP0A	RE	ADD				SP3B	RG	LINK		SP2B			Perform doubleword add, where RESULT =
14756																UBA,SP2B = (NUM_OF_DAYS*MS_PER_DAY)+NUM_TICK
14757	0E9F			RD	INC			ROX4		UBA	ADD				DATA	Read old time stamp (LSW) from entry0(1)
14759																Write new time stamp (MSW) into entry0(0)
14760	0EA0			SP2B	ADD			DATA		OPB	ADD					Write new time stamp (LSW) into entry0(1)
14762																Get old time stamp (MSW) from OPB
14763	0EA1			UBB	ADD			DATA			ADD					Write old time stamp (MSW) into entry0(2)
14765																- this moves the old stamp up so we have 2
14766	0EA2			OPA	ADD			DATA	0008		ADDL		SP1B			Write old time stamp (LSW) into entry0(3)
14768																Set SMON INDEX to 8 (entry 1 - we wrapped)
14769	0EA3	MMDN		RH	ADD			WRX3			ADD					Set up address of SMON INDEX for write
14771	0EA4			SP1B	ADD			DATA			ADD					Write SMON INDEX back (possibly wrapped)
14773	0EA5	MMXT			ADD						ADD				NEXT	Return to caller!!!
14775	0EA6				ADD						ADD					NOP to keep system happy
14777	0EA7				ADD						ADD					NOP to keep system happy

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14782          %0F00
14783          ********************************************************************
14784          *          RSW - READ SWITCH REGISTER INSTRUCTION                  *
14785          ********************************************************************
14786          *
14787    0F00  RSW            JSZ  PSHM      SR7        SP4A JSL  IOLT            ODD   Empty one TOS; JMP if I/O instruction
14789    0F01                 ADD             EPSH  00B2        ADDL      CTR           EPSH; SET CTR FOR SWCH REGISTER
14791    0F02                 ADD                               ADD                     Wait around for CTR
14793    0F03                 ADD                         REGN  ADD       RH    CCA  NEXT  (S):=SWITCH REGISTER 0; NEXT INSTRUCTION
14795    0F04                 ADD                               ADD
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL  RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14797           ***********************************************************************
14798           *                                                                     *
14799           *     WMST    this routine resets any extra IOAs that may be          *
14800           *             installed on the system.                                *
14801           *                                                                     *
14802           ***********************************************************************
14803           *
14804   0F05    WMST 0C02       ADDL                    1000      ADDL      XR1             UBA := go BUSY cmd; XRB1 := write command
14806   0F06         0054       ADDL      XR0               UBA   ADD             BUSC      XRA0 := enable masks cmd (will be swapped);
14808                                                                                        Go BUSY
14809   0F07    WMS1       RF   ADD                                  JSL   L2PF BKX5      UNC   UBA.(11:2) := IMB #;
14811                                                                                          BKX5 := 0, jump to get physical IOA #
14812   0F08               UBA  JSB  *+3        NEG   B000      ADDL      SP3B              Don't reset if IOA not present;
14814                                                                                          SP3B := IOCL command
14815   0F09                    ADD                                 XR1   JSL   IOMS SP3B      UNC   ; SP3B := write IMBI reg# 0, flush IOA
14817   0F0A               XR0  ADD  SWAB RG                            JSL   IOMS            UNC   RG := enable masks command, enable masks
14819   0F0B         0080  RF   ADDL      RF          0180  RF    SUBL                        RF := next IOA #; UBB = 0 if done
14821   0F0C               XR15 ADD                                 UBB   JSB   WMS1            NZRO  UBA := load or dump?  Jump back if not done
14823   0F0D                    JSZ  CLDE       SF1         UBA   JSL   DUMP              EVEN  Jump to dump else jump to load, set F1 to
14825                                                                                          inhibit checksum & module mapping if load
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
14827	0F0E	CKCH	E020		ADDL		XR13			RG	JSL	CPEX			ZERO	ERROR CODE; #CPEX IF NO SPOLL2 RESPONSE
14829	0F0F		RG	RG	ADD			NEG	0900		ADDL		XR24			; REG# 9
14831	0F10				ADD			RSB	4008		ADDL		SP3B			
14832	0F11				ADD						JSL	IOR	BKX5		ZERO	
14833	0F12				ADD		XR14	F1HB		RG	ADD	RRZ	BKX5	CF5B		
14834	0F13			UBA	ADD	LSR	XR12				JSL	PAUL			UNC	

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14836           ********************************************************************
14837           *                                                                  *
14838           *    ADJS - ADJuSt the number of TOS registers                     *
14839           *                                                                  *
14840           *    1.  If SR agrees with the number of TOS registers needed then *
14841           *        return.                                                   *
14842           *    2.  If SR is greater than the number of TOS registers needed  *
14843           *        then write the excess TOS registers to memory and update  *
14844           *        SR and SM.                                                *
14845           *    3.  If SR is less than the number of TOS registers needed then*
14846           *        fetch the remainder of the needed TOS registers from      *
14847           *        memory and update SR and SM.                              *
14848           *                                                                  *
14849           ********************************************************************
14850           *
14851   0F14    ADJS SR   RH   SUB           NZRO  SR   RH   SUB                ZERO  Done if SR = n, otherwise adjust # of TOS
14853   0F15         SR   RH   ADD           RSB                ADD        PSHR       Return or save return address
14855   0F16         SR   RH   JSBS ADJ1     NCRY               ADD                   Jump if SR < n
14857   0F17              SM   INC           WRS        SM      INC        SM   DCSR  Write @ SM+1; Increment SM, decrement SR
14859   0F18              QDWN ADD           DATA  Z    SREG  JSZC STO1          NEG   Write QDWN; Stack overflow if SM > Z
14861   0F19                   JSB  ADJS      UNC                ADD        POPR       Do again; Restore return address
14863   0F1A    ADJ1      SM   ADD           ROBS  FFFF SM     ADDL       SM         Read @ SM; Decrement SM
14865   0F1B         UBA  Q    JSZC STU1      NCRY  SR           ADD        CTR  INSR  Stack underflow if SM < Q; Increment SR
14867   0F1C                   JSB  ADJS      UNC          OPB   ADD        REGN POPR  Do again; Store (SM) into TOS register,
14869                                                                                    restore return address
14870   0F1D                   ADD                               ADD
```

```
 RECORD   C.S.          ********** ALU A **********  ************** ALU B ***************
   NO     ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14872            ****************************************************************
14873            *
14874            *    Optional Instruction Decoding  (Group 7)
14875            *
14876            *    This module decodes the Group 7 optional instructions.  At
14877            *    this time this only includes the instructions DDIV and DMUL
14878            *    Therefore, a jump table approach is not used.  CIR.(12:4)
14879            *    is verified to be of the form 100x and then the jump is mad
14880            *    to the instruction code.  Otherwise the unimplemented
14881            *    instruction trap is taken.  The SR preadjust is set to zero
14882            *    because of those unimplemented instructions.
14883            *
14884            ****************************************************************
14885            *
14886            $LUT INSTR=DMUL/DDIV:0 010 000 101 11x xxx, ENTRY=GRP7
14887            *
14888     0F1E   GRP7 000F CIR  ANDL              0008      ADDL                    UBA := CIR.(12:4); UBB := mask for bit 12
14890     0F1F        UBA  UBB  AND        NZRO  0006 UBA  ANDL            ZERO   Skip if bit 12 set; Skip if CIR.(13:2) = 0
14892     0F20                  JSZ  TRP       UNC         JSZ  TRP        UNC    Unimplemented instruction trap
14894     0F21        FFFD      ADDL      SP4A              JSL  DMDV       UNC    SP4A := -3 (for SR preadjust); JSB
14896     0F22                  ADD                         ADD                     NOP to keep system happy
14898     0F23                  ADD                         ADD                     NOP to keep system happy
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14901           ********************************************************************
14902           *                                                                  *
14903           *         The entry point decodes the single word opcodes in the   *
14904           *    range %020460/020476.  The opcode %020477 denotes a double    *
14905           *    word opcode.  Flag 5 is turned on if the user is not in       *
14906           *    split stack mode, off if the user is in split stack mode.     *
14907           *    Flag 1 is turned on.  Flag 2 is set to the least significant  *
14908           *    bit of the opcode for SDEC and PB/DB testing.  BKX3 is        *
14909           *    zeroed for Bank 0 addressing with the CST in XBR, PARC, ENDP, *
14910           *    and code segment set up.  The table is indexed by pairs       *
14911           *    since most instructions occupy an even-odd pair of opcodes.   *
14912           *    Instructions which occupy only a single word opcode are       *
14913           *    distinguished by Flag 2.                                      *
14914           *                                                                  *
14915           ********************************************************************
14916           *
14917           $LUT INSTR=COBL:0 010 000 100 11,ENTRY=COBL,DSPL=4
14918           *
14921           %1000
14922           *
14923    1000   COBL       DSPL ADD  LSR            COBT      ADDL LBL                 OPCODE.(12:4); TABLE ADDRESS
14925    1001          UBA  UBB  ADD                 WRX3   DB  ADD                      YRA:=TABLE START+OFFSET;
14927    1002               ADD                    NFSS  UBB  DL  SUB            CF1  NCRY  SKIP IF NOT SPLIT BANKS OR DB < DL
14929    1003               ADD                    CF1   Z    DB  SUB            CTF1       SPLIT STACK IF SPLIT BANK OR DB<DL OR DB>Z
14931    1004               ADD                    RAR            ADD       BKX3 CF5B NF1   RAR:=YRA; F5:=SPLIT STACK; SKIP IF SO
14933    1005          DSPL CSR                    HBF2           ADD            SF5B       F2:=OPCODE.(15:1);NOT SPLIT STACK MODE
14935    1006               CIR  ADD        SP4A              OPB  ADD       SP1B SF1   RSB  RANDOM INDEXED BRANCH; SAVE SECOND WORD OP.
14937    1007   COBT             ADD                                JSL  ALGN             UNC  OPCODE BRANCH TABLE; ALGN (%020460/1)
14939    1008                     ADD                                JSL  ABSN             UNC  ABSN (%020462/3)
14941    1009                     ADD                                JSZ  TRP              UNC  UNIMPLEMENTED INSTRUCTION (%020464/5)
14943    100A                     ADD                                JSZ  TRP              UNC  UNIMPLEMENTED INSTRUCTION (%020466/7)
14945    100B                     ADD                                JSL  EDIT             UNC  EDIT (%020470/1)
14947    100C                     ADD                                JSL  CMPS             UNC  CMPS (%020472/3)
14949    100D                     JSB  XBR            NF2           JSB  PARC             F2   XBR (%020474); PARC (%020475)
14951    100E                     JSB  ENDP           NF2           ADD                       ENDP (%020476);
```

```
 RECORD   C.S.        *********** ALU A ***********   ************** ALU B **************
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

14954           ******************************************************************
14955           *                                                                *
14956           *         If the single word opcode is %020477, then the second  *
14957           *    word of the opcode is decoded here.  P is incremented for   *
14958           *    the second word of the opcode.  Second word opcodes greater *
14959           *    than %51 are invalid.  Flag 2 is set to bit 14.  This table *
14960           *    is indexed in groups of four to decode most instructions    *
14961           *    which have a multiple of four opcodes.  All instructions    *
14962           *    have at least two opcodes and Flag 2 is used to determine   *
14963           *    these cases.                                                *
14964           *                                                                *
14965           ******************************************************************
14966     100F       002A      ADDL                       P    INC       P    RONP        INCREMENT P FOR DOUBLE WORD INSTRUCTIONS
14968     1010            SP1B ADD  LSR             UBA  SP1B JSZC TRP             NCRY  MAX VALID OPCODE = %51
14970     1011            UBA  ADD  LSR             COB2      ADDL LBL                   TABLE INDEX; TABLE START
14972     1012       UBA  UBB  ADD            WRX3       SP1B CSR                        YRA:=TABLE START + INDEX
14974     1013                 ADD            RAR        UBB  CSR            HBF2 TICB  RAR:=YRA; F2:=OPCODE.(14:1)
14976     1014            SP1B ADD       SP4A                 ADD                  RSB   RANDOM INDEXED BRANCH
14978     1015  COB2           ADD                            JSZ  TRP             UNC   UNIMPLEMENTED INSTRUCTIONS(%0/3)
14980     1016                 JSZ  TRP       NF2             JSL  CMPT            F2    UNIM.(%4/5);COMPARE STRINGS TRANSLATED(%6/7)
14982     1017                 ADD                            JSB  TCCS            UNC   TEST CONDITION CODE AND SET TOS(%8/11)
14984     1018                 ADD                            JSB  TCCS            UNC   TEST CONDITION CODE AND SET TOS(%12/17)
14986     1019                 ADD                            JSL  CVND            UNC   CONVERT NUMERIC DISPLAY(%20/23)
14988     101A                 ADD                            JSL  CVND            UNC   CONVERT NUMERIC DISPLAY(%24/27)
14990     101B                 ADD                            JSL  CVND            UNC   CONVERT NUMERIC DISPLAY(%30/33)
14992     101C                 ADD                            JSL  CVND            UNC   CONVERT NUMERIC DISPLAY(%34/37)
14994     101D                 JSB  LDW       NF2             JSB  LDDW            F2    LOAD WORD(%40/41); LOAD DOUBLE WORD(%42/43)
14996     101E                 JSB  TR        NF2             JSL  ABSD            F2    TRANSLATE(%44/45); ABS DECIMAL(%46/47)(CSTX)
14998     101F                 ADD                            JSL  NEGD            NF2   ;NEGATE DECIMAL(%50/51)               (CSTX)
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15001           ***************************************************************************
15002           *                                                                         *
15003           *    LDW                                                                  *
15004           *                                                                         *
15005           *    1.  Pull the RBA parameter into the TOS registers if it does         *
15006           *        not already exist.                                               *
15007           *    2.  Pop the RBA parameter if the SDEC = 1.                           *
15008           *    3.  Calculate the effective bank address (EA).                       *
15009           *    4.  PUSH a TOS register to memory to make room for the word          *
15010           *        to be loaded if necessary.                                       *
15011           *    5.  Perform bounds checking DL <= EA <= SR+SM if not in split        *
15012           *        stack mode.  If the bounds test fails in DB-S area, then         *
15013           *        32k is added to the effective address to check the DL-DB         *
15014           *        area and bounds checking is performed again.                     *
15015           *    6.  Fetch the word at the effective address.                         *
15016           *    7.  Determine whether the effective address fals in memory           *
15017           *        or a TOS register, and place the appropriate value in            *
15018           *        the return parameter.                                            *
15019           *    8.  Return if the RBA is even.                                       *
15020           *    9.  Increment the effective address and fetch the following          *
15021           *        word.  Bounds check the new effective address.                   *
15022           *   10.  Determine whether the new EA falls in memory or a TOS            *
15023           *        register.  Combine the left byte of the apporpriate value        *
15024           *        with the right byte of the previous word to form a word          *
15025           *        which spans the two words and place in the return                *
15026           *        parameter.                                                       *
15027           *   11.  Done.                                                            *
15028           *                                                                         *
15029           ***************************************************************************
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
15031	1020	LDW			ADD			CF1	SR	SM	JSZ	PULM	SP3B		SRZ	DEFAULT EVEN RBA
15033																GET PARM FROM MEMORY IF NEEDED
15034	1021				ADD					SP1B	ADD				EVEN	-
15036																SDEC = 0 OR 1?
15037	1022		RA		ADD	LSR		SF1	SP3B		CAD		SP3B	POP		ODD RBA
15039																POP IF SDEC = 1
15040	1023		UBA	DB	ADD		RH	ROD		RA	JSL	LDW1	SP2B		SR7	EFFECTIVE ADDRESS
15042																MAKE ROOM FOR RESULT; SAVE RBA
15043	1024				ADD						JSB	*+2			NF5B	-
15045																SKIP TESTS IF SPLIT STACK
15046	1025		RH	DL	JSBS	LDWC		NCRY	SP3B	RH	JSBS	LDWC			NCRY	BOUNDS TEST EA >= DL
15048																BOUNDS TEST SM >= EA
15049	1026			OPA	ADD		SP4A		RH		ADD					-
15051																-
15052	1027		UBB	SM	CAD			EPSH	SP3B	UBB	SUB		CTR		TICB	-
15054																VALUE IN MEMORY OR TOS REGISTERS?
15055	1028			SP2B	ADD			ODD		SP4A	ADD		RH		UNC	EVEN OR ODD RBA?
15057																VALUE IN MEMORY
15058	1029		RH		INC			NEXT		REGN	ADD		RH			DONE IF EVEN RBA
15060																VALUE IN TOS REGISTERS
15061	102A		UBA		ADD			ROBD	RH		ADD	RLZ	RH			FETCH NEXT WORD
15063																LEFT:=FIRST WORD ODD BYTE
15064	102B		UBA	SM	CAD				SP3B	UBA	BNDE		CTR		TICB	-
15066																VALUE IN MEMORY TO TOS REGISTERS? SM >= EA?
15067	102C				ADD					OPB	ADD	LRZ	SP2B		UNC	-
15069																RIGHT:=MEMORY EVEN BYTE
15070	102D				ADD					REGN	ADD	LRZ	SP2B			-
15072																RIGHT:=TOS REGISTER EVEN BYTE
15073	102E				ADD						ADD					-
15075																-
15076	102F		RH	SP2B	IOR		RH				ADD				NEXT	COMBINE LEFT AND RIGHT BYTES FOR RESULT
15078																DONE

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15080         ***************************************************************************
15081         *                                                                         *
15082         *     LDDW                                                                *
15083         *                                                                         *
15084         *     1.  PULL the RBA parameter into the TOS registers if it does        *
15085         *         not already exist.                                              *
15086         *     2.  POP the RBA parameter if the SDEC = 1.                          *
15087         *     3.  Calculate the effective bank address (EA).                      *
15088         *     4.  PUSH 2 TOS registers into memory to make room for the           *
15089         *         double word to be loaded if necessary.                          *
15090         *     5.  Performs bounds checking DL <= EA <= SM+SR if not in split      *
15091         *         stack mode.  If the bounds test fails in the DB-S area,         *
15092         *         then 32k is added to the effective address to check the         *
15093         *         DL-DB area and bounds checking is performed again.              *
15094         *     6.  Fetch the word at the effective address.                        *
15095         *     7.  Determine whether the EA falls in memory or a TOS register      *
15096         *         and place the appropriate value in the first word of the        *
15097         *         return parameter.                                               *
15098         *     8.  Increment EA, bounds check, and fetch the next word in          *
15099         *         the bank.                                                       *
15100         *     9.  Determine whether the second word falls in memory or a TOS      *
15101         *         register and place the appropriate value in the second          *
15102         *         word of the return parameter.                                   *
15103         *    10.  Return if the RBA is odd.                                       *
15104         *    11.  Increment EA again, bounds check, and fetch the next word       *
15105         *         of the bank.                                                    *
15106         *    12.  Combine the right byte fo the first word and left byte of       *
15107         *         the second word to form a new first word.                       *
15108         *    13.  Determine whether the new effective address falls in            *
15109         *         memory or a TOS register.  Combine the left byte of the         *
15110         *         appropriate value with the right byte of the second word        *
15111         *         to form a new second word.                                      *
15112         *    14.  Done.                                                           *
15113         *                                                                         *
15114         ***************************************************************************
```

RECORD NO	C S ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
15116	1030	LDDW			ADD			CF1	SR	SM	JSZ	PULM	SP3B		SRZ	DEFAULT EVEN RBA
15118																GET PARM FROM MEMORY IF NOT PRESENT
15119	1031				ADD					SP1B	ADD				EVEN	-
15121																SDEC = 0 OR 1?
15122	1032		RA		ADD	LSR		SF1	SP3B		CAD		SP3B	POP		ODD RBA
15124																POP IF SDEC = 1
15125	1033		UBA	DB	ADD		RH	ROD		RA	JSL	LDW2	SP2B		SRG5	EFFECTIVE ADDRESS
15127																MAKE ROOM FOR RESULT IF NEEDED; SAVE RBA
15128	1034				ADD						JSB	*+2			NF5B	-
15130																SKIP BOUNDS TEST IF SPLIT STACK
15131	1035		RH	DL	JSBS	LDWC		NCRY	SP3B	RH	JSBS	LDWC			NCRY	BOUNDS TEST EA >= DL
15133																BOUNDS TEST SM >= EA
15134	1036			OPA	ADD		SP4A	EPSH	RH		ADD					-
15136																-
15137	1037		UBB	SM	CAD			EPSH	SP3B	UBB	SUB		CTR		TICB	-
15139																VALUE IN MEMORY OR TOS REGISTERS?
15140	1038		RREG		ADD					SP4A	ADD		RH		UNC	-
15142																FIRST WORD IN MEMORY
15143	1039		UBA		INC		SP0A	ROBD		REGN	ADD		RH			FETCH NEXT WORD
15145																FIRST WORD IN TOS REGISTERS
15146	103A		UBA	SM	CAD				SP3B	UBA	BNDE		CTR		TICB	-
15148																VALUE IN MEMORY OR TOS REGISTERS?
15149	103B			SP2B	ADD			ODD		OPB	ADD		RG		UNC	EVEN OR ODD RBA?
15151																SECOND WORD IN MEMORY
15152	103C		SP0A		INC			NEXT		REGN	ADD		RG			DONE IF EVEN
15154																SECOND WORD IN TOS REGISTERS
15155	103D		UBA		ADD			ROBD	RH		ADD	RLZ	RH			FETCH THIRD WORD
15157																LEFT BYTE FIRST WORD
15158	103E		UBA	SM	CAD				SP3B	UBA	BNDE		CTR		TICB	-
15160																VALUE IN MEMORY OR TOS REGISTERS; S >= EA?
15161	103F				ADD					OPB	ADD	LRZ	SP2B		UNC	-
15163																RIGHT BYTE SECOND WORD IN MEMORY
15164	1040		RG		ADD	LRZ				REGN	ADD	LRZ	SP2B			RIGHT BYTE FIRST WORD FROM LEFT BYTE SECOND
15166																RIGHT BYTE SECOND WORD IN TOS REGISTERS
15167	1041		RH	UBA	IOR		RH		RG		ADD	RLZ				FIRST WORD
15169																LEFT BYTE SECOND WORD FROM RIGHT BYTE SECOND
15170	1042		UBB	SP2B	IOR		RG				ADD				NEXT	SECOND WORD
15172																DONE

```
 RECORD   C.S.       ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15174           ***************************************************************************
15175           *                                                                         *
15176           *    LDWC  -  DL-DB Bounds test and fetch                                 *
15177           *                                                                         *
15178           *    1.  Add 32K to effective address and refetch.                        *
15179           *    2.  Bounds test SM >= EA >= DL.                                      *
15180           *    3.  Return.                                                          *
15181           *                                                                         *
15182           *    LDW1  -  Make room for 1 TOS register in LDW                         *
15183           *                                                                         *
15184           *    1.  Increment SM.                                                    *
15185           *    2.  Write bottom TOS register to SM+1.                               *
15186           *    3.  Check for stack overflow.                                        *
15187           *    4.  Reread ALUA operand.                                             *
15188           *    5.  Return.                                                          *
15189           *                                                                         *
15190           *    LDW2  -  Make room for 2 TOS registers in LDDW                       *
15191           *                                                                         *
15192           *    1.  Add 2 to SM.                                                     *
15193           *    2.  Write bottom two TOS registers to SM+1, SM+2.                    *
15194           *    3.  Check for stack overflow.                                        *
15195           *    4.  Reread ALUA operand.                                             *
15196           *    5.  Return.                                                          *
15197           *                                                                         *
15198           ***************************************************************************
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
15200		*														
15201		*														
15202		*	DL-DB BOUNDS CHECKING													
15203		*														
15204		*														
15205	1043	LDWC			ADD				8000	RH	ADDL		RH			EXTENDED BOUNDS CHECK FOR DL-DB
15207																ADD 32K TO EA
15208	1044		UBB	DL	BNDE				SP3B	UBB	BNDE					BOUNDS TEST S >= EA AND EA >= DL (2527)
15210	1045		RH		ADD		ROD				ADD				RSB	FIRE UP MEMORY AND RETURN W/O S KILL (2527)
15212		*														
15213		*														
15214		*	MAKE ROOM FOR ONE TOS REGISTER TO BE LOADED													
15215		*														
15216		*														
15217	1046	LDW1		SM	INC		WRS			SM	INC		SM	DCSR		WRITE AT SM+1
15219																SM <= SM+1; SR <= SR-1
15220	1047			QDWN	ADD		DATA	Z		SREG	JSZC	STO2			NEG	STORE LAST TOS REGISTER
15222																CHECK FOR STACK OVERFLOW
15223	1048			RH	ADD		ROD				ADD				RSB	REFETCH WORD
15225																RETURN
15226		*														
15227		*														
15228		*	MAKE ROOM FOR TWO TOS REGISTERS TO BE LOADED													
15229		*														
15230		*														
15231	1049	LDW2		SM	INC		WRS				ADD			DCSR		WRITE AT SM+1
15233																SR <= SR - 1
15234	104A			QDWN	ADD		DATA		UBA		INC		SM	DCSR		STORE LAST TOS REGISTER
15236																SM <= SM + 2; SR <= SR - 1
15237	104B			QDWN	ADD		DATA	Z		SREG	JSZC	STO2			NEG	STORE NEXT TO LAST TOS REGISTER
15239																CHECK FOR STACK OVERFLOW
15240	104C			RH	ADD		ROD				ADD				RSB	FETCH WORD
15242																RETURN

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15244           ***************************************************************
15245           *                                                             *
15246           *    PARC                                                     *
15247           *                                                             *
15248           *    1.  Get at least 3 valid TOS registers.                  *
15249           *    2.  Swap segment #'s in the STATUS register and S-1.     *
15250           *    3.  Calculate new P and fetch the instruction.           *
15251           *    4.  Calculate current delta P and store in S-3.          *
15252           *    5.  If the code segment #'s are not equal,               *
15253           *        then goto CSEG -- the code segment set up routine.   *
15254           *    6.  Bounds check PB <= new P <= PL.                      *
15255           *    7.  Update P.                                            *
15256           *    8.  Next instruction.                                    *
15257           *                                                             *
15258           ***************************************************************
15259    104D   PARC      STA  ADD  RRZ  SPOA                  JSZ  PUL2          SRL2  EXTRACT CURRENT SEGMENT NUMBER FROM STATUS
15261                                                                                    NEED VALID RA,RB
15262    104E          0003      ADDL                                ADD                 -
15264    104F          SR   UBA  JSZS PULM      NCRY       P     ADD       RH          GET ANOTHER TOS REGISTER IF SR < 3
15266                                                                                    RH := CURRENT P
15267    1050               STA  ADD  LLZ            RB          ADD  RRZ  SP1B        CURRENT STATUS INFO
15269                                                                                    SP1B := NEW SEGMENT #
15270    1051               STA  ADD       RB         UBA  UBB    IOR       SP2B        RB := CURRENT STATUS
15272                                                                                    SP2B := NEW STATUS
15273    1052          RH   PB   SUB       RC         3FFF RC     ANDL      SP3B        RC := CURRENT DELTA P
15275                                                                                    SP3B := NEW DELTA P
15276    1053               ADD                       4000 RC     ANDL                  ;UBB:= NEWMAP FLAG                    (CSTX)
15278    1054          UBB  ADD       SPOA                UBB    ADD                   SPOA:= NEWMAP FLAG                    (CSTX)
15280    1055          SPOA SP1B JSBS CSEG      NZRO  UBB  XR12 JSBS CSEG          NZRO  EXTERNAL BRANCH?
15282                                                                                    EXTERNAL BRANCH IF PHY MAP <> NEW MAP (CSTX)
15283    1056               JSB  CNXT      UNC   SP3B PB     ADD            RONP        JUMP TO BNDS CHECK AND NEXT           (CSTX)
15285                                                                                    READ NEXT INSTRUCTION IF INTERNAL BR  (CSTX)
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15287           ************************************************************************
15288           *                                                                      *
15289           *     ENDP                                                             *
15290           *                                                                      *
15291           *     1.  Get at least 2 valid TOS registers - the test paragraph #    *
15292           *         and the return paragraph #.                                  *
15293           *     2.  Pop the test paragraph # from the stack.                     *
15294           *     3.  Next instruction if the paragraph #'s are not equal.         *
15295           *     4.  Pop the return paragraph number.                             *
15296           *     5.  Perform an XBR with the parameters remaining on the stack.   *
15297           *                                                                      *
15298           *     XBR                                                              *
15299           *                                                                      *
15300           *     6.  Get at least two TOS registers - the segment # and           *
15301           *         delta P.                                                     *
15302           *     7.  Calculate new P and fetch the instruction.                   *
15303           *     8.  Pop the two parameters from the stack.                       *
15304           *     9.  If the code segment #'s are not equal, then goto CSEG        *
15305           *         -- the code segment set up routine.                          *
15306           *    10.  Bounds check PB <= new P <= PL.                              *
15307           *    11.  Update P.                                                    *
15308           *    12.  Next instruction.                                            *
15309           *                                                                      *
15310           ************************************************************************
15311     1057  ENDP RB         JSZ  PULM      SRL2            JSZ  PUL2          SRZ   GET 1 TOS REGISTER IF ALREADY 1
15313                                                                                    GET 2 TOS REGISTERS IF CURRENTLY NONE
15314     1058       RA   UBA  JSZS NEXT      NZRO            ADD           POP        NEXT INST IF PARAGRAPH #S DON'T MATCH
15316                                                                                    POP CURRENT PARAGRAPH #
15317     1059                 ADD            CF1             ADD           POP        F1:=ENDP
15319                                                                                    POP THE TERMINATION PARAGRAPH #
15320     105A                 ADD                            ADD                      ONE CLOCK NEEDED TO MAKE SR TESTS VALID
15322                                                                                        AFTER THE POP
15323     105B  XBR           JSZ  PULM      SRL2            JSZ  PUL2          SRZ   GET 1 TOS REGISTER IF 1 ALREADY EXISTS
15325                                                                                    GET 2 TOS REGISTERS IF NONE CURRENTLY EXIST
15326     105C                 ADD                  4000      ADDL                     ;UBB:= NEWMAP FLAG                        (2324)
15328     105D       UBB  RB   AND       SP0A               ADD                      $P0A:= NEWMAP FLAG                        (2324)
15330     105E            STA  ADD  LLZ              RA      ADD  RRZ  SP1B POP       EXTRACT USER INFO FROM STATUS REGISTER
15332                                                                                  SP1B := NEW SEGMENT #; POP RA
15333     105F       RB        ADD       SP4A       UBA  UBB  IOR       SP2B POP     SAVE RB FOR ABSCENCE/TRAP HANDLING
15335                                                                                  SP2B := NEXT STATUS; POP RB
15336     1060            STA  ADD  RRZ              3FFF UBA  ANDL      SP3B         EXTRACT CURRENT SEGMENT NUMBER
15338                                                                                  SP3B := DELTA P WITHOUT TRACE BIT
15339     1061       SP0A      ADD                        UBA  ADD                    UBA:= NEWMAP FLAG, UBB:= CURRENT SEG #(CSTX)
15341     1062       UBB  SP1B JSBS CSEG       NZRO  UBA  XR12 JSBS CSEG      NZRO  SEG SETUP NEEDED FOR EXTERNAL BRANCH? (CSTX)
15343                                                                                  SEG SETUP NEEDED IF PHY MAP <> NEW MAP(2635)
15344     1063                 ADD                  SP3B PB   ADD           RONP     ;READ NEXT INSTRUCTION IF INTERNAL BR (CSTX)
15346     1064  CNXT UBB  PB   UBNE                 PL   UBB  UBNE                   BOUNDS CHECK PB <= NEW P
15348                                                                                  BOUNDS CHECK NEW P <= PL
15349     1065       SP0A      ADD                        SREG ADD       P            UBA:= NEWMAP; NEW P                       (CSTX)
15351     1066                 ADD                        UBA  ADD       XR12    NEXT XRB12:= NEWMAP FLAG, NEXT INSTRUCTION
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15354           ************************************************************************
15355           *                                                                      *
15356           *    CSEG                                                              *
15357           *                                                                      *
15358           *    1.  Determine whether segment entry is in CST (<192) or in        *
15359           *        extended CST (>192).                                          *
15360           *    2.  Read pointer to the CST table (0,0) if CST; (0,1) if CSTX.    *
15361           *    3.  Read CST table length contained in the first word of the      *
15362           *        CST table.                                                    *
15363           *    4.  Read new P bank contained in the third word of the table      *
15364           *        entry for the segment.                                        *
15365           *    5.  Read AMRTL/4 contained in first word of the table entry       *
15366           *        for the segment.                                              *
15367           *    6.  If CST table length = 0, then CST length violation abort.     *
15368           *    7.  If segment number = 0 or segment # > CST table length,        *
15369           *        then CST violation abort.                                     *
15370           *    8.  Get new P bank.                                               *
15371           *    9.  Set reference bit in AMRTL/4 and rewrite the word.            *
15372           *   10.  Fetch PB location for segment from the forth word of the      *
15373           *        table entry for the segment and get new PB.                   *
15374           *   11.  Get new PL = PB + (L/4)*4 - 1.                                *
15375           *   12.  Trap to mode violation if mode in STATUS register is          *
15376           *        non-privileged and the new segment is privileged.             *
15377           *   13.  Jump to Absence/Trace handler if the code segment is         *
15378           *        absent or if calls and returns are being traced.              *
15379           *   14.  Read next instruction.                                        *
15380           *   15.  Bounds check PB <= new P <= PL.                               *
15381           *   16.  Update P, BNKP, PB, PL, and STATUS registers.                 *
15382           *   17.  Next instruction                                              *
15383           *                                                                      *
15384           ************************************************************************
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15386           ***********************************************************************
15387           *                                                                     *
15388           * CST EXTENSION CHANGES FOR COBOL INSTRUCTIONS PARC, ENDP, XBR        *
15389           *                                                                     *
15390           * EXPECTED SP1B:= NEWSEG, SPOA:= NEWMAPFLAG                           *
15391           *                                                                     *
15392           * RETURNS OPA:= ABS(0) OR ABS(1), RH:= SEG NUMBER                     *
15393           *                                                                     *
15394           ***********************************************************************
15395           *
15396     1067  CSEG 0292       ADDL      RH                    ADD                       UBA=%1222                          (2319,2334)
15398     1068                   ADD                             ADD                       NOP FOR POSSIBLE TIMING PROBLEM    (2336)
15400     1069                   ADD             ROX3      SP1B  JSZ  CSTV RH        ZERO  READ ABS(0); RH:= SEG, JUMP IF SEG= 0 (2319)
15402     106A       SPOA        JSB  CONT       NZRO  RH          ADD                       JUMP IF NEWMAPFLAG<>0; UBB:=0292 (2320,2334)
15404     106B       UBB         CAD                             UBB  INC            ROX3      UBA:= 291; READ NRPGMSEGS (ABS %1223) (2320)
15406     106C             SP1B  ADD                             UBA  ADD            ROX3      UBA=SEG;READ LSTT BANK (ABS %1221)    (2320)
15408     106D                   INC             ROX3  UBA  OPB  JSBS CONT        NCRY      READ ABS(1); JUMP IF SEG < NPRGMSEGS  (2320)
15410     106E             UBB   JSB  CONT       ZERO  0292        ADDL                      JUMP IF SEG=NPRGMSEGS; UBB:=292  (2320,2336)
15412     106F             UBB   ADD             ROX3        OPB   ADD                       RD LSTT ADDR(ABS %1222); UBB:=LSTT BNK(CSTX)
15414     1070             UBB   JSB  *+2        NZRO        UBB   ADD        BKX3           JUMP IF LSTT BANK <> 0; BKX3:=LSTT BNK(CSTX)
15416     1071             OPA   JSZ  CSTV       ZERO              ADD                       JUMP TO CSTV IF LSTT ADDR = 0          (CSTX)
15418     1072             UBA   ADD       RH                  UBA   ADD            ROX3      RH:= LSTT; READ LSTT(0)                (CSTX)
15420     1073                   ADD                             ADD
15421     1074             SP1B  ADD  LSL                        OPB   ADD                       UBA:= SEG*2; UBB:= LSTT(0)             (CSTX)
15423     1075       UBB   SP1B  JSZS CSTV       NCRY  UBA   RH    ADD            ROX3      JUMP TO CSTV IF SEG > LSTT(0)          (CSTX)
15425                                                                                         READ LSTT(SEG*2)                       (CSTX)
15426     1076                   ADD             ROX3              ADD        BKX3           READ ABS(0); BKX3:= 0                  (CSTX)
15428     1077                   ADD                               OPB  JSZ  CSTV RH        ZERO  ;RH:= LSTT(SEG*2), JMP TO CSTV IF ZERO(CSTX)
15430     1078  CONT SPOA        ADD                               SP4A ADD        XR3           UBA:= NEWMAP FLAG; DELTA P FOR ENDP    (CSTX)
15432     1079                   ADD       SPOA  F1HB              UBA  ADD        BKX5          SAVE XBR,PARC (1)/ ENDP(0) ENTRY FLAG
15434                                                                                         BKX5 := NEWMAP FLAG                    (2635)
15435     107A             OPA   ADD             ROX3  RH    RH    INC  LSL        CF1   NPRV  READ CST ENTRY LENGTH
15437                                                                                         (SEG# * 4) + 2; F1 := NON PRIVELEGED MODE
15438     107B       UBA   UBB   ADD       SP4A  ROB3  RH    RH    ADD  LSL        SF1         READ NEW P BANK#
15440                                                                                         (SEG# * 4); F1 := PRIVELEGED MODE FROM STA
15441     107C       RREG  UBB   ADD             ROX3        UBB   JSZ  CSTV           ZERO  READ AMRTL/4
15443                                                                                         CST VIOLATION IF SEG# = 0
15444     107D             OPA   JSZ  CTLV       ZERO  00FF  OPB   ANDL       XR0         CST LENGTH VIOLATION IF CSTL = 0
15446                                                                                         SET NEW P BANK
15447     107E             OPA   ADD       RH          RH    UBA   JSZC CSTV           CRRY  RH := AMRTL/4
15449                                                                                         CST VIOLATION IF SEG# > CSTL
15450     107F       2000  UBA   IORL                        SP4A  INC            ROX3        TURN ON REFERENCE BIT IN AMRTL/4
15452                                                                                         READ PB LOCATION FOR NEW SEGMENT
15453     1080             UBA   ADD             DATA  0FFF  UBA   ANDL                       REWRITE AMRTL/4 WITH NEW REFERENCE BIT
15455                                                                                         MASK OFF L/4
15456     1081             OPA   ADD  LSL        HBF2  UBB   UBB   ADD  LSL                  F2 := MODE BIT FROM CST
15458                                                                                         CODE SEGMENT LENGTH
15459     1082       UBB         CAD       SP4A              OPB   ADD        XR1           CODE SEGMENT LENGTH - 1
15461                                                                                         SET NEW PB FROM CST ENTRY
15462     1083       RH          CSL                         UBB   UBA  ADD        XR2   NF2   GET BIT(0), WORD(0) OF CST ENTRY IN    (2635)
15464                                                                                           THE LSB OF UBA                       (2635)
15465                                                                                         SET PL := NEW PB + LENGTH - 1
15466     1084       UBA         JSZC CSTV       ZERO              JSZ  TRP6           NF1   GO TO CST VIOLATION IF TRYING TO       (2635)
15468                                                                                           ACCESS A FREE ENTRY                  (2635)
15469                                                                                         MODE VIOLATION IF CODE=PRIV AND STA=NONPRIV
15470     1085       SPOA        ADD             HBF2        BKX5  ADD        XR12          F2 := XBR,PARC (1)/ ENDP(0) ENTRY FLAG
```

RECORD NO	C S ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
15472																XRB12 := NEWMAP FLAG (2635)
15473	1086				ADD				9000		ADDL					MASK FOR CST ABSENCE AND TRACE BITS (2635)
15475	1087		UBB	RH	AND			NF2	RH	XR3	IOR				F2	MASK WITH CST ABSENCE AND TRACE BITS
15477																HB := ABSENCE FROM CST OR TRACE FROM DEL. P
15478	1088		UBA		JSB	CABT		NZRO	UBB		JSB	CABT			NEG	ABSENCE/TRACE FOR XBR,PARC
15480																ABSENCE/TRACE FOR ENDP
15481	1089			SP4A	ADD				SP3B		ADD					SEGMENT LENGTH (PL-PB)
15483																DELTA P
15484	108A				ADD				UBA	UBB	UBNE					-
15486																BOUNDS TEST SEGMENT LENGTH >= DELTA P
15487	108B			SP2B	ADD		STA			XR0	ADD		BNKP			NEW STATUS
15489																SET NEW P BANK
15490	108C				ADD				SP3B	XR1	ADD		P	RONP		-
15492																NEW P LOCATION; FETCH
15493	108D				ADD					XR1	ADD		PB			-
15495																SET NEW PB
15496	108E				ADD			NEXT		XR2	ADD		PL			NEXT INSTRUCTION
15498																SET NEW PL

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15500           ************************************************************************
15501           *                                                                      *
15502           *    CABT                                                              *
15503           *                                                                      *
15504           *    1.  Set trap parameter SPOA to zero.                              *
15505           *    2.  Clear Flag 1 for trap routine.                                *
15506           *    3.  Write all TOS registers to memory and set SR = 0.             *
15507           *    4.  Write a 4 word stack marker to memory containing X, new       *
15508           *        delta P (from RB for trace bit if ENDP), new STATUS, and      *
15509           *        delta Q.                                                      *
15510           *    5.  Update SM and Q.                                              *
15511           *    6.  Jump to the interrupt handler with either the abscence        *
15512           *        or trace label.                                               *
15513           *                                                                      *
15514           ************************************************************************
15515    108F   CABT          ADD       SPOA                ADD           CF1  F2     TRAP PARAMETER = 0
15517                                                                                  XBR,PARC(1) OR ENDP(0)?
15518    1090           OPA   ADD             HBF2       XR3  ADD       SP3B          F2 <= ABSCENCE BIT FROM AMRTL/4
15520                                                                                  DELTA P WITH TRACE BIT FOR ENDP
15521    1091           SM    INC             WRS   SR   SM   REPC      SM   DCSR SRZ  WRITE TOS REGISTERS AT SM+1,2,...
15523                                                                                  NEW SM
15524    1092           QDWN  ADD             DATA                ADD           DCSR SRL2 EMPTY TOS REGISTERS TO MEMORY
15526                                                                                  UNTIL NO TOS REGISTERS ARE LEFT
15527    1093      X          ADD             DATA  SP3B          ADD           CLSR      X => STACK MARKER
15529                                                                                  DELTA P; NO TOS REGISTERS LEFT
15530    1094      UBB        ADD             DATA  0004 SM       ADDL      SM          DELTA P => STACK MARKER
15532                                                                                  ACCOUNT FOR NEW STACK MARKER WITH SM
15533    1095           SP2B  ADD       RD    DATA  UBB           ADD       Q           STATUS => STACK MARKER
15535                                                                                  POINT Q TO NEW STACK MARKER
15536    1096      UBB  Q     SUB             DATA                ADD           INCN      DELTA Q => STACK MARKER
15538                                                                                  -
15539    1097           JSZ   IR1             F2    9F01          ADDL      SP1B        ABSCENCE TRAP
15541                                                                                  ABSCENCE LABEL
15542    1098           JSZ   IR1             UNC   A001          ADDL      SP1B        TRACE TRAP
15544                                                                                  TRACE LABEL
```

```
RECORD   C S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15546            $NOWARN
15547            ********************************************************************
15548            *                                                                  *
15549            *     TCCS                                                         *
15550            *                                                                  *
15551            *     1.  Make room for the return parameter in RH to be pushed    *
15552            *         on the stack.                                            *
15553            *     2.  Default return parameter - FALSE (0);                    *
15554            *     3.  Check for undefined condition code in the STATUS         *
15555            *         register. If so, then next instruction.                  *
15556            *     4.  Construct a G-E-L 3 bit mask from the two bit condition  *
15557            *         code contained in the status register.  L and E map to   *
15558            *         the same bits, but G must be specifically set when the   *
15559            *         condition code is zero                                   *
15560            *     5.  If any bits in the STATUS register and opcode G-E-L      *
15561            *         masks match, then set the return parameter to true.      *
15562            *     6.  Next instruction.                                        *
15563            *                                                                  *
15564            *                                                                  *
15565            ********************************************************************
15566     1099   TCCS          ADD       RH                  JSZ  PSHM          SR7   DEFAULT RETURN VALUE IS FALSE
15568                                                                                  PUSH 1 TOS REG FOR RETURN VALUE IF FULL
15569     109A         0300 STA ANDL                  0007 SP1B ANDL     RG            MASK OFF STATUS CONDITION CODE BITS
15571                                                                                  MASK OFF INSTRUCTION G-E-L FIELD
15572     109B         FCFF STA IORL                  UBA       ADD  SWAB SP2B EPSH NZRO  %FFFF IF UNDEFINED CONDITION CODE
15574                                                                                  G-E-L MASK FOR L & E CONDITION
15575     109C         UBA      INC            NZRO  0004      ADDL     SP2B          TEST FOR UNDEFINED CONDITION CODE
15577                                                                                  G-E-L MASK FOR G
15578     109D                  ADD            NEXT              ADD                  FALSE AND NEXT INST. IF UNDEFINED CON. CODE
15580                                                                                  -
15581     109E         RG   SP2B AND            ZERO              ADD                  ANY BITS IN THE G-E-L MASKS MATCH?
15583                                                                                  -
15584     109F                  CAD       RH                  ADD            NEXT  TRUE (-1) IF ANY BITS MATCH
15586                                                                                  NEXT INSTRUCTION
15587            $WARN
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15589           ***************************************************************
15590           *                                                             *
15591           *     CMPS / CMPT                                             *
15592           *     -----------                                             *
15593           *                                                             *
15594           *     1.  Adjust the number of TOS registers (4-CMPS,5-CMPT).  *
15595           *     2.  Initialize target info if target length > 0 using   *
15596           *         either PB or DB relative info.                      *
15597           *     3.  Initialize source info if source length > 0.        *
15598           *     4.  Branch to apporpriate comparison routine.           *
15599           *                                                             *
15600           ***************************************************************
15601    10A0   CMPT 0005       ADDL      RH                    ADD           SF4B UNC   NEED 5 TOS REGISTERS FOR COMPARE TRANSLATED
15603                                                                                     F4 SIGNALS TRANSLATION NEEDED
15604    10A1   CMPS       SP4A ADD             EVEN  0004      ADDL      RH             PB/DB TARGET STRING?
15606                                                                                     NEED 4 TOS REGISTERS FOR COMPARE STRINGS
15607    10A2                   ADD             SF3A            JSLZ ADJS          UNC   F3A=DB RELATIVE TARGET STRING
15609                                                                                     ADJUST THE NUMBER OF TOS REGISTERS
15610    10A3       RA   RC     IOR                   0020      ADDL      RF             ARE BOTH SOURCE AND TARGET LENGTHS ZERO
15612                                                                                     BLANK FOR FILL CHARACTER
15613    10A4            SM     ADD       SPOA        UBA       JSB  CM21          ZERO  UPPER LIMIT FOR SOURCE BOUNDS TESTS
15615                                                                                     DONE IF BOTH LENGTHS ARE ZERO
15616    10A5                   JSB  CM18       NF3A  RC        JSB  CMS1          ZERO  PB RELATIVE TARGET?
15618                                                                                     TARGET LENGTH = 0?
15619    10A6       RD          ADD  LSR                  BNKD ADD       BKX4           TARGT RBA/2
15621                                                                                     TARGET BANK IS DB RELATIVE
15622    10A7       UBA  DB     ADD       RH  ROD         SM   JSB  *+2  SP3B      NF5B  FETCH FROM TARGET BANK ADDRESS
15624                                                                                     SKIP BOUNDS TEST IF SPLIT STACK MODE;
15625                                                                                        UPPER LIMIT FOR TARGET BOUNDS TESTS
15626    10A8       UBA  DL     JSBS CM19       NCRY  UBB  UBA  JSBS CM19          NCRY  BOUNDS TEST TARGET ADDRESS >= DL
15628                                                                                     BOUNDS TEST TARGET ADDRESS <= SM
15629    10A9   CMS0 RD         CSR             HBF2  RH        ADD       XR0  ROX4      F2 DETERMINES EVEN OR ODD TARGET BYTE
15631                                                                                     TARGET OPERATIONS PERFORMED IN ALUB
15632    10AA   CMS1 RB         ADD  LSR                  RA        JSB  CM30          ZERO  SOURCE RBA/2                          (0903)
15634                                                                                     SOURCE LENGTH = 0?
15635    10AB       UBA  DB     ADD       RH  ROD         SM   JSB  *+2           NF5B  FETCH FROM SOURCE BANK ADDRESS
15637                                                                                     SKIP BOUNDS TEST IF SPLIT STACK MODE
15638    10AC       UBA  DL     JSBS CM19       NCRY  UBB  UBA  JSBS CM19          NCRY  BOUNDS TEST SOURCE ADDRESS >= DL
15640                                                                                     BOUNDS TEST SOURCE ADDRESS <= SM
15641    10AD       RH          ADD       XR0       RB        ADD           CF1  EVEN  SOURCE OPERATIONS PERFORMED IN ALUA
15643                                                                                     F1 DETERMINES EVEN OR ODD SOURCE BYTE
15644    10AE       RC          JSB  CM12       ZERO            ADD           SF1       TARGET LENGTH = 0?
15646                                                                                     ODD SOURCE BYTE
```

```
 RECORD   C.S.       ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15648           ************************************************************************
15649           *                                                                      *
15650           *    LOOP                                                              *
15651           *         1.  Extract the appropriate source and target bytes.         *
15652           *         2.  Update the RBA's and lengths.                            *
15653           *         3.  Translate source and/or target bytes if CMPT.            *
15654           *         4.  Compare source and target bytes.                         *
15655           *         5.  Fetch next source and/or target word if needed.          *
15656           *         6.  Check for interrupts.                                    *
15657           *    WHILE source and target lengths are both > 0.                     *
15658           *                                                                      *
15659           ************************************************************************
15660    10AF   CMS2       OPA  ADD  LRZ  RG   NF1         OPB  ADD  LRZ  SP1B       NF2   EVEN SOURCE BYTE
15662                                                                                       EVEN TARGET BYTE
15663    10B0   CMS3       OPA  ADD  RRZ  RG               OPB  ADD  RRZ  SP1B             ODD SOURCE BYTE
15665                                                                                       ODD TARGET BYTE
15666    10B1         RA         CAD       RA                    ADD                      DECREMENT SOURCE LENGTH
15668                                                                                       -
15669    10B2         RB         INC       RB                    ADD                      INCREMENT SOURCE RBA
15671                                                                                       -
15672    10B3         RC         CAD       RC                    ADD                      DECREMENT TARGET LENGTH
15674                                                                                       -
15675    10B4         RD         INC       RD                    JSB  CM15           F4B   INCREMENT TARGET RBA
15677                                                                                       TRANSLATE IF CMPT
15678    10B5         RA         JSB  CMS7       ZERO  RG   SP1B JSBS CM20           NZRO  SOURCE LENGTH = 0?
15680                                                                                       ARE THE SOURCE AND TARGET BYTES EQUAL?
15681    10B6                    JSB  CMS4       F1    RC         JSB  CM11           ZERO  FETCH NEXT SOURCE WORD?
15683                                                                                       TARGET LENGTH = 0?
15684    10B7                    JSB  CMS5       F2                 ADD            SF1       FETCH NEXT TARGET WORD?
15686                                                                                       SIGNAL ODD SOURCE BYTE
15687    10B8                    JSB  CMS3       UNC                ADD            SF2       NEXT ITERATION FOR BOTH ODD BYTES
15689                                                                                       SIGNAL ODD TARGET BYTE
15690    10B9   CMS4       XR0  INC       XR0   ROD                ADD            CF1   F2  FETCH NEXT SOURCE WORD
15692                                                                                       SIGNAL EVEN SOURCE BYTE; FETCH TARGET TOO?
15693    10BA         SP0A UBA  BNDE            SF2                JSB  CMS2           UNC   BOUNDS TEST EA <= SM; SIGNAL ODD TARGET BYTE
15695                                                                                       NEXT ITERATION IF ODD TARGET BYTE
15696    10BB   CMS5            JSB  CMS6       TEST       XR0     INC       XR0  ROX4      INTERRUPT PENDING?
15698                                                                                       FETCH NEXT TARGET WORD
15699    10BC                    JSB  CMS2       F3A   SP3B UBB    BNDE            CF2       NEXT ITERATION IF DB RELATIVE TARGET
15701                                                                                       BOUNDS TEST EA <= SM; SIGNAL EVEN TARGET
15702    10BD                    JSB  CMS2       UNC   SP3B SREG   UBNE                      NEXT ITERATION
15704                                                                                       BOUNDS TEST EA <= PL IF PB RELATIVE
15705    10BE   CMS6            ADD                               ADD                 NF4B  LOCAL INTERRUPT HANDLER
15707                                                                                       CMPS/CMPT?
15708    10BF                    JSZ  IRDN       ZERO  FFFF P      ADDL       P             SLOWLY TO SYSTEM INTERRUPT HANDLER
15710                                                                                       DECREMENT P FOR CMPT (DOUBLE WORD OPCODE)
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15712           ************************************************************************
15713           *                                                                      *
15714           *     1.  Fetch next target word if needed.                            *
15715           *     2.  Use blank (translated if CMPT) for source byte.              *
15716           *     LOOP                                                             *
15717           *          3.  Extract the appropriate target byte.                    *
15718           *          4.  Update the target RBA and length.                       *
15719           *          5.  Translate target byte if CMPT.                          *
15720           *          6.  Compare source (blank) and target bytes.                *
15721           *          7.  Fetch the next target word if needed.                   *
15722           *          8.  Check for interrupts.                                   *
15723           *     WHILE target length > 0.                                         *
15724           *                                                                      *
15725           ************************************************************************
15726    10C0   CMS7 RF         ADD       RG          RC         JSB  CM21           ZERO  USE BLANK FOR SOURCE BYTE
15727                                                                                      BOTH SOURCE AND TARGET LENGTHS = 0?
15728    10C1                   ADD                                JSB  CM17           F4B   -
15729                                                                                      TRANSLATE BLANK IF CMPT
15730    10C2                   JSB  CMS8       F2                 JSB  CM10           NF2   FETCH NEXT TARGET WORD?
15731                                                                                      SAME TARGET WORD?
15732           *
15733           *
15734    10C3   CMS9            ADD                          OPB   ADD  LRZ  SP1B      NF2   -
15735                                                                                      EVEN TARGET BYTE
15736    10C4   CM10 RC         CAD       RC                 OPB   ADD  RRZ  SP1B SF2       DECREMENT TARGET LENGTH
15737                                                                                      ODD TARGET BYTE; SIGNAL ODD TARGET BYTE
15738    10C5        RD         INC       RD                       JSB  CM16           F4B   INCREMENT TARGET RBA
15739                                                                                      TRANSLATE IF CMPT
15740    10C6        RC         JSB  CM21       ZERO  RG   SP1B JSBS CM20           NZRO  TARGET LENGTH = 0?
15741                                                                                      SOURCE AND TARGET BYTES EQUAL?
15742    10C7                   JSB  CMS6       TEST               JSB  CM10           NF2   INTERRUPTS PENDING?
15743                                                                                      NEXT ITERATION IF TARGET RBA IS EVEN
15744    10C8   CMS8            ADD                          XR0   INC       XR0  ROX4      -
15745                                                                                      FETCH NEXT TARGET BYTE
15746    10C9                   JSB  CMS9       F3A   SP3B UBB  BNDE            CF2       NEXT ITERATION IF TARGET IS DB RELATIVE
15747                                                                                      BOUNDS TEST EA <= SM; SIGNAL EVEN TARGET
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15759           ***************************************************************
15760           *                                                               *
15761           *     1.  Fetch next source word if needed.                     *
15762           *     2.  Use blank (translated if CMPT) for target byte.       *
15763           *     LOOP                                                      *
15764           *         3.  Extract the appropriate source byte.              *
15765           *         4.  Update the source RBA and length.                 *
15766           *         5.  Translate source byte if CMPT.                    *
15767           *         6.  Compare target (blank) and source bytes.          *
15768           *         7.  Fetch the next source word if needed.             *
15769           *         8.  Check for interrupts.                             *
15770           *     WHILE source length > 0.                                  *
15771           *                                                               *
15772           ***************************************************************
15773    10CA                   JSB  CMS9       UNC   SP3B SREG UBNE                    NEXT ITERATION
15775                                                                                   BOUNDS TEST EA <= PL IF PB RELATIVE
15776    10CB  CM11             ADD                   RF          JSB  CM26 SP1B  F4B   -
15778                                                                                   TRANSLATE TARGET BLANK IF CMPT
15779    10CC                   JSB  CM22       F1                JSB  CM14       NF1   FETCH NEXT TARGET WORD?
15781                                                                                   SAME TARGET WORD?
15782           *
15783           *
15784    10CD  CM12             ADD                   RF          JSB  CM26 SP1B  F4B   -
15786                                                                                   TRANSLATE TARGET BLANK IF CMPT
15787    10CE  CM13       OPA   ADD  LRZ  RG   NF1                 ADD                   EVEN SOURCE BYTE
15789                                                                                   -
15790    10CF  CM14       OPA   ADD  RRZ  RG   SF1                 ADD                   ODD SOURCE BYTE
15792                                                                                   -
15793    10D0       RA          CAD       RA                      ADD                   DECREMENT SOURCE LENGTH
15795                                                                                   -
15796    10D1       RB          INC       RB                      JSB  CM17       F4B   INCREMENT SOURCE RBA
15798                                                                                   TRANSLATE IF CMPT
15799    10D2       RA          JSB  CM21       ZERO  RG   SP1B JSBS CM20       NZRO  SOURCE LENGTH = 0?
15801                                                                                   SOURCE AND TARGET BYTES EQUAL?
15802    10D3                   JSB  CMS6       TEST              JSB  CM14       NF1   INTERRUPTS PENDING?
15804                                                                                   NEXT ITERATION AFTER EVEN SOURCE
15805    10D4  CM22       XR0   INC       XR0   ROD               ADD                   FETCH NEXT SOURCE WORD
15807                                                                                   -
15808    10D5       SPOA UBA   BNDE            CF1               JSB  CM13       UNC   BOUNDS TEST EA <= SM; SIGNAL EVEN SOURCE
15810                                                                                   NEXT ITERATION
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15812           *****************************************************************
15813           *                                                               *
15814           *     Translate both source and target bytes.                   *
15815           *                                                               *
15816           *     1.  Use source byte as an index into the translation table.*
15817           *     2.  Fetch the word containing the translated source byte. *
15818           *     3.  Extract left/right byte.                              *
15819           *     4.  Do not translate target if PB relative.               *
15820           *     5.  Use target byte as an index into translation table.   *
15821           *     6.  Fetch the word containing the translated target byte. *
15822           *     7.  Extract the left/right byte.                          *
15823           *     8.  Return for comparison.                                *
15824           *                                                               *
15825           *****************************************************************
15826    10D6   CM15 RE   RG   ADD  LSR                          ADD             PSHR       TRANSLATION TABLE INDEXED BY SOURCE BYTE/2
15828                                                                                       SAVE RETURN ADDRESS
15829    10D7        UBA  DB   ADD       RH   ROD        SM     JSB  *+2                NF5B  FETCH FROM TABLE BANK ADDRESS
15831                                                                                       SKIP BOUNDS TEST IF SPLIT STACK
15832    10D8        UBA  DL   JSBS CM19      NCRY       UBB    UBA  JSBS CM19           NCRY  BOUNDS TEST TABLE EA >= DL
15834                                                                                       BOUNDS TEST TABLE EA <= SM
15835    10D9             OPA  ADD                       RE     RG   ADD             POPR ODD   TABLE WORD CONTAINING TRANSLATED BYTE
15837                                                                                       EXTRACT EVEN OR ODD BYTE?
15838    10DA             XR0  ADD            ROD        UBA         ADD  LRZ  RG         UNC   REFETCH THE SOURCE WORD
15840                                                                                       EXTRACT THE EVEN BYTE
15841    10DB                  ADD            F3A        RREG        ADD  RRZ  RG              PB OR DB RELATIVE TARGET?
15843                                                                                       EXTRACT THE ODD BYTE
15844    10DC                  ADD            RSB                    ADD                    RETURN IF TARGET IS PB RELATIVE
15846                                                                                       -
15847    10DD        RE   SP1B ADD  LSR                          ADD             PSHR       TRANSLATION TABLE INDEXED BY TARGET BYTE/2
15849                                                                                       SAVE RETURN ADDRESS
15850    10DE        UBA  DB   ADD       RH   ROD        SM     JSB  *+2                NF5B  FETCH FROM TABLE BANK ADDRESS
15852                                                                                       SKIP BOUNDS TEST IF SPLIT STACK MODE
15853    10DF        UBA  DL   JSBS CM19      NCRY       UBB    UBA  JSBS CM19           NCRY  BOUNDS TEST TABLE EA >= DL
15855                                                                                       BOUNDS TEST TABLE EA <= SM
15856    10E0             OPA  ADD                       RE     SP1B ADD             POPR ODD   TABLE WORD CONTAINING TRANSLATED BYTE
15858                                                                                       EXTRACT EVEN OR ODD BYTE?
15859    10E1             XR0  ADD            ROD        UBA         ADD  LRZ  SP1B       UNC   REFETCH SOURCE WORD
15861                                                                                       EXTRACT EVEN BYTE
15862    10E2                  ADD                       RREG        ADD  RRZ  SP1B             -
15864                                                                                       EXTRACT ODD BYTE
15865    10E3                  ADD            RSB                    ADD                    TRANSLATIONS FINISHED
15867                                                                                       -
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15869           $NOWARN
15870           ********************************************************************
15871           *                                                                  *
15872           *     Translate target byte.                                       *
15873           *                                                                  *
15874           *     1.  Do not translate target if PB relative.                  *
15875           *     2.  Use target byte as an index into translation table.      *
15876           *     3.  Fetch the word containing the translated target byte.    *
15877           *     4.  Extract the left/right byte.                             *
15878           *     5.  Return for comparison.                                   *
15879           *                                                                  *
15880           *     Translate source byte.                                       *
15881           *                                                                  *
15882           *     1.  Use source byte as an index into the translation table.  *
15883           *     2.  Fetch the word containing the translated source byte.    *
15884           *     3.  Extract left/right byte.                                 *
15885           *     4.  Return for comparison.                                   *
15886           *                                                                  *
15887           ********************************************************************
15888    10E4   CM16            ADD             F3A             ADD                     PB OR DB RELTIVE TARGET?
15890    10E5                   ADD             RSB             ADD                     -
15891                                                                                   PB RELATIVE TARGET NEEDS NO TRANSLATION
15893    10E6   CM26 RE  SP1B  ADD  LSR                         ADD             PSHR    -
15894                                                                                   TRANSLATION TABLE INDEXED BY TARGET BYTE/2
15896    10E7         UBA DB    ADD       RH    ROD       SM    JSB  *+2              NF5B  SAVE RETURN ADDRESS
15897                                                                                   FETCH FROM TABLE BANK ADDRESS
15899    10E8         UBA DL    JSBS CM19       NCRY UBB  UBA   JSBS CM19            NCRY  SKIP BOUNDS TEST IF SPLIT STACK
15900                                                                                   BOUNDS TEST TABLE EA >= DL
15902    10E9               OPA ADD                   RE   SP1B  ADD             POPR ODD  BOUNDS TEST TABLE EA <= SM
15903                                                                                   TABLE WORD CONTAINING TRANSLATED BYTE
15905    10EA               XR0 ADD             ROD   UBA       ADD  LRZ  SP1B       UNC   EXTRACT EVEN OR ODD BYTE
15906                                                                                   REFETCH THE SOURCE WORD
15908    10EB                   ADD                   RREG      ADD  RRZ  SP1B             EXTRACT THE EVEN BYTE
15909                                                                                   -
15911    10EC                   ADD             RSB             ADD                     EXTRACT THE ODD BYTE
15912                                                                                   TRANSLATION FINISHED
15914           *                                                                       -
15915           *
15916    10ED   CM17 RE  RG    ADD  LSR                         ADD             PSHR    TRANSLATION TABLE INDEXED BY SOURCE BYTE/2
15917                                                                                   SAVE RETURN ADDRESS
15919    10EE         UBA DB    ADD       RH    ROD       SM    JSB  *+2              NF5B  FETCH FROM TABLE BANK ADDRESS
15920                                                                                   SKIP BOUNDS TEST IF SPLIT STACK
15922    10EF         UBA DL    JSBS CM19       NCRY UBB  UBA   JSBS CM19            NCRY  BOUNDS TEST TABLE EA >= DL
15923                                                                                   BOUNDS TEST TABLE EA <= SM
15925    10F0               OPA ADD                   RE   RG    ADD             POPR ODD  TABLE WORD CONTAINING TRANSLATED BYTE
15926                                                                                   EXTRACT EVEN OR ODD BYTE?
15928    10F1         UBA       ADD  RRZ  RG          UBA       ADD  LRZ  RG             EXTRACT ODD BYTE
15929                                                                                   EXTRACT EVEN BYTE
15931    10F2               XR0 ADD             ROD             ADD                 RSB   REFETCH THE SOURCE WORD
15932                                                                                   TRANSLATION FINISHED
15934           $WARN
15935
```

```
 RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15937            ***************************************************************
15938            *                                                               *
15939            *    PB relative target.                                        *
15940            *                                                               *
15941            *    1.  Initialize PB relative target info                     *
15942            *                                                               *
15943            *    DL-DB extended bounds test.                                *
15944            *                                                               *
15945            *    1.  Add 32k to bank address and refetch.                   *
15946            *    2.  Re-perform bounds tests.                               *
15947            *    3.  Return.                                                *
15948            *                                                               *
15949            *    FINISH.                                                    *
15950            *    1.  Set CCE if the strings are equal.                      *
15951            *    2.  Set CCG/CCL if the last bytes compared are not equal.  *
15952            *    3.  Pop all TOS register parameters.                       *
15953            *    4.  Next instruction.                                      *
15954            *                                                               *
15955            ***************************************************************
15956     10F3   CM18 RD         ADD  LSR            PL        ADD       SP3B          TARGET RBA/2
15958                                                                                  UPPER LIMIT FOR BOUNDS TESTS OF TARGET
15959     10F4        UBA  PB    ADD       RH              BNKP ADD       BKX4          PB RELATIVE BANK ADDRESS
15961                                                                                  TARGET BANK IS PB RELATIVE
15962     10F5        UBA  PB    UBNE                 PL   UBA  UBNE                   BOUNDS TEST TARGET EA >= PB
15964                                                                                  BOUNDS TEST TARGET EA <= PL
15965     10F6                   JSB  CMS0      UNC              ADD                   RETURN TO INITIALIZATION PHASE
15967                                                                                  -
15968            *
15969            *
15970     10F7   CM19       SM   ADD                  8000 RH   ADDL      RH            EXTENDED BOUNDS CHECK
15972                                                                                  ADD 32K TO EA FOR DL-DB BOUNDS CHECK
15973     10F8        UBB  DL   BNDE                 UBA  UBB  BNDE                   BOUNDS TEST EA <= SM, EA >= DL        (2527)
15975     10F9        RH         ADD           ROD              ADD               RSB   FIRE UP MEMORY TO REFETCH AND RETURN
15977                                                                                  WITHOUT KILLING S                    (2527)
15978            *
15979            *
15980     10FA   CM20             ADD           UNC  RG   SP1B SUB            CCA       SKIP CCE SETTING FOR EQUAL STRINGS
15982                                                                                  SET CCG/CCL BASED ON LAST COMPARISON
15983     10FB   CM21             ADD           CCA              ADD            CLSR      SET CCE FOR EQUAL STRINGS
15985                                                                                  POP ALL PARAMETERS
15986     10FC                   ADD                            ADD                NEXT  -
15988                                                                                  NEXT INSTRUCTION
15989     10FD   CM30 RF         ADD       RG                   JSB  CM17          F4B   RG:= RF: JUMP TO CM17 IF F4          (0903)
15991     10FE                   JSB  CMS9      UNC              ADD                   JUMP TO CMS9                         (0903)
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

15994           ************************************************************************
15995           *                                                                      *
15996           *   TRANSLATE                                                          *
15997           *                                                                      *
15998           *   1.  Get 4 valid TOS registers containing the length, the source    *
15999           *       byte address, target byte address, and the translation         *
16000           *       table address.                                                 *
16001           *   2.  If the length = 0, then do the next instruction.               *
16002           *   3.  Convert the source and target RBA'S to effective bank          *
16003           *       addresses, fetch, and bounds test.                             *
16004           *   4.  Initialize table pointers with PB or DB relative               *
16005           *       information.                                                   *
16006           *   5.  Determine whether the translation is to be rippled or not      *
16007           *       so a different algorithm can be used.                          *
16008           *                                                                      *
16009           ************************************************************************
16010           %10FF
16011    10FF   TR   0004      ADDL      RH                    ADD           CF2       NEED 4 TOS REGISTERS FOR TRANSLATE
16013                                                                                  -
16014    1100                  ADD             CF1             JSLZ ADJS           UNC   -
16016                                                                                  ADJUST THE NUMBER OF TOS REGISTERS
16017    1101       RB         ADD  LSR                    RA    JSL  TR7            ZERO  TARGET RBA/2
16019                                                                                  LENGTH = 0?
16020    1102       UBA  DB    ADD       RH    ROD       SM    JSB  *+2  SP3B       NF5B  EFFECTIVE TARGET ADDRESS; FETCH
16022                                                                                  SKIP BOUNDS TESTS IF SPLIT STACK
16023    1103       UBA  DL    JSBS TR8        NCRY  UBB   UBA   JSBS TR8            NCRY  BOUNDS TEST TARGET EA >= DL
16025                                                                                  BOUNDS TEST TARGET EA <= SM
16026    1104       RC         ADD  LSR                RH          ADD       RE   ROD       SOURCE RBA/2
16028                                                                                  TARGET OPERATIONS PERFORMED IN ALUB
16029    1105       UBA  DB    ADD       RH    ROD               JSB  *+2            NF5B  EFFECTIVE SOURCE ADDRESS; FETCH
16031                                                                                  SKIP BOUNDS TESTS IF SPLIT STACK
16032    1106       UBA  DL    JSBS TR8        NCRY  SP3B  UBA   JSBS TR8            NCRY  BOUNDS TEST SOURCE EA >= DL
16034                                                                                  BOUNDS TEST SOURCE EA <= SM
16035    1107       RH         ADD       RF              SP1B  JSB  TRN1           EVEN  SOURCE OPERATIONS PERFORMED IN ALUA
16037                                                                                  PB/DB RELATIVE TRANSLATION TABLE
16038    1108            SM    ADD       SP0A            BNKD  ADD       BKX3          SOURCE/TARGET UPPER BOUND
16040                                                                                  DB BANK FOR TRANSLATION TABLE
16041    1109            DL    ADD       SP4A      RD          ADD  LSR       SF4B      LOWER BOUND FOR TRANSLATION TABLE
16043                                                                                  TRANSLATION TABLE RBA/2; F4 := DB RELATIVE
16044    110A       UBB  DB    ADD                         DB    ADD       SP2B          EFFECTIVE ADDRESS OF FIRST WORD IN TABLE
16046                                                                                  TABLE BASE IS DB RELATIVE
16047    110B       UBA  DL    SUB                   SP3B  UBA   SUB                 TICB  BOUNDS TEST TABLE ORIGIN >= DL
16049                                                                                  BOUNDS TEST TABLE ORIGIN <= SM
16050    110C                  JSB  TRN2       UNC   8000  SP2B  ADDL      SP2B          -
16052                                                                                  ADJUST TABLE BASE BY 32K FOR DL-DB
16053    110D  TRN1    PB     ADD       SP4A            BNKP  ADD       BKX3 CF4B     LOWER BOUND FOR TABLE
16055                                                                                  PB BANK FOR TRANSLATION TABLE; F4:=PB REL.
16056    110E            SM    ADD       SP0A            PB    ADD       SP2B          SOURCE/TARGET UPPER BOUND
16058                                                                                  TABLE BASE IS PB RELATIVE
16059    110F                  ADD                 PL          ADD       SP3B          -
16061                                                                                  UPPER BOUND FOR TRANSLATION TABLE
16062    1110  TRN2 RC         ADD             EVEN  RB          CSR           HBF2    EVEN/ODD TARGET RBA?
16064                                                                                  F2:=EVEN /ODD SOURCE RBA
16065    1111                  ADD             SF1         OPB   ADD  LLZ  RG          F1:=ODD TARGET RBA
16067                                                                                  SAVE LEFT BYTE OF FIRST TARGET WORD
16068    1112       RB   RC    JSBC TR55       ZERO              ADD                 SPECIAL CASE FOR RIPPLE TRANSLATION
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16071           $NOWARN
16072           ********************************************************************
16073           *                                                                  *
16074           *    Translate string                                              *
16075           *                                                                  *
16076           *    while COUNT > 0                                               *
16077           *    loop                                                          *
16078           *      1. extract left/right source byte.                          *
16079           *      2. decrement count                                          *
16080           *      3. increment source and target RBA's.                       *
16081           *      4. fetch translation table word containing the translated   *
16082           *         byte and bounds test the table address (PB/DB relative). *
16083           *      5. extract the translated byte.                             *
16084           *      6. write the target word and bounds test if both the left   *
16085           *         and right target bytes have been filled.                 *
16086           *      7. fetch next source word and bounds test if the source     *
16087           *         RBA is even.                                             *
16088           *    endloop                                                       *
16089           *                                                                  *
16090           ********************************************************************
16091    1113   TR1       OPA  ADD  LRZ  RH   NF1             ADD                      LEFT SOURCE BYTE
16093                                                                                 -
16094    1114   TR2       OPA  ADD  RRZ  RH   SF1   RA        JSB  TR7           ZERO  RIGHT SOURCE BYTE
16096                                                                                 LOOP UNTIL COUNT = 0
16097    1115        RA        CAD       RA                  ADD                      DECREMENT COUNT
16099                                                                                 -
16100    1116        RH   RD   ADD  LSR            RB        INC       RB             (TABLE RBA + TABLE BYTE OFFSET)/2
16102                                                                                 INCREMENT SOURCE RBA
16103    1117        UBA  SP2B ADD            ROX3  RC        JSBI *+2  RC       NF4B  TABLE EFFECTIVE ADDRESS; FETCH
16105                                                                                 INCREMENT TARGET RBA; PB/DB RELATIVE TABLE?
16106    1118        UBA  SP4A BNDE           UNC   SP3B UBA  BNDE                UNC   BOUNDS TEST TABLE EA >= DL (DB RELATIVE)
16108                                                                                 BOUNDS TEST TABLE EA <= SM (DB RELATIVE)
16109    1119        RREG SP4A UBNE                 SP3B SREG UBNE                    BOUNDS TEST TABLE EA >= PB (PB RELATIVE)
16111                                                                                 BOUNDS TEST TABLE EA <= PL (PB RELATIVE)
16112    111A             OPA  ADD                  RH   RD   ADD                EVEN  TRANSLATION TABLE WORD
16114                                                                                 INDEXED TABLE RBA EVEN/ODD?
16115    111B        UBA       ADD  LRZ  RH         UBA       ADD  RRZ  RH           LEFT BYTE FOR EVEN RBA
16117                                                                                 RIGHT BYTE FOR ODD RBA
16118    111C        RF        ADD            ROD               JSB  TR3           F2  REFETCH SOURCE WORD
16120                                                                                 STORE NEXT TARGET WORD?
16121    111D        RH        ADD  RLZ  RG   SF2               JSB  TR4           F1  LEFT BYTE OF TARGET WORD; SIGNAL LEFT
16123                                                                                 FETCH NEXT SOURCE WORD?
16124    111E                  ADD            SF1               JSB  TR2           UNC SIGNAL LEFT SOURCE BYTE
16126                                                                                 NEXT ITERATION
16127    111F   TR3  RE        INC       RE   CF2   RG   RH   IOR             DATA      INCREMENT TARGET EFFECTIVE ADDRESS
16129                                                                                 WRITE TARGET WORD OF TRANSLATED BYTES
16130    1120                  JSB  TR5       TEST  UBA       ADD              ROD F1    TEST FOR INTERRUPTS
16132                                                                                 FETCH NEXT TARGET WORD, IN CASE IT IS LAST
16133    1121        SPOA RE   BNDE                                JSB  TR2           UNC BOUNDS TEST TARGET ADDRESS
16135                                                                                 NEXT ITERATION
16136    1122   TR4  RF        INC       RF   ROD   RA        JSB  TR7           ZERO  INCREMENT SOURCE EFFECTIVE ADDRESS; FETCH
16138                                                                                 DONE IF LENGTH = 0 (SKIP BOUNDS TEST)
16139    1123        SPOA UBA  BNDE           CF1               JSB  TR1           UNC BOUNDS TEST NEXT SOURCE WORD
16141                                                                                 NEXT ITERATION
```

```
 RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16143          ***************************************************************
16144          *                                                             *
16145          *    RIPPLE TRANSLATION                                       *
16146          *                                                             *
16147          *    extract left/right starting source byte.                 *
16148          *                                                             *
16149          *    while COUNT > 0                                          *
16150          *    loop                                                     *
16151          *      1. decrement count.                                    *
16152          *      2. increment source and target RBA's.                  *
16153          *      3. fetch translation table word containing translated byte *
16154          *         and bounds test the table address (PB/DB relative). *
16155          *      4. extract the translated byte.                        *
16156          *      5. write the target word and bounds test if both left and *
16157          *         right bytes have been filled.                       *
16158          *    endloop                                                  *
16159          *                                                             *
16160          ***************************************************************
16161    1124  TR55       OPA  ADD  LRZ  RH   NF1            ADD                      LEFT INITIAL SOURCE BYTE
16163                                                                                 -
16164    1125             OPA  ADD  RRZ  RH                  ADD                UNC   RIGHT INITIAL SOURCE BYTE
16166                                                                                 SKIP LEFT BYTE ASSIGNMENT ON FIRST ITERATION
16167    1126  TR6  RA         JSB  TR7        ZERO  RH      ADD  RLZ  RG             LOOP UNTIL COUNT = 0
16169                                                                                 PREVIOUS BYTE IS LEFT BYTE
16170    1127       RA         CAD       RA                 ADD                      DECREMENT COUNT
16172                                                                                 -
16173    1128       RH   RC    ADD  LSR             RB       INC       RB             (TABLE RBA + TABLE BYTE OFFSET)/2
16175                                                                                 INCREMENT SOURCE RBA
16176    1129       UBA  SP2B  ADD             ROX3  RC      JSBI *+2  RC        NF4B  TABLE EFFECTIVE ADDRESS; FETCH
16178                                                                                 INCREMENT TARGET RBA; PB/DB RELATIVE?
16179    112A       UBA  SP4A  BNDE            UNC   SP3B UBA  BNDE                UNC   BOUNDS TEST TABLE EA >= DL (DB RELATIVE)
16181                                                                                 BOUNDS TEST TABLE EA <= SM (DB RELATIVE)
16182    112B       RREG SP4A  UBNE                   SP3B SREG UBNE                      BOUNDS TEST TABLE EA >= PB (PB RELATIVE)
16184                                                                                 BOUNDS TEST TABLE EA <0 PL (PB RELATIVE)
16185    112C             OPA  ADD                    RH   RC   ADD                EVEN  TRANSLATION TABLE WORD
16187                                                                                 INDEXED TABLE RBA EVEN/ODD?
16188    112D       UBA       ADD  LRZ  RH   F2    UBA       ADD  RRZ  RH             LEFT BYTE FOR EVEN RBA; STORE TARGET WORD?
16190                                                                                 RIGHT BYTE FOR ODD RBA
16191    112E                 JSB  TR6        UNC             ADD            SF2       NEXT ITERATION
16193                                                                                 -
16194    112F       RE        INC       RE    CF2   RG   RH  IOR            DATA      INCREMENT TARGET EFFECTIVE ADDRESS
16196                                                                                 WRITE TARGET WORD OF TRANSLATED BYTES
16197    1130                 JSB  TR5        TEST  UBA       ADD            ROD       TEST FOR INTERRUPTS
16199                                                                                 FETCH NEXT TARGET WORD IN CASE IT IS LAST
16200    1131       SP0A RE   BNDE                              JSB  TR6        UNC   BOUNDS TEST TARGET WORD
16202                                                                                 NEXT ITERATION
16203          $WARN
```

```
 RECORD  C S          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16205            ************************************************************************
16206            *                                                                      *
16207            *    INTERRUPT HANDLING                                                *
16208            *                                                                      *
16209            *    1. Bounds test the target address just stored.                    *
16210            *    2. Decrement P to restart the double word instruction.            *
16211            *    3. Branch to the interrupt handler.                               *
16212            *                                                                      *
16213            *    TRANSLATION FINISHED                                              *
16214            *                                                                      *
16215            *    1. Store last target word and Bounds test if needed.              *
16216            *    2. Pop all the TOS register parameters.                           *
16217            *    3. Next instruction.                                              *
16218            *                                                                      *
16219            ************************************************************************
16220     1132   TR5  SPOA RE   BNDE                FFFF P    ADDL                      BOUNDS TEST TARGET ADDRESS JUST STORED
16222                                                                                   DECREMENT P SINCE IS IS TWO WORD OPCODE
16223     1133                  JSZ  IRDN      ZERO        UBB  ADD       P            SLOW JUMP SO THAT P GETS UPDATED FIRST
16225                                                                                   RESET P FOR SECOND WORD OF DOUBLE OPCODE
16226            *
16227            *
16228     1134   TR7  RE        INC       RE   NF2         OPB  ADD  RRZ       EPP4 NF2  INC TARGET EFFECTIVE ADDRESS; EVEN RBA?
16230                                                                                   EXTRACT RIGHT BYTE OF LAST WORD; EVEN RBA?
16231     1135        SPOA UBA  BNDE                UBB  RG   IOR            DATA      BOUNDS TEST TARGET EFFECTIVE ADDRESS
16233                                                                                   UPDATE LAST TARGET WORD WITH LEFT BYTE
16234     1136                  ADD                        ADD                 NEXT  -
16236                                                                                   NEXT INSTRUCTION
16237     1137   TR8            ADD                8000 RH   ADDL      RH            DL-DB BOUNDS CHECKING FOR BYTE ADDRESSES
16239                                                                                   ADD 32K TO EFFECTIVE ADDRESS
16240     1138        UBB  DL   BNDE                SP3B UBB  BNDE                    BOUNDS TEST SM >= EA, EA >= DL        (2527)
16242     1139        RH        ADD           ROD              ADD                 RSB   FIRE UP MEMORY AND RETURN W/O S KILL  (2527)
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16245          ************************************************************************
16246          *                                                                      *
16247          *    ABSN                                                              *
16248          *                                                                      *
16249          *    1. Get two valid TOS registers containing the source length       *
16250          *       and the source RBA.                                            *
16251          *    2. If the source length is zero, then next instruction.           *
16252          *    3. If the source length > maximum (28 ascii digits) or < 0,       *
16253          *       then signal an illegal length trap.                            *
16254          *    4. Convert the source RBA to an effective address and bounds      *
16255          *       test.                                                          *
16256          *    5. Initialize the source pointers and flags.                      *
16257          *    6. Zerofill leading blanks.                                       *
16258          *       for the first LENGTH-1 digits                                  *
16259          *       loop                                                           *
16260          *         extract the current byte (left or right).                    *
16261          *         if the byte is not an ascii blank, then exit the loop.       *
16262          *         replace the blank with an ascii zero.                        *
16263          *         if the current byte is the right byte                        *
16264          *         then rewrite the source word,                                *
16265          *              fetch the next source word, and                         *
16266          *              bounds test the new source address.                     *
16267          *         endif                                                        *
16268          *       endloop                                                        *
16269          *    7. If the loop was exited on a non-blank character                *
16270          *       then                                                           *
16271          *           back up the counter since the byte wasn't blank            *
16272          *           rewrite the last source word if the left byte was zero     *
16273          *           filled                                                     *
16274          *       endif                                                          *
16275          *                                                                      *
16276          ************************************************************************
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL  RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16278   113A  ABSN                JSZ  PULM      SRL2  SR   SM   JSZ  PUL2 SP3B       SRZ   NEED 2 TOS REGISTERS CONTAINING THE SOURCE
16280                                                                                        STRING LENGTH AND RELATIVE BYTE ADDRESS
16281   113B        RA          JSB  ABS2      ZERO  FFE3       ADDL                        DONE IF SOURCE LENGTH = 0
16283                                                                                        MAXIMUM LENGTH IS 28 DIGITS
16284   113C        RA    UBB   JSB  ABS2      CRRY  RB         ADD  LSR         SF4B       ILLEGAL LENGTH TRAP IF LENGTH < 0 OR > MAX
16286                                                                                        SOURCE RBA/2; SIGNAL ILLEGAL LENGTH
16287   113D        UBB   DB    ADD        RH  ROD               JSB  *+2                NF5B  SOURCE EFFECTIVE ADDRESS; FETCH
16289                                                                                        SKIP BOUNDS TESTING IF IN SPLIT STACK MODE
16290   113E        UBA   DL    JSBS AEBC      NCRY  SP3B UBA    JSBS AEBC               NCRY  BOUNDS TEST SOURCE EA >= DL
16292                                                                                        BOUNDS TEST SOURCE EA <= S
16293   113F        RH          ADD        SP4A                  ADD                         SOURCE EFFECTIVE ADDRESS
16295                                                                                        -
16296   1140              OPA   ADD              SF3A  RA         CAD        CTR  CF4B       WORD CONTAINING LEADING BYTE
16298                                                                                        NUMBER OF DIGITS EXCLUDING SIGN BYTE; RESET
16299                                                                                            ILLEGAL SOURCE LENGTH FLAG
16300   1141        RB          CSR                HBF2  UBA        ADD  LLZ  SP1B           F2 DETERMINES IF SOURCE BYTE IS EVEN OR ODD
16302                                                                                        SAVE LEFT BYTE IN CASE RIGHT STARTING BYTE
16303           *
16304           *
16305           *   ZERO FILL LEADING BLANKS
16306           *
16307           *
16308   1142              OPA   ADD  LRZ  RH    NF2               ADD                      EXTRACT EVEN SOURCE BYTE IF EVEN RBA
16310                                                                                        -
16311   1143              OPA   ADD  RRZ  RH    SF2               JSB  ABS0           CTR0  EXTRACT ODD SOURCE BYTE IF ODD RBA
16313                                                                                        IF DIGITS REMAINING = 0, CHECK SIGN BYTE
16314   1144        0020        ADDL                     3000      ADDL                       CONSTANT FOR LEADING BLANK COMPARISON
16316                                                                                        ZERO FILL FOR LEFT BYTE REPLACEMENT
16317   1145        RH    UBA   JSBS *+5         NZRO  UBB         ADD              DCTR F2  IS CURRENT SOURCE BYTE A LEADING BLANK?
16319                                                                                        DECREMENT # OF DIGITS REMAINING
16320   1146        0030        ADDL                     UBB       JSB  *-3  SP1B       UNC   ZERO FILL FOR RIGHT BYTE REPLACEMENT
16322                                                                                        SAVE LEFT BYTE ZERO FILL AND ITERATE IF EVEN
16323   1147        UBA   SP1B  IOR              DATA              ADD                        COMBINE AND WRITE LEFT AND RIGHT BYTES
16325                                                                                        -
16326   1148              SP4A  INC        SP4A  ROD               ADD                      INCREMENT SOURCE EFFECTIVE ADDRESS; FETCH
16328                                                                                        -
16329   1149                    JSB  *-7         UNC   SP3B UBA    BNDE           CF2       NEXT ITERATION
16331                                                                                        BOUNDS TEST SOURCE EA <= S; SIGNAL EVEN RBA
16332           *
16333           *
16334           *    CLEAN UP AFTER LEADING BLANK FILL
16335           *
16336           *
16337   114A        RH    SP1B  IOR              NF2               ADD              ICTR      COMBINE LEFT AND RIGHT BYTES OF LAST WORD
16339                                                                                        RETRY LAST SOURCE BYTE AS A DIGIT
16340   114B        UBA         ADD              DATA  0039        ADDL       SP2B           REWRITE LAST WORD IF CURRENT BYTE IS ODD
16342                                                                                        CONSTANT ("9") FOR RANGE TEST
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16344           ************************************************************************
16345           *                                                                      *
16346           *    ABSN (continued)                                                  *
16347           *                                                                      *
16348           *    1. Validate the remaining digits.                                 *
16349           *       for the each remaining digit                                   *
16350           *       loop                                                           *
16351           *         extract the current byte (left or right).                    *
16352           *         If the byte is not an ascii digit                            *
16353           *         then go to the trap exit.                                    *
16354           *         If the current byte is the right byte                        *
16355           *         then                                                         *
16356           *             rewrite the source word.                                 *
16357           *             fetch the next source word, and                          *
16358           *             bounds test the new source address.                      *
16359           *         endif                                                        *
16360           *       endloop                                                        *
16361           *    2. Strip the possible overpunch from the last byte.               *
16362           *    3. Test the stripped byte for invalid ascii digit.                *
16363           *    4. Set the condition code for CCA based on the sign of the        *
16364           *       source (only positive CCG or negative CCL).                    *
16365           *    5. Rewrite the last source word containing the stripped sign      *
16366           *       byte.                                                          *
16367           *    6. Perform proper SDEC.                                           *
16368           *    7. Clear overflow bit in the STATUS register.                     *
16369           *    8. Check for illegal ascii digit trap or illegal operand          *
16370           *       length trap.                                                   *
16371           *    9. Next instruction.                                              *
16372           *                                                                      *
16373           ************************************************************************
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16375           *
16376           *     VALIDATE NUMERIC DIGITS FOR REMAINDER OF SOURCE STRING
16377           *
16378   114C            OPA  ADD  LRZ  RH   NF2             ADD                      EXTRACT EVEN SOURCE BYTE IF EVEN RBA
16380                                                                                 -
16381   114D            OPA  ADD  RRZ  RH   SF2             JSB  ABS1           CTR0 EXTRACT ODD SOURCE BYTE IF ODD RBA
16383                                                                                 IF DIGITS REMAINING = 0, CHECK SIGN BYTE
16384   114E      0030       ADDL                      SP2B ADD            DCTR       CONSTANT ("0") FOR RANGE TEST
16386                                                                                 DECREMENT # OF DIGITS REMAINING; ("9")
16387   114F      RH   UBA  JSBS ABS2       NCRY  UBB  RH   JSBS ABS2          NCRY ILLEGAL ASCII CHARACTER IF BYTE < "0"
16389                                                                                 ILLEGAL ASCII CHARACTER IF BYTE > "9"
16390   1150            JSB  *-3        NF2        RH   ADD  RLZ  SP1B            NEXT ITERATION IF CURRENT RBA IS EVEN
16392                                                                                 SAVE EVEN BYTE
16393   1151      SP4A INC        SP4A ROD                ADD                      INCREMENT SOURCE EA AND FETCH NEXT WORD
16395                                                                                 -
16396   1152            JSB  *-6        UNC   SP3B UBA  BNDE           CF2        NEXT ITERATION
16398                                                                                 BOUNDS TEST SOURCE EA < S; SIGNAL EVEN RBA
16399           *
16400           *   CLEAN UP AFTER LEADING BLANK FILL AND CHECK FOR LAST BLANK BYTE
16401           *
16402   1153  ABS0      ADD                       0020      ADDL                     -
16404                                                                                 BLANK FOR LAST BYTE COMPARISON
16405   1154            ADD                       RH   UBB  SUB                NZRO -
16407                                                                                 IS CURRENT CHARACTER A BLANK
16408   1155            ADD                       0030      ADDL      RH             -
16410                                                                                 ZERO FILL LAST BYTE IF BLANK
16411           *
16412           *   STRIP OVERPUNCH AND TEST VALIDITY
16413           *
16414   1156  ABS1      ADD                                 JSB  STPO           UNC  -
16416                                                                                 STRIP POSSIBLE OVERPUNCH FROM LAST BYTE
16417   1157      0030       ADDL                      0039      ADDL                 CONSTANT ("0") FOR NUMERIC RANGE TEST
16419                                                                                 CONSTANT ("9") FOR NUMERIC RANGE TEST
16420   1158      RH   UBA  JSBS ABS2       NCRY  UBB  RH   JSBS ABS2          NCRY ILLEGAL ASCII CHARACTER IF BYTE < "0"
16422                                                                                 ILLEGAL ASCII CHARACTER IF BYTE > "9"
16423   1159            OPA  ADD  RRZ  SP4A CF3A            INC            CCA  F2  SAVE RIGHT BYTE; RESET ASCII DIGIT TRAP
16425                                                                                 POSITIVE OR NEGATIVE OVERPUNCH?  SET CCG
16426   115A      RB   RA   ADD            ODD             CAD            CCA       IS SIGN IN LEFT OR RIGHT BYTE OF LAST WORD?
16428                                                                                 SET CCL IF SIGN WAS NEGATIVE OVERPUNCH
16429   115B      RH   SP1B IOR            DATA            ADD                      OLD LEFT BYTE AND STRIPPED RIGHT BYTE
16431                                                                                 -
16432   115C      RB   RA   ADD            EVEN  RH        ADD  RLZ                 REWRITE LAST WORD IF RIGHT SIGN BYTE
16434                                                                                 SHIFT STRIPPED SIGN BYTE TO LEFT BYTE
16435   115D      UBB  SP4A IOR            DATA            ADD                      STRIPPED LEFT BYTE AND OLD RIGHT BYTE
16437                                                                                 -
16438   115E  ABS2      CIR  ADD            EVEN            ADD            EPOP      SDEC = 0 OR 1 (LAST BIT OF OPCODE)?
16440                                                                                 POP THE LENGTH PARAMETER
16441   115F            ADD                 EPOP            ADD            CLO       POP THE RBA PARAMETER IF SDEC=1
16443                                                                                 CLEAR THE OVERFLOW BIT IN STATUS WORD
16444   1160            JSZ  TRPA       F3A             JSZ  TRPR           F4B  ILLEGAL ASCII DIGIT TRAP?
16446                                                                                 ILLEGAL OPERAND LENGTH TRAP?
16447   1161            ADD                             ADD                 NEXT -
16449                                                                                 NEXT INSTRUCTION
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16451           ************************************************************************
16452           *                                                                      *
16453           *   ALGN                                                               *
16454           *                                                                      *
16455           *   1. Get 4 valid TOS registers containing the source length,         *
16456           *      source RBA, target length, and target RBA.                      *
16457           *   2. If either the source length or the target length are zero,      *
16458           *      then next instruction.                                          *
16459           *   3. Illegal source length trap if either the source or target       *
16460           *      length are > 28 or the source leading > source length or        *
16461           *      the target leading > target length.                             *
16462           *                                                                      *
16463           ************************************************************************
16464           *
16465    1162   ALGN 0004      ADDL      RH                    ADD                     NEED 4 TOS REGISTERS CONTAINING OPERANDS
16467                                                                                   -
16468    1163        FFE3      ADDL      SP4A                  JSLZ ADJS          UNC  MAX SOURCE/TARGET COUNT LENGTH = 28 DIGITS
16470                                                                                   ADJUST THE # OF TOS REGISTERS
16471    1164        RA        ADD  RRZ  SP0A CF3A  RC         ADD  RRZ  SP3B CF4B     EXTRACT SOURCE COUNT; CLEAR LENGTH TRAP FLAG
16473                                                                                   EXTRACT TARGET COUNT; CLEAR ASCII DIGIT TRAP
16474    1165        UBA       JSB  ALG2      ZERO  UBB        JSB  ALG2          ZERO DONE IF SOURCE COUNT = 0
16476                                                                                   DONE IF TARGET COUNT = 0
16477    1166        RA        ADD  LRZ  XR1  SF3A  RC         ADD  LRZ  XR1           EXTRACT SOURCE TRAILING; SET LENGTH TRAP
16479                                                                                   EXTRACT TARGET TRAILING
16480    1167        SP0A UBA  JSBS ALG2      NCRY  SP3B UBB   JSBS ALG2          NCRY SOURCE TRAILING < SOURCE COUNT?
16482                                                                                   TARGET TRAILING < TARGET COUNT?
16483    1168        RREG SP4A JSB  ALG2      CRRY  RREG SP4A  JSB  ALG2          CRRY SOURCE COUNT < MAX COUNT?
16485                                                                                   TARGET COUNT < MAX COUNT?
16486    1169        SP0A XR1  SUB       XR0  CF3A  SP3B XR1   SUB       XR0           SOURCE LEADING, TARGET LEADING        (2635)
16488                                                                                    RESET LENGTH TRAP FLAG               (2635)
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16490           $NOWARN
16491           ********************************************************************
16492           *                                                                  *
16493           *     ALGN (continued)                                             *
16494           *                                                                  *
16495           *     1. Convert the RBA of the last source byte to an effective   *
16496           *        address, fetch, and bounds test.                          *
16497           *     2. Save the last source word, so that it may be restored.    *
16498           *     3. Strip a possible overpunch from the last source byte and  *
16499           *        rewrite the last source byte with its unsigned value.     *
16500           *     4. Save the source sign conditions for use in overpunching   *
16501           *        the last target byte with a sign.                         *
16502           *                                                                  *
16503           ********************************************************************
16504           *
16505   116A          SPOA       CAD                          SM   ADD        RG              SOURCE COUNT - 1
16507                                                                                          MAKE SM AVAILABLE FOR BOUNDS TESTING
16508   116B          RB    UBA  ADD  LSR             RB   UBA  ADD        RF              RBA OF LAST SOURCE BYTE/2
16510                                                                                          RBA OF LAST SOURCE BYTE
16511   116C          UBA   DB   ADD        RH   ROD             JSB  *+2            NF5B  EFFECTIVE ADDRESS OF LAST SOURCE WORD
16513                                                                                          SKIP BOUNDS TEST IF SPLIT STACK MODE
16514   116D          UBA   DL   JSBS AEBC       NCRY RG   UBA  JSBS AEBC           NCRY  BOUNDS TEST EA >= DL
16516                                                                                          BOUNDS TEST EA <= SM
16517   116E               OPA  ADD        XR3        RF        ADD                 EVEN  SAVE CONTENTS OF LAST SOURCE WORD
16519                                                                                          LEFT OR RIGHT SIGN BYTE?
16520   116F          UBA       ADD  LRZ  RH         UBA       ADD  RRZ  RH              EXTRACT LEFT SOURCE BYTE
16522                                                                                          EXTRACT RIGHT SOURCE BYTE
16523   1170          RH        ADD       XR4                  JSB  STPO           UNC   SAVE ADDRESS OF LAST SOURCE WORD
16525                                                                                          STRIP POSSIBLE OVERPUNCH FROM SOURCE BYTE
16526   1171          RF        JSB  *+3       ODD             JSB  *+5            NF1   LEFT OR RIGHT SIGN BYTE?
16528                                                                                          DO NOTHING IF ALREADY UNSIGNED
16529   1172               OPA  ADD  RRZ             RH        ADD  RLZ                  RIGHT SOURCE BYTE
16531                                                                                          STRIPPED LEFT SIGN BYTE
16532   1173          UBA  UBB  IOR             DATA             JSB  *+3            UNC   REWRITE LAST SOURCE WORD
16534                                                                                          ELSE
16535   1174               OPA  ADD  LLZ             RH        ADD  RRZ                  LEFT SOURCE BYTE
16537                                                                                          STRIPPED RIGHT SIGN BYTE
16538   1175          UBA  UBB  IOR             DATA             ADD                       REWRITE LAST SOURCE WORD
16540                                                                                          -
16541   1176                    CAD       XR2  F2               ADD        XR2      F1    SOURCE IS POSITIVE                (2310)
16543                                                                                          SOURCE IS SIGNED
16544   1177                    ADD       XR2                    CAD        XR2           SOURCE IS NEGATIVE                (2310)
16546                                                                                          SOURCE IS UNSIGNED
16547           $WARN
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16549           ************************************************************************
16550           *                                                                      *
16551           *    ALGN (continued)                                                  *
16552           *                                                                      *
16553           *    1. Initialize the target pointers, bounds test, and set up        *
16554           *       for all target  store/fetch operations to be performed         *
16555           *       in ALUB.                                                       *
16556           *    2. Initialize the source pointers, bounds test, and set up        *
16557           *       for all source store/fetch operations to be performed          *
16558           *       in ALUA.                                                       *
16559           *                                                                      *
16560           ************************************************************************
16561           *
16562    1178        RD        ADD  LSR             RD        CSR             HBF2        TARGET RBA/2
16564                                                                                     F2:=EVEN/ODD TARGET BYTE
16565    1179        UBA  DB   ADD       RH   ROD              JSB  *+2             NF5B  TARGET EFFECTIVE ADDRESS
16567                                                                                     SKIPS BOUNDS TESTS IF SPLIT STACK
16568    117A        UBA  DL   JSBS AEBC      NCRY  RG   UBA  JSBS AEBC            NCRY  BOUNDS TEST EA >= DL
16570                                                                                     BOUNDS TEST EA <= SM
16571    117B        RB        ADD  LSR             RH        ADD       RF   ROD       SOURCE RBA/2
16573                                                                                     TARGET OPERATIONS IN ALUB
16574    117C        UBA  DB   ADD       RH   ROD   RH        JSB  *+2  XR4       NF5B  SOURCE EFFECTIVE ADDRESS
16576                                                                                     SKIP BOUNDS TESTS IF SPLIT STACK
16577                                                                                     SAVE EFFECTIVE ADDR. OF FIRST TARGET  (2635)
16578                                                                                      BYTE IN XR4.                         (2635)
16579    117D        UBA  DL   JSBS AEBC      NCRY  RG   UBA  JSBS AEBC            NCRY  BOUNDS TEST EA >= DL
16581                                                                                     BOUNDS TEST EA <= SM
16582    117E        RB        ADD            ODD   RH        ADD       RE              EVEN OR ODD SOURCE RBA?
16584                                                                                     SAVE SOURCE EA
16585    117F                  ADD            CF1         OPB  ADD  LRZ  SP3B SF1       F1=EVEN/ODD SRC BYTE
16587                                                                                     SAVE LEFT TARGET BYTE
```

```
 RECORD   C S          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16589           ************************************************************************
16590           *                                                                      *
16591           *    ALGN (continued)                                                  *
16592           *                                                                      *
16593           *    1. Define the valid digit range '0'..'9' and blank for filling    *
16594           *       by '0'.                                                        *
16595           *    2. If the number of leading source and target digits differs,     *
16596           *       then do either 3 or 4. Otherwise continue with 5.              *
16597           *    3. If the number of leading target digits > leading source        *
16598           *       digits, then zero fill the excess leading target digits.       *
16599           *    4. If the number of leading source digits > leading target        *
16600           *       digits, then validate the excess leading source digits.        *
16601           *    5. Copy the remainder of the leading digits from the source       *
16602           *       field to the target field.                                     *
16603           *    6  Copy the MIN(trailing source digits,trailing target digits)    *
16604           *       digits from the source field to the target field.              *
16605           *    7. If the number of trailing source and target digits differs,    *
16606           *       then do either 8 or 9.  Otherwise continue with 10.            *
16607           *    8. If the number of trailing target digits > trailing source      *
16608           *       digits, then zero fill the excess trailing target digits.      *
16609           *    9. If the number of trailing source digits > trailing target      *
16610           *       digits, then validate the excess trailing source digits.       *
16611           *   10. If an illegal ascii digit is encountered, then go to the       *
16612           *       trap handling section.                                         *
16613           *                                                                      *
16614           ************************************************************************
```

RECORD NO	C.S. ADDR	LABL	********** ALU A ********** RREG	SREG	FUNC	SFNC	STOR	SPSK	********** ALU B ********** RREG	SREG	FUNC	SFNC	STOR	SPEC	SKIP	COMMENT
16616	1180		0039		ADDL		SP0A		0030		ADDL		SP1B			UPPER BOUND FOR ASCII DIGIT TEST
16618																LOWER BOUND FOR ASCII DIGIT TEST
16619	1181		RF		CAD		RF		0020		ADDL		SP2B			SYNCHRONIZE TARGET BYTE EFFECTV ADDR (2307)
16621																USED FOR LEADING BLANK COMPARISONS
16622	1182			XR0	ADD					XR0	ADD					NUMBER OF LEADING SOURCE DIGITS
16624																NUMBER OF LEADING TARGET DIGITS
16625	1183		UBA	UBB	JSBS	FILL		NCRY	UBB	UBA	JSBS	VLB	BKX7		NCRY	ZEROFILL TARGET IF TL > SL
16627																VALIDATE SOURCE IF SL > TL
16628	1184				ADD				SP3B		JSB	ALG1	RH		F4B	-
16630																DONE IF ASCII DIGIT TRAP OCCURRED
16631	1185			XR0	ADD		SP4A			XR1	ADD		SP3B			DEFAULT MIN IS SOURCE LEADING LENGTH
16633																DEFALUT MIN IS TARGET TRAILING LENGTH
16634	1186		UBA	XR1	XFRS					XR0	ADD					RREGA:=SOURCE LEADING; UBA:=SOURCE TRAILING
16636																UBB:=TARGET LEADING
16637	1187		RREG	UBB	SUB			NCRY	SP3B	UBA	SUB				NCRY	MIN(SOURCE LEADING, TARGET LEADING)?
16639																MIN(SOURCE TRAILING, TARGET TRAILING)?
16640	1188			SREG	ADD		SP4A			SREG	ADD		SP3B			MIN IS TARGET LEADING LENGTH
16642																MIN IS SOURCE TRAILING LENGTH
16643	1189				ADD				RH		ADD		SP3B			-
16645																RESTORE LAST TARGET BYTE
16646	118A				ADD				SP3B	SP4A	JSB	VMLB	BKX7		NZRO	-
16648																VALIDATE AND MOVE SUM OF MINIMUMS
16649	118B			XR1	ADD					XR1	JSB	ALG1			F4B	UBA=SOURCE TRAILING;UBB=TARGET TRAILING
16651																DONE IF ASCII DIGIT TRAP OCCURRED (2307)
16652	118C		UBA	UBB	JSBS	FILL		NCRY	UBB	UBA	JSBS	VLB	BKX7		NCRY	ZEROFILL TARGET IF TF > SF
16654																VALIDATE SOURCE IF SF > TF
16655	118D				ADD			CF3A			JSB	ALG1			F4B	RESET LENGTH TRAP/LEADING BLANK FLAG
16657																DONE IF ASCII DIGIT TRAP

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16659           ************************************************************************
16660           *                                                                      *
16661           *    ALGN (continued)                                                  *
16662           *                                                                      *
16663           *    1. If the source field was signed, then overpunch the last        *
16664           *       target field byte with the sign obtained from the last         *
16665           *       source byte.                                                   *
16666           *    2. Write the last target word.                                    *
16667           *    3. Rewrite the last source word to restore its overpunched        *
16668           *       sign.                                                          *
16669           *    4. Perform the proper SDEC.                                       *
16670           *    5. Check for either of the possible traps (illegal ascii digit    *
16671           *       or illegal operand length).                                    *
16672           *    6. Next instruction.                                              *
16673           *                                                                      *
16674           ************************************************************************
16675           *
16676    118E                   JSB  *+5       F2          XR2  JSB  *+9             NEG   LAST BYTE LEFT OR RIGHT?
16678                                                                                       SOURCE SIGNED OR UNSIGNED?
16679    118F              XR2  ADD            HBF2  RF          ADD            ROD         POSITIVE OR NEGATIVE OVERPUNCH?
16681                                                                                       READ LAST TARGET WORD
16682    1190                   ADD                       SP3B        JSB  OVPN RH        UNC   -
16684                                                                                       OVERPUNCH LAST CHARACTER
16685    1191                   ADD                             OPB   ADD  LLZ
16687                                                                                       TARGET LEFT BYTE
16688    1192                   JSBS *+9       UNC   UBB   RH    IOR            DATA        DONE
16690                                                                                       REWRITE LAST TARGET WORD
16691    1193              XR2  ADD            HBF2  RF          INC            ROD         POSITIVE OR NEGATIVE OVERPUNCH?
16693                                                                                       READ LAST TARGET WORD
16694    1194         RG   UBB  BNDE                    SP3B        JSB  OVPN RH        UNC   BOUNDS TEST LAST TARGET WORD
16696                                                                                       OVERPUNCH LAST CHARACTER
16697    1195                   ADD                             OPB   ADD  RLZ
16699                                                                                       TARGET RIGHT BYTE
16700    1196                   JSB  *+5       UNC   RH    UBB   ADD  SWAB      DATA        DONE
16702                                                                                       REWRITE LAST TARGET WORD
16703    1197                   JSB  *+4       NF2   RF          INC            ROD         EVEN OR ODD LAST BYTE
16705                                                                                       FETCH LAST TARGET WORD
16706    1198         RG   UBB  BNDE                    SP3B        ADD                     BOUNDS TEST LAST TARGET WORD
16708                                                                                       -
16709    1199         UBB       ADD  RLZ                       OPB   ADD  RRZ             LAST TARGET BYTE
16711                                                                                       RIGHT TARGET BYTE
16712    119A                   ADD                       UBA   UBB   IOR            DATA  -
16714                                                                                       WRITE LAST TARGET BYTE
16715    119B  ALG1       XR4  ADD                             XR4   ADD            ROD         EFFECTIVE ADDRESS OF LAST SOURCE BYTE;
16717                                                                                       READ FIRST TARGET WORD                  (2635)
16718    119C                   ADD                       UBA   UBB   JSBS *+3           NZRO  NOP;JUMP IF EFFECTIVE ADDRESS OF LAST SOURCE
16720                                                                                       BYTE AND FIRST TARGET BYTE DO NOT OVERLAP
16721                                                                                                                               (2635)
16722    119D              XR3  ADD  LLZ                       OPB   ADD  RRZ             GET LAST SOURCE BYTE;
16724                                                                                       GET FIRST TARGET BYTE                   (2635)
16725    119E                   JSB  ALG2      UNC   UBA   UBB   IOR            DATA        JUMP TO REMOVE PARAMETERS;REWRITE OVERLAPPED
16727                                                                                       SOURCE AND TARGET WORD                  (2635)
16728    119F              XR4  ADD            WRD               ADD                     EFFECTIVE ADDRESS OF LAST SOURCE BYTE
16730                                                                                       -
16731    11A0              XR3  ADD            DATA              ADD                     REWRITE WORD CONTAINING LAST SOURCE BYTE
16733                                                                                       -
```

RECORD NO	C S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
16734	11A1	ALG2		CIR	ADD			ODD			ADD			EPP4		SDEC = 0 OR 1?
16736																POP ALL 4 PARAMETERS
16737	11A2				ADD			EPSH			ADD			CLO		LEAVE TARGET RBA ON STACK IF SDEC=0
16739																CLEAR OVERFLOW BIT
16740	11A3				JSZ	TRPR		F3A			JSZ	TRPA			F4B	ILLEGAL OPERAND LENGTH TRAP?
16742																ILLEGAL ASCII DIGIT TRAP?
16743	11A4				ADD						ADD				NEXT	-
16745																NEXT INSTRUCTION

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16747           ***************************************************************
16748           *                                                               *
16749           *    EXTENDED BOUNDS CHECK (DL-DB)                              *
16750           *                                                               *
16751           *    Bounds test an effective address that may be in DL-DB after *
16752           *    it has failed the DB-S inline bounds test.                 *
16753           *                                                               *
16754           ***************************************************************
16755           *
16756    11A5   AEBC          ADD                 8000 RH   ADDL      RH            EXTENDED BOUNDS CHECKING FOR DL-DB
16758                                                                                ADD 32K TO THE EFFECTIVE ADDRESS
16759    11A6        UBB  DL  BNDE                RG   UBB  BNDE                     BOUNDS TEST EA <= SM, EA >= DL        (2527)
16761    11A7        RH       ADD          ROD            ADD                 RSB   RETURN IF OK W/O S KILL, FIRE UP MEM (2527)
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16764           ************************************************************************
16765           *                                                                      *
16766           *    VALIDATE LEADING BLANKS                                           *
16767           *                                                                      *
16768           *    1. Bounds test the first byte.                                    *
16769           *    2. If the significance trigger has already been set, then         *
16770           *       just validate the remaining digits.                            *
16771           *    3. For each source byte                                           *
16772           *       loop                                                           *
16773           *         extract the source byte from the current source word.        *
16774           *         decrement the count.                                         *
16775           *         if the source byte is not a blank, then exit the loop.       *
16776           *         fetch the next source word and bounds test if needed.        *
16777           *       endloop                                                        *
16778           *    4. If all source bytes were processed, then return.               *
16779           *    5. Set the significance trigger and increment the number of       *
16780           *       bytes remaining since the current byte is non-blank.           *
16781           *                                                                      *
16782           ************************************************************************
16783           *
16784    11A8   VLB  RG   RE   BNDE                          ADD            PSHR        BOUNDS TEST
16786                                                                                   SAVE THE RETURN ADDRESS
16787    11A9                JSB  VLD       F3A       BKX7 SUB       CTR             LEADING BLANKS ALLOWED?
16789                                                                                   COMPLEMENT AND TRANSFER THE COUNT
16790    11AA          OPA  ADD  LRZ  RH    NF1             ADD                       EXTRACT LEFT SOURCE BYTE
16792                                                                                   -
16793    11AB          OPA  ADD  RRZ  RH    SF1             JSB  VRET           CTRO  EXTRACT RIGHT SOURCE BYTE
16795                                                                                   ANY BYTES REMAINING?
16796    11AC                ADD                           ADD            DCTR        -
16798                                                                                   DECREMENT BYTES REMAINING
16799    11AD                JSB  *-2       NF1   RH   SP2B JSBS *+3           NZRO  NEXT ITERATION FOR RIGHT BYTE
16801                                                                                   LEADING BLANK?
16802    11AE       RE       INC       RE  ROD             ADD            CF1  CTRO  FETCH NEXT SOURCE WORD
16804                                                                                   SIGNAL LEFT BYTE
16805    11AF                JSB  *-5       UNC   RG   UBA  BNDE                      NEXT ITERATION
16807                                                                                   BOUNDS TEST
16808    11B0                ADD            SF3A            ADD            ICTR        NON-BLANK ENCOUNTERED
16810                                                                                   RESET COUNTER
```

```
RECORD  C.S        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16812           ***************************************************************
16813           *                                                               *
16814           *    VALIDATE LEADING DIGITS                                    *
16815           *                                                               *
16816           *    1. For each source byte                                    *
16817           *       loop                                                    *
16818           *         Extract the source byte from the current source word. *
16819           *         Decrement the count.                                  *
16820           *         Validate the ASCII digit.                             *
16821           *         Fetch the next source word and bounds test if needed. *
16822           *       endloop                                                 *
16823           *    2. Return;                                                 *
16824           *                                                               *
16825           ***************************************************************
16826           *
16827    11B1   VLD       OPA  ADD  LRZ  RH   NF1            ADD                      EXTRACT RIGHT SOURCE BYTE
16829                                                                                 -
16830    11B2             OPA  ADD  RRZ  RH   SF1            JSB  VRET          CTR0  EXTRACT LEFT SOURCE BYTE
16832                                                                                 ANY BYTES REMAINING?
16833    11B3                  ADD                           ADD                      -
16835                                                                                 -
16836    11B4       SPOA RH    JSBS VTRP       NCRY  RH   SP1B JSBS VTRP       NCRY  ILLEGAL ASCII DIGIT TRAP?
16838                                                                                 ILLEGAL ASCII DIGIT TRAP?
16839    11B5                  JSB  *-3        NF1            ADD            DCTR      NEXT ITERATION FOR RIGHT SOURCE BYTE
16841                                                                                 DECREMENT BYTES REMAINING
16842    11B6       RE         INC       RE   ROD            ADD            CF1  CTR0  FETCH NEXT SOURCE WORD
16844                                                                                 SIGNAL LEFT BYTE
16845    11B7                  JSB  *-6        UNC   RG   UBA  BNDE                    NEXT ITERATION
16847                                                                                 BOUNDS TEST
```

```
RECORD  C.S                 ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16849         ************************************************************************
16850         *                                                                      *
16851         *    VALIDATE AND ZERO FILL LEADING BLANKS                             *
16852         *                                                                      *
16853         *    1. Bounds test the first byte.                                    *
16854         *    2. If the significance trigger has already been set, then         *
16855         *       just validate the remaining digits.                            *
16856         *    3. For each source byte                                           *
16857         *       loop                                                           *
16858         *         extract the source byte from the current source word.        *
16859         *         decrement the count.                                         *
16860         *         if the source byte is not a blank, then exit the loop.       *
16861         *         place a zero in the next target byte.                        *
16862         *         write the target word if both bytes have been filled.        *
16863         *         fetch the next source word and bounds test if needed.        *
16864         *       endloop                                                        *
16865         *    4. If all source bytes were processed, then return.               *
16866         *    5. Set the significance trigger and increment the number of       *
16867         *       bytes remaining since the current byte is non-blank.           *
16868         *                                                                      *
16869         ************************************************************************
16870         *
16871  11B8   VMLB RG   RE   BNDE                  SP3B      ADD       RH   PSHR        BOUNDS TEST
16873                                                                                   SAVE RETURN ADDRESS
16874  11B9                JSB  VMLD      F3A         BKX7 ADD       CTR               LEADING BLANKS ALLOWED?
16876                                                                                   TRANSFER LENGTH TO COUNTER
16877  11BA          OPA   ADD  LRZ  RH   NF1             ADD                         EXTRACT LEFT SOURCE BYTE
16879                                                                                   -
16880  11BB          OPA   ADD  RRZ  RH   SF1   RH        JSB  VRET SP3B      CTRO   EXTRACT RIGHT SOURCE BYTE
16882                                                                                   ANY BYTES REMAINING?
16883  11BC                ADD                  UBB       ADD  RLZ  SP3B DCTR        -
16885                                                                                   DECREMENT BYTES REMAINING
16886  11BD                JSB  *+4       F2    RH  SP2B JSBS *+8            NZRO   WRITE TARGET WORD?
16888                                                                                   LEADING BLANK?
16889  11BE                JSB  *-3       NF1       SP1B ADD       RH   SF2         NEXT ITERATION FOR RIGHT BYTE
16891                                                                                   REPLACE BLANK WITH ZERO
16892  11BF       RE       INC       RE   ROD             ADD                 CF1  CTRO  FETCH NEXT SOURCE WORD
16894                                                                                   SIGNAL LEFT BYTE
16895  11C0                JSB  *-6       UNC   RG  UBA   BNDE                        NEXT ITERATION
16897                                                                                   BOUNDS TEST
16898  11C1       RF       INC       RF        SP3B SP1B IOR                 DATA F1   INCREMENT TARGET ADDRESS
16900                                                                                   WRITE TARGET WORD
16901  11C2       RG   UBA BNDE            CF2             JSB  *-7            UNC    BOUNDS TEST
16903                                                                                   NEXT ITERATION FOR RIGHT BYTE
16904  11C3       RE       INC       RE   ROD             ADD                 CF1  CTRO  FETCH NEXT SOURCE WORD
16906                                                                                   SIGNAL LEFT BYTE
16907  11C4                JSB  *-A       UNC   RG  UBA   BNDE                        NEXT ITERATION
16909                                                                                   BOUNDS TEST
16910  11C5                ADD            SF3A  SP3B      ADD  LRZ  RH   ICTR        NON-BLANK ENCOUNTERED
16912                                                                                   RESET COUNTER
```

```
RECORD  C S      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16914           ************************************************************************
16915           *                                                                      *
16916           *    VALIDATE AND MOVE LEADING DIGITS                                  *
16917           *                                                                      *
16918           *    1. For each source byte                                           *
16919           *       loop                                                           *
16920           *         Extract the source byte from the current source word.        *
16921           *         Decrement the count.                                         *
16922           *         Validate the ASCII digit.                                    *
16923           *         Place the source byte in the next target byte.               *
16924           *         Write the target word if both bytes have been filled.        *
16925           *         Fetch the next source word and bounds test if needed.        *
16926           *       endloop                                                        *
16927           *    2. Return;                                                        *
16928           *                                                                      *
16929           ************************************************************************
16930           *
16931    11C6   VMLD      OPA  ADD  LRZ  RH   NF1            ADD                        EXTRACT LEFT SOURCE BYTE
16933                                                                                   -
16934    11C7           OPA  ADD  RRZ  RH   SF1   RH         JSB  VRET SP3B      CTR0  EXTRACT RIGHT SOURCE BYTE
16936                                                                                   ANY BYTES REMAINING?
16937    11C8           XR3  ADD  LLZ            RF         INC                        MASK LAST WORD OF SOURCE TO HI ORDER
16939                                                                                   ADJUST RF TO BE SAME AS YREG A      (2307)
16940    11C9      UBB  XR4  SUB            NZRO  UBA        ADD                        NOP IF NOT SHARED SRCE/TARGET WORD(2307)
16942    11CA      RH   UBB  IOR       XR3       SP3B       ADD  RLZ  SP3B            IF SHARED,MERGE;SAVE JUST LEFT BYT(2307)
16944    11CB      SPOA RH   JSBS VTRP       NCRY  RH   SP1B JSBS VTRP           NCRY  ILLEGAL ASCII DIGIT TRAP?
16946                                                                                   ILLEGAL ASCII DIGIT TRAP?
16947    11CC                JSB  *+4        F2              ADD            DCTR F1    WRITE TARGET WORD?
16949                                                                                   DECREMENT BYTES REMAINING
16950    11CD                ADD              SF2            JSB  *-6           UNC   SIGNAL RIGHT TARGET BYTE
16952                                                                                   NEXT ITERATION FOR RIGHT BYTE
16953    11CE      RE        INC       RE     ROD            ADD            CF1  CTR0  FETCH NEXT SOURCE WORD
16955                                                                                   SIGNAL LEFT SOURCE BYTE
16956    11CF                JSB  *-9        UNC   RG   UBA  BNDE                       NEXT ITERATION
16958                                                                                   BOUNDS TEST
16959    11D0      RF        INC       RF          SP3B RH   IOR            DATA F1    INCREMENT TARGET ADDRESS
16961                                                                                   WRITE TARGET WORD
16962    11D1      RG   UBA  BNDE             CF2            JSB  *-A           UNC   BOUNDS TEST
16964                                                                                   NEXT ITERATION FOR RIGHT BYTE
16965    11D2      RE        INC       RE     ROD            ADD            CF1  CTR0  FETCH NEXT SOURCE WORD
16967                                                                                   SIGNAL LEFT BYTE
16968    11D3                JSB  *-D        UNC   RG   UBA  BNDE                       NEXT ITERATION
16970                                                                                   BOUNDS TEST
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

16972           ************************************************************************
16973           *                                                                      *
16974           *    FILL                                                              *
16975           *                                                                      *
16976           *    1. Bounds test the next source byte.                              *
16977           *    2. For COUNT iterations                                           *
16978           *       loop                                                           *
16979           *         Place the fill byte in the next target byte.                 *
16980           *         Write the next target word if both bytes have been filled.   *
16981           *         Decrement count.                                             *
16982           *       endloop                                                        *
16983           *    3. Return                                                         *
16984           *                                                                      *
16985           ************************************************************************
16986           *
16987    11D4   FILL RG   RE   BNDE                SP3B      ADD       RH   PSHR       BOUNDS TEST
16989                                                                                   SAVE RETURN ADDRESS
16990    11D5                JSB  *+3       F2          BKX7 ADD       CTR              EVEN OR ODD STARTING BYTE
16992                                                                                   TRANSFER LENGTH TO COUNTER
16993    11D6           SP1B ADD       RH         RH        JSB  *+2  SP3B      CTRO  FILL BYTE CONTAINED IN SP1B       (2310)
16995                                                                                   ANY BYTES REMAINING
16996    11D7                ADD            SF2             ADD            DCTR       SIGNAL RIGHT BYTE
16998                                                                                   DECREMENT BYTES REMAINING
16999    11D8       RH       ADD  RLZ           RH        JSB  VRET SP3B      CTRO  LAST BYTE IS LEFT BYTE
17001                                                                                   ANY BYTES REMAINING?
17002    11D9       RF       INC       RF   CF2  UBA  SP1B IOR            DATA       SIGNAL LEFT BYTE; INCREMENT TARGET ADDRESS
17004                                                                                   WRITE TARGET WORD
17005    11DA                JSB  *-4       UNC  RG   UBA  BNDE           DCTR       NEXT ITERATION
17007                                                                                   BOUNDS TEST AND DECREMENT BYTES REMAINING
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17009           ********************************************************************
17010           *                                                                  *
17011           *    VTRP - Trap for ILLEGAL ASCII DIGIT                           *
17012           *                                                                  *
17013           *    1.  Set illegal ascii digit flag.                             *
17014           *    2.  Return.                                                   *
17015           *                                                                  *
17016           *    VRET - Return from the V routines.                            *
17017           *                                                                  *
17018           *    1.  Restore the return address.                               *
17019           *    2.  Return.                                                   *
17020           *                                                                  *
17021           ********************************************************************
17022           *
17023    11DB   VTRP           ADD                           ADD           SF4B      -
17025                                                                                 ILLEGAL ASCII DIGIT FLAG
17026    11DC   VRET           JSB  *+1       MEDJ           ADD           POPR      FORCE RSB AFTER POPR
17028                                                                                 RESTORE RETURN ADDRESS
17029    11DD                  ADD            RSB            ADD                     RETURN
17031                                                                                 -
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17033           $NOWARN
17034           ********************************************************************
17035           *                                                                  *
17036           *    OVPN (overpunch ascii digit)                                  *
17037           *                                                                  *
17038           *    Overpunch an ascii digit corresponding to the table below.    *
17039           *    The sign of the overpunch is determined by FLAG 2.  The       *
17040           *    sign is positive if FLAG 2 is true, otherwise it is negative. *
17041           *                                                                  *
17042           *    Original          Positive overpunch      Negative overpunch  *
17043           *                                                                  *
17044           *        0                     {                       }         *
17045           *        1                     A                       J         *
17046           *        2                     B                       K         *
17047           *        3                     C                       L         *
17048           *        4                     D                       M         *
17049           *        5                     E                       N         *
17050           *        6                     F                       O         *
17051           *        7                     G                       P         *
17052           *        8                     H                       Q         *
17053           *        9                     I                       R         *
17054           *                                                                  *
17055           ********************************************************************
17056    11DE   OVPN 0030      ADDL                          ADD                    '0' FOR COMPARISON
17058    11DF        UBA  RH   SUB           ZERO  0010 RH   ADDL              F2   SKIP FOR SPECIAL 0 OVERPUNCH
17060    11E0             UBB  ADD       RH   RSB   0019 RH   ADDL      RH        OVERPUNCH POSITIVE OR NEGATIVE 1-9; RETURN
17062    11E1                  ADD           RSB   007B       ADDL      RH   F2   POSITIVE 0 OVERPUNCH
17064    11E2                  ADD           RSB   007D       ADDL      RH        NEGATIVE 0 OVERPUNCH
17066           $WARN
```

```
RECORD  C.S.          ********** ALU A **********  ************ ALU B ************
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17068           *****************************************************************
17069           *                                                               *
17070           *     STPO (Strip overpunch)                                    *
17071           *                                                               *
17072           *     Strips the overpunch from an ascii character and set FLAG 1   *
17073           *     to TRUE if the character is signed or FALSE if the character  *
17074           *     is unsigned.  If signed, FLAG 2 is set to TRUE if positive,   *
17075           *     otherwise it is set to FALSE.                             *
17076           *                                                               *
17077           *     NOTE: This routine does not check for invalid overpunch.  *
17078           *                                                               *
17079           *****************************************************************
17080    11E3   STPO 0040      ADDL                          ADD           CF1       '@'; DEFAULT UNSIGNED QUANTITY
17082    11E4        UBA  RH   SUB           NCRY            ADD           SF2       SKIP IF RH > '@'; DEFAULT POSITIVE
17084    11E5             ADD              RSB   0049       ADDL                    RETURN IF NOT OVERPUNCHED; 'I'
17086    11E6        UBB  RH   SUB           NCRY  FFF0 RH   ADDL                    SKIP IF RH > 'I'; 'A-I' => '1-9'
17088    11E7        UBB       ADD      RH   RSB            ADD           SF1       POSITIVE 1-9 OVERPUNCHED
17090    11E8        0030      ADDL                  FF85 RH ADDL                NZRO  '0'; SKIP IF RH <> '{'
17092    11E9        UBA       ADD                   UBA     ADD       RH        RSB   POSITIVE 0;
17094    11EA        UBA       ADD           CF2     FF83 RH ADDL                NZRO  NEGATIVE; SKIP IF RH <> '}'
17096    11EB        0052      ADDL                  UBA     ADD       RH        RSB   'R'; NEGATIVE 0
17098    11EC        UBA  RH   SUB           NCRY    FFE7 RH ADDL                    SKIP IF RH > 'R'; 'J-R' => (-) '1-9'
17100    11ED        UBB       ADD      RH   RSB            ADD                     NEGATIVE 1-9 OVERPUNCHED
17102    11EE                  ADD           CF1             ADD           SF2  RSB  INVALID- DEFAULT UNSIGNED POSITIVE
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17105           ***************************************************************
17106           *                                                               *
17107           *    REMO (Remove overpunch)                                    *
17108           *                                                               *
17109           *    Strips the overpunch from the ascii digit, but does not set*
17110           *    FLAGS 1 and 2.                                             *
17111           *                                                               *
17112           *    NOTE: This routine does not check for invalid overpunch.   *
17113           *                                                               *
17114           ***************************************************************
17115   11EF    REMO 0040      ADDL                      ADD                     '@'; DEFAULT UNSIGNED QUANTITY
17117   11F0         UBA  RH   SUB          NCRY         ADD                     SKIP IF RH > '@'; DEFAULT POSITIVE
17119   11F1              RH   ADD          RSB   0049   ADDL                    RETURN IF NOT OVERPUNCHED; 'I'
17121   11F2         UBB  RH   SUB          NCRY  FFF0 RH ADDL                    SKIP IF RH > 'I'; 'A-I' => '1-9'
17123   11F3         UBB       ADD     RH   RSB         ADD                     POSITIVE 1-9 OVERPUNCHED
17125   11F4         0030      ADDL                FF85 RH ADDL          NZRO    '0'; SKIP IF RH <> '{'
17127   11F5         UBA       ADD                 UBA     ADD      RH   RSB     POSITIVE 0;
17129   11F6         UBA       ADD                 FF83 RH ADDL          NZRO    NEGATIVE; SKIP IF RH <> '}'
17131   11F7         0052      ADDL                UBA     ADD      RH   RSB     'R'; NEGATIVE 0
17133   11F8         UBA  RH   SUB          NCRY  FFE7 RH ADDL                    SKIP IF RH > 'R'; 'J-R' => (-) '1-9'
17135   11F9         UBB       ADD     RH   RSB         ADD                     NEGATIVE 1-9 OVERPUNCHED
17137   11FA                   ADD                      ADD                RSB  INVALID- DEFAULT UNSIGNED POSITIVE
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17140           ********************************************************************
17141           *                                                                  *
17142           *   STPS (Strip sign)                                              *
17143           *                                                                  *
17144           *   Decodes an ascii sign byte (non-overpunched).  The only valid  *
17145           *   ASCII signs are:                                               *
17146           *      +,blank  for positive, and                                  *
17147           *      -        for negative.                                      *
17148           *                                                                  *
17149           *   NOTE:  FLAG 4 is set for an invalid sign.                      *
17150           *                                                                  *
17151           ********************************************************************
17152    11FB  STPS 0020       ADDL                          ADD            CF1          ' '; DEFAULT UNSIGNED
17154    11FC                  ADD            SF2   UBA  RH  SUB                NZRO  POSITIVE; SKIP RETURN IF RH <> ' '
17156    11FD       002B       ADDL                          ADD                RSB   '+'
17158    11FE                  ADD            SF1   UBA  RH  SUB                NZRO  SIGN EXISTS; SKIP IF RH <> '+'
17160    11FF       002D       ADDL                          ADD                RSB   '-'
17162    1200                  ADD            CF2   UBA  RH  SUB                NZRO  NEGATIVE; SKIP RETURN IF RH <> '-'
17164    1201                  ADD                          ADD                RSB
17165    1202                  ADD            RSB           ADD            SF4B       RETURN WITH INVALID ASCII DIGIT
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17168            ************************************************************************
17169            *                                                                      *
17170            *    CVND (Convert Numeric Display)                                    *
17171            *                                                                      *
17172            *    1. Get three valid TOS registers including the source count,      *
17173            *       source RBA, and target RBA.                                    *
17174            *    2. If the source length is zero, then next instruction.           *
17175            *    3. Illegal operand length trap if source length < 0 or source     *
17176            *       length > 28.                                                   *
17177            *    4. Initialize the target pointers, bounds test, and set up for    *
17178            *       all target fetch/store operations to be performed in ALUB.     *
17179            *    5. Initialize the source pointers, bounds test, and set up for    *
17180            *       all source fetch/store operations to be performed in ALUA.     *
17181            *    6. Define the valid digit range ('0' -> '9') and blank for        *
17182            *       filling by zero.                                               *
17183            *                                                                      *
17184            ************************************************************************
17185            *
17186            *
17187     1203   CVND 0003      ADDL      RH        FFE3      ADDL      SP2B            CVND HAS 3 TOS REGISTER PARAMETERS
17189                                                                                    THE MAXIMUM SOURCE COUNT IS 28
17190     1204                  ADD                           JSLZ ADJS           UNC   -
17192                                                                                    ADJUST THE NUMBER OF TOS REGISTERS
17193     1205            SM    ADD       RG        RA        JSB  COUT           ZERO  SAVE SM FOR BOUNDS TESTING
17195                                                                                    DONE IF THE SOURCE COUNT = 0
17196     1206       RC         ADD  LSR       SF3A  RA   SP2B JSB  COUT           CRRY  TARGET RBA/2; SET ILLEGAL LENGTH TRAP
17198                                                                                    ILLEGAL LENGTH TRAP IF COUNT < 0 OR > 31
17199     1207       UBA  DB    ADD       RH        ROD       JSB  *+2            NF5B  FETCH EFFECTIVE TARGET ADDRESS
17201                                                                                    SKIP BOUNDS TESTS IF SPLIT STACK MODE
17202     1208       UBA  DL    JSBS CEBC      NCRY  RG   UBA  JSBS CEBC          NCRY  BOUNDS TEST EA >= DL
17204                                                                                    BOUNDS TEST SM >= EA
17205     1209       RB         ADD  LSR       CF3A  RH        ADD       RF   ROD       SOURCE RBA/2; RESET ILLEGAL LENGTH TRAP
17207                                                                                    TARGET OPERATIONS PERFORMED IN ALUB
17208     120A       UBA  DB    ADD       RH        ROD       JSB  *+2            NF5B  FETCH EFFECTIVE SOURCE ADDRESS
17210                                                                                    SKIP BOUNDS TESTS IF SPLIT STACK MODE
17211     120B       UBA  DL    JSBS CEBC      NCRY  RG   UBA  JSBS CEBC          NCRY  BOUNDS TEST EA >= DL
17213                                                                                    BOUNDS TEST SM >= EA
17214     120C       RB         ADD            ODD   RC        CSR            HBF2      EVEN/ODD SOURCE RBA?
17216                                                                                    F2 <- LEFT/RIGHT TARGET BYTE
17217     120D                  ADD            CF1       OPB  ADD  LRZ  SP3B SF1       F1 <- LEFT/RIGHT SOURCE BYTE
17219                                                                                    SAVE LEFT BYTE OF TARGET WORD
17220     120E                  ADD                 RF        CAD       RF              -
17222                                                                                    SYNCHRONIZE TARGET ADDRESS
17223     120F       RH         ADD       RE        0030      ADDL      SP1B            SOURCE OPERATIONS PERFORMED IN ALUA
17225                                                                                    LOWER BOUND ("0") FOR NUMERIC TEST
17226     1210       0039       ADDL      SP0A      0020      ADDL      SP2B            UPPER BOUND ("9") FOR NUMERIC TEST
17228                                                                                    USED FOR LEADING BLANK TESTS
```

```
RECORD  C S.                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17230           ***************************************************************
17231           *                                                               *
17232           *    CVND (continued)                                           *
17233           *                                                               *
17234           *    1. Determine which of the five types of source field exist *
17235           *       based on the bits 12,13,14 of the second word of the opcode.*
17236           *       0 - Leading Sign Separate                               *
17237           *       1 - Trailing Sign Separate                              *
17238           *       2 - Leading Sign Overpunch or Unsigned                  *
17239           *       3 - Trailing Sign Overpunch or Unsigned                 *
17240           *       4-7 - Unsigned                                          *
17241           *    2. If Unsigned (options 4-7), then move all the source     *
17242           *       digits to the target and zerofill the leading blanks.   *
17243           *                                                               *
17244           ***************************************************************
17245           *
17246   1211          SP4A ADD  LSR                       ADD                    SECOND WORD OF OPCODE
17248                                                                            -
17249   1212               ADD                  0007 UBA  ANDL                   -
17251                                                                            EXTRACT SOURCE TYPE BITS
17252   1213    UBB        JSBC CTSS      ZERO  UBB       JSB  CLSS        ZERO  TRAILING SIGN SEPARATE
17254                                                                            LEADING SIGN SEPARATE
17255   1214    UBA        JSBC CLSO      ZERO              ADD                   LEADING SIGN OVERPUNCHED
17257                                                                            -
17258   1215    UBA        JSBC CTSO      ZERO  RA          ADD       BKX7        TRAILING SIGN OVERPUNCHED
17260                                                                            SOURCE CHARACTER COUNT
17261   1216               ADD                              JSL  VMLB        UNC   UNSIGNED
17263                                                                            VALIDATE AND MOVE (SOURCE CHARACTER COUNT)
17264   1217               ADD            CF3A              JSB  CUNS        UNC   RESET SIGNIFICANCE TRIGGER
17266                                                                            FIXUP LAST TARGET WORD FOR UNSIGNED
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17268           ************************************************************************
17269           *                                                                      *
17270           *    NBAF (Next Byte Address Fetch)                                    *
17271           *                                                                      *
17272           *    if the fetch flag (F1) is false                                   *
17273           *    then                                                              *
17274           *      extract left byte                                               *
17275           *      set the fetch flag (F1) to true                                 *
17276           *    else                                                              *
17277           *      extract right byte                                              *
17278           *      set the fetch flag (F1) to false                                *
17279           *      fetch the next source word                                      *
17280           *    endif                                                             *
17281           *                                                                      *
17282           *                                                                      *
17283           *    NBAS (Next Byte Address Store)                                    *
17284           *                                                                      *
17285           *    if the store flag (F2) is false                                   *
17286           *    then                                                              *
17287           *      deposit the left byte                                           *
17288           *      set the store flag (F2) to true                                 *
17289           *    else                                                              *
17290           *      combine left and right bytes                                    *
17291           *      set the store flag (F2) to false                                *
17292           *      store the next target word                                      *
17293           *      bounds test                                                     *
17294           *    endif                                                             *
17295           *                                                                      *
17296           ************************************************************************
17297    1218   NBAF       OPA  ADD  LRZ  RH   NF1             ADD                F1    EXTRACT LEFT BYTE
17299                                                                                   CURRENTLY RIGHT BYTE?
17300    1219              OPA  ADD  RRZ  RH                   ADD           SF1  RSB   EXTRACT RIGHT BYTE
17302                                                                                   SIGNAL RIGHT BYTE; RETURN
17303    121A       RE          INC       RE   ROD             ADD           CF1  RSB   FETCH NEXT SOURCE WORD
17305                                                                                   SIGNAL LEFT BYTE; RETURN
17306           *
17307           *
17308           *
17309           *
17310           *
17311    121B   NBAS            ADD            F2   RH         ADD       SP3B         CURRENTLY RIGHT BYTE?
17313                                                                                   SAVE FOR NEXT LEFT BYTE
17314    121C                   ADD            RSB  SP3B       ADD  RLZ       SF2    RETURN IF LEFT BYTE
17316                                                                                   LAS BYTE IS LEFT; SIGNAL RIGHT BYTE
17317    121D       RF          INC       RF   CF2  UBB  RH   IOR            DATA    NEXT TARGET WORD; SIGNAL LEFT BYTE
17319                                                                                   WRITE TARGET WORD
17320                                                                                   TICB TO SLOW DOWN RSB                    (1000 )
17321    121E                   ADD                       RG   UBA  BNDE             (1010)
17323    121F                   ADD                            ADD                RSB  RETURN  BUT SLOWLY                       (1010 )
17325                                                                                   BOUNDS TEST ADDRESS OF WORD BEGIN WRITTEN
```

```
 RECORD   C S          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17327           ************************************************************************
17328           *                                                                      *
17329           *    CVND (continued)                                                  *
17330           *                                                                      *
17331           *    if leading sign separate                                          *
17332           *    then                                                              *
17333           *      fetch the leading sign byte and save it                         *
17334           *      validate, zerofill, and move the remainder of the source        *
17335           *        field to the target                                           *
17336           *    else                                                              *
17337           *      validate, zerofill, and move the first N-1 digits of the        *
17338           *        source field to the target                                    *
17339           *      fetch the trailing sign byte and save it                        *
17340           *    endif                                                             *
17341           *                                                                      *
17342           *    NOTE: (Special Case) If the source length is only 1, there is     *
17343           *          a sign in the source, but no digits, so a zero is           *
17344           *          transferred to the target field to be overpunched.          *
17345           *                                                                      *
17346           *    Strip the source sign byte and set the sign flags so that the     *
17347           *    target may be overpunched.                                        *
17348           *                                                                      *
17349           ************************************************************************
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17351           *
17352           *
17353           *   LEADING SIGN SEPARATE
17354           *
17355           *
17356     1220  CLSS           JSB  NBAF      UNC   RA        CAD       BKX7           FETCH LEADING SEPARATE SIGN
17358                                                                                   SOURCE CHARACTER COUNT -1
17359     1221       RH        ADD       RD                   JSL  VMLB           UNC  SAVE LEADING SIGN
17361                                                                                   VALIDATE AND MOVE (SOURCE COUNT - 1)
17362     1222                 JSB  *+6       UNC             JSB  COUT           F4B  SIGN HANDLING BELOW
17364                                                                                   ILLEGAL ASCII DIGIT TRAP
17365           *
17366           *
17367           *   TRAILING SIGN SEPARATE
17368           *
17369           *
17370     1223  CTSS           ADD                   RA        CAD       BKX7           -
17372                                                                                   SOURCE CHARACTER COUNT - 1
17373     1224                 ADD                             JSL  VMLB           UNC  -
17375                                                                                   VALIDATE AND MOVE (SOURCE COUNT - 1)
17376     1225       RG   RE   BNDE                            JSB  COUT           F4B  BOUNDS TEST FOR SIGN BYTE FETCH
17378                                                                                   ILLEGAL ASCII DIGIT TRAP
17379     1226                 JSB  NBAF      UNC             ADD                      FETCH TRAILING SIGN
17381                                                                                   -
17382     1227       RH        ADD       RD                   ADD                      SAVE TRAILING SIGN
17384                                                                                   -
17385           *
17386           *
17387           *  SIGN HANDLING FOR SEPARATE SIGN TYPES
17388           *
17389           *
17390     1228       RA        JSBC NBAS      ZERO       SP1B ADD       RH             DOES SOURCE CONSIST OF ONLY A SIGN BYTE
17392                                                                                   ZERO REQUIRED FOR OVERPUNCH
17393     1229       RD        ADD       RH                   ADD       XRO  F2HB      RESTORE SIGN BYTE
17395                                                                                   SAVE LEFT/RIGHT TARGET BYTE
17396     122A                 ADD                             JSL  STPS           UNC  -
17398                                                                                   STRIP SIGN FROM SIGN BYTE
17399     122B                 JSB  COVP      UNC             JSB  COUT           F4B  OVERPUNCH LAST TARGET BYTE
17401                                                                                   ILLEGAL ASCII DIGIT TRAP
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17403           ************************************************************************
17404           *                                                                      *
17405           *    CVND (continued)                                                  *
17406           *                                                                      *
17407           *    If leading sign overpunched                                       *
17408           *    then                                                              *
17409           *      fetch the sign byte                                             *
17410           *      if the sign byte is blank                                       *
17411           *      then                                                            *
17412           *        zerofill the byte                                             *
17413           *        clear the significance trigger so that leading blanks are     *
17414           *          allowed                                                     *
17415           *      endif                                                           *
17416           *      save the sign byte                                              *
17417           *      remove the overpunch from the sign byte                         *
17418           *      test the stripped sign byte for validity and move it to the     *
17419           *        target field                                                  *
17420           *      validate, zerofill, and move the remainder of the source        *
17421           *        field to the target                                           *
17422           *    else                                                              *
17423           *      validate, zerofill, and move the first N-1 digits from the      *
17424           *        source to the target                                          *
17425           *      fetch the sign byte                                             *
17426           *      if the significance trigger is off and the sign is blank        *
17427           *      then                                                            *
17428           *        zerofill the sign byte                                        *
17429           *      endif                                                           *
17430           *      remove the overpunch from the sign byte                         *
17431           *      test the stripped sign byte for validity and move it to the     *
17432           *        target field                                                  *
17433           *    endif                                                             *
17434           *                                                                      *
17435           *    strip the overpunch and set the sign so that the target may       *
17436           *      be overpunched                                                  *
17437           *                                                                      *
17438           ************************************************************************
```

```
 RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17440          *
17441          *
17442          *    LEADING SIGN OVERPUNCHED
17443          *
17444          *
17445    122C  CLSO 0020      ADDL      RD                    JSB  NBAF           UNC   BLANK FOR TEST
17447                                                                                    FETCH LEADING OVERPUNCH SIGN
17448    122D       RH        ADD       RD   SF3A  RH   RD    JSBS *+2             NZRO  SAVE LEADING OVERPUNCH, SET SIGNIFICANCE
17450                                                                                    CHECK FOR LEADING BLANK
17451    122E                 ADD            CF3A  0030        ADDL      RH              CLEAR SIGNIFICANCE
17453                                                                                    ZERO FILL LEADING BLANK
17454    122F                 ADD                              JSL  REMO            UNC   -
17456                                                                                    STRIP LEADING OVERPUNCH FOR TARGET BYTE
17457    1230       SPOA RH   JSBS CTRP      NCRY  RH   SP1B  JSBS CTRP            NCRY  ILLEGAL ASCII DIGIT TRAP
17459                                                                                    ILLEGAL ASCII DIGIT TRAP
17460    1231                 JSB  NBAS      UNC   RA         CAD       BKX7            STORE STRIPPED LEADING BYTE
17462                                                                                    SOURCE CHARACTER COUNT - 1
17463    1232                 ADD                              JSL  VMLB            UNC   -
17465                                                                                    VALIDATE AND MOVE (SOURCE COUNT - 1)
17466    1233                 JSB  *+A       UNC               JSB  COUT            F4B   SIGN HANDLING BELOW
17468                                                                                    ILLEGAL ASCII DIGIT TRAP
17469          *
17470          *
17471          *    TRAILING SIGN OVERPUNCHED
17472          *
17473          *
17474    1234  CTSO           ADD                    RA         CAD       BKX7            -
17476                                                                                    SOURCE CHARACTER COUNT - 1
17477    1235                 ADD                              JSL  VMLB            UNC   -
17479                                                                                    VALIDATE AND MOVE (SOURCE COUNT - 1)
17480    1236       RG   RE   BNDE                             JSB  COUT            F4B   BOUNDS TEST OVERPUNCHED BYTE
17482                                                                                    ILLEGAL ASCII DIGIT TRAP
17483    1237       0020      ADDL      RD                    JSB  NBAF           UNC   BLANK FOR TEST
17485                                                                                    FETCH TRAILING OVERPUNCH SIGN
17486    1238                 JSB  *+2       F3A   RH   RD    SUB                  NZRO  ALLOW BLANK FILLING?
17488                                                                                    TRAILING BLANK?
17489    1239                 ADD                    0030      ADDL      RH              -
17491                                                                                    ZERO FILL TRAILING BLANK
17492    123A       RH        ADD       RD                    JSL  REMO            UNC   SAVE TRAILING OVERPUNCH
17494                                                                                    STRIP OVERPUNCH
17495    123B       SPOA RH   JSBS CTRP      NCRY  RH   SP1B  JSBS CTRP            NCRY  ILLEGAL ASCII DIGIT TRAP
17497                                                                                    ILLEGAL ASCII DIGIT TRAP
17498    123C                 JSB  NBAS      UNC               ADD                        STORE STRIPPED TRAILING BYTE
17500                                                                                    -
17501          *
17502          *
17503          *    OVERPUNCH HANDLING FOR OVERPUNCHED TYPES
17504          *
17505          *
17506    123D       RD        ADD       RH                    ADD       XRO  F2HB        RESTORE OVERPUNCH
17508                                                                                    SAVE LEFT/RIGHT TARGET BYTE
17509    123E                 ADD                              JSL  STPO            UNC   -
17511                                                                                    STRIP OVERPUNCH
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17513           ********************************************************************
17514           *                                                                  *
17515           *    CVND (continued)                                              *
17516           *                                                                  *
17517           *    1. Overpunch the last target byte and write the last target   *
17518           *       word if the source was signed.                             *
17519           *    2. If the source was unsigned, then just write the last       *
17520           *       target word.                                               *
17521           *                                                                  *
17522           ********************************************************************
17523           *
17524    123F   COVP           ADD            CF3A       XR0   JSB  *+6            NEG   RESET LEADING BLANK FLAG
17526                                                                                      LEFT OR RIGHT TARGET BYTE?
17527    1240                  ADD                       RF          ADD            ROD   -
17529                                                                                      FETCH LAST SOURCE WORD
17530    1241                  ADD                       SP3B        ADD       RH        -
17532                                                                                      LAST BYTE IS TO BE OVERPUNCHED
17533    1242                  ADD                                   JSL  OVPN           UNC   -
17535                                                                                      OVERPUNCH LAST TARGET BYTE
17536    1243                  ADD                             OPB   ADD  LLZ            -
17538                                                                                      LEFT BYTE FROM OLD TARGET WORD
17539    1244                  JSB  COUT      UNC   RH    UBB   IOR            DATA      DONE
17541                                                                                      REWRITE LAST TARGET WORD
17542           *
17543           *
17544           *
17545           *
17546           *
17547    1245                  ADD                       RF          INC            ROD   -
17549                                                                                      FETCH NEXT TARGET WORD
17550    1246          RG  UBB BNDE                      SP3B        ADD       RH        BOUNDS TEST LAST TARGET WORD
17552                                                                                      LAST BYTE IS TO BE OVERPUNCHED
17553    1247                  ADD                                   JSL  OVPN           UNC   -
17555                                                                                      OVERPUNCH LAST TARGET BYTE
17556    1248                  ADD                             OPB   ADD  RLZ            -
17558                                                                                      RIGHT BYTE FROM OLD TARGET WORD
17559    1249                  JSB  COUT      UNC   RH    UBB   ADD  SWAB       DATA      DONE
17561                                                                                      WRITE LAST TARGET WORD
17562           *
17563           *
17564           *
17565           *
17566           *
17567    124A   CUNS           JSB  COUT      NF2   RF          INC            ROD       WAS LAST TARGET WORD WRITTEN?
17569                                                                                      FETCH NEXT TARGET WORD
17570    124B          RG  UBB BNDE                      SP3B        ADD                   BOUNDS TEST LAST TARGET WORD
17572                                                                                      -
17573    124C          UBB     ADD  RLZ                        OPB   ADD  RRZ            OVERPUNCHED LEFT BYTE
17575                                                                                      RIGHT BYTE FROM OLD TARGET WORD
17576    124D                  JSB  COUT      UNC   UBA   UBB   IOR            DATA      DONE
17578                                                                                      WRITE LAST TARGET WORD
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17580           ************************************************************************
17581           *                                                                      *
17582           *    CTRP (CVND trap for illegal ascii digit)                          *
17583           *                                                                      *
17584           *    Signal an illegal ascii digit.                                    *
17585           *                                                                      *
17586           *    COUT (CVND finish routine)                                        *
17587           *                                                                      *
17588           *    1. Perform the proper SDEC.                                       *
17589           *    2. Check for either of the traps that may have occurred           *
17590           *       (illegal operand length or illegal ascii digit)                *
17591           *    3. Next instruction.                                              *
17592           *                                                                      *
17593           ************************************************************************
17594           *
17595    124E   CTRP           ADD                             ADD             SF4B       -
17597                                                                                      ILLEGAL ASCII DIGIT TRAP EXIT
17598    124F   COUT     SP4A  ADD             EVEN            ADD             EPP2       SECOND WORD OF OPCODE
17600                                                                                      POP 2 PARAMETERS
17601    1250                  ADD             EPOP            ADD             CLO        POP THIRD PARAMETER IF SDEC = 1
17603                                                                                      CLEAR THE OVERFLOW BIT IN STATUS REGISTER
17604    1251                  JSZ   TRPR      F3A             JSZ   TRPA           F4B   ILLEGAL SOURCE LENGTH TRAP
17606                                                                                      ILLEGAL ASCII DIGIT TRAP
17607    1252                  ADD                             ADD                   NEXT  -
17609                                                                                      NEXT INSTRUCTION
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17611           ***************************************************************
17612           *                                                               *
17613           *     EXTENDED BOUNDS CHECK (DL-DB)                             *
17614           *                                                               *
17615           *     Bounds test an effective address that may be in DL-DB after  *
17616           *     it has failed the DB-S inline bounds test.                *
17617           *                                                               *
17618           ***************************************************************
17619           *
17620     1253  CEBC            ADD                 8000 RH   ADDL      RH              EXTENDED BOUNDS CHECKING FOR DL-DB
17622                                                                                     ADD 32K TO THE EFFECTIVE ADDRESS
17623     1254        UBB  DL   BNDE                RG   UBB  BNDE                      BOUNDS TEST EA <= SM, EA >= DL       (2527)
17625     1255        RH        ADD           ROD               ADD                RSB   RETURN IF OK W/O S KILL, FIRE UP MEM (2527)
```

```
 RECORD   C.S.       ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17628           ************************************************************************
17629           *                                                                      *
17630           *    EDIT                                                              *
17631           *                                                                      *
17632           *    1. Get 4 valid TOS registers.                                     *
17633           *    2. Initialize the fill char (' '), float char ('$'), loop         *
17634           *       count (0), and significance trigger (off).                     *
17635           *    3. If returning from an interrupt (S = 0), then restore           *
17636           *       fill char, float char, loop count, and the significance        *
17637           *       trigger from the two extra words placed in the interrupt       *
17638           *       marker.                                                        *
17639           *    4. Initialize the target pointers, fetch the first target         *
17640           *       word and bounds test.  All the target operations will be       *
17641           *       performed in ALUB.                                             *
17642           *    5. Initialize the source pointers, fetch the first source         *
17643           *       word and bounds test.  All the source operations will be       *
17644           *       performed in ALUA.                                             *
17645           *                                                                      *
17646           ************************************************************************
```

```
 RECORD   C S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17648     1256  EDIT 0004      ADDL      RH        0020      ADDL      XR0             NEED 4 TOS REGISTER PARAMETERS
17650                                                                                  FILL CHARACTER DEFAULT IS A BLANK
17651     1257                 ADD       XR0                 JSLZ ADJS          UNC   LOOP COUNT DEFAULT IS ZERO
17653                                                                                  ADJUST THE NUMBER OF TOS REGISTERS
17654     1258       RA        JSB  *+4        ZERO  0024      ADDL      XR1             RESTARTED AFTER AN INTERRUPT?
17656                                                                                  FLOAT CHARACTER DEFAULT IS A DOLLAR SIGN ($)
17657               *
17658               *
17659               *    RESTORE STACK IF RETURNING FROM AN INTERRUPT
17660               *
17661               *
17662     1259       RB        ADD  RRZ  XR0   CF3A  RC          ADD  RRZ  XR1   POP       RESTORE LOOP COUNT
17664                                                                                  RESTORE FLOAT CHARACTER
17665     125A       RB        ADD             POS   RC          ADD  LRZ  XR0   POP       SIGNIFICANCE TRIGGER TRUE OR FALSE?
17667                                                                                  RESTORE FILL CHARACTER
17668     125B                 ADD             SF3A              JSZ  PUL2          UNC   SET SIGNIFICANCE TRIGGER IF TRUE
17670                                                                                  FETCH NEXT TWO PARAMETERS
17671     125C                 ADD                               CAD       RA    CLO   -
17673                                                                                  SET MARKER WORD; CLEAR OVERFLOW
17674               *
17675               *
17676               *    INITIALIZE BOTH SOURCE AND TARGET POINTERS, OPERANDS, AND FLAGS
17677               *
17678               *
17679     125D       RC        ADD  LSR                    SM    INC       RG              TARGET RBA/2
17681                                                                                  SM+1 FOR INITIAL BOUNDS CHECKING
17682     125E       UBA  DB   ADD       RH    ROD               JSB  *+2             NF5B  TARGET EFFECTIVE ADDRESS
17684                                                                                  SKIP BOUNDS TESTING IF SPLIT STACK MODE
17685     125F       UBA  DL   JSBS CEBC       NCRY  RG    UBA   JSBS CEBC          NCRY  BOUNDS TEST EA >= DL
17687                                                                                  BOUNDS TEST SM >= EA
17688     1260       RB        ADD  LSR              RH          ADD       RF    ROD       SOURCE RBA/2
17690                                                                                  TARGET OPERATIONS PERFORMED IN ALUB
17691     1261       UBA  DB   ADD       RH    ROD               JSB  *+2             NF5B  SOURCE EFFECTIVE ADDRESS
17693                                                                                  SKIP BOUNDS TESTING IF SPLIT STACK MODE
17694     1262       UBA  DL   JSBS CEBC       NCRY  RG    UBA   JSBS CEBC          NCRY  BOUNDS TEST EA >= DL
17696                                                                                  BOUNDS TEST SM >= EA
17697     1263       RH        ADD       RE        RC          CSR               HBF2      SOURCE OPERATIONS PERFORMED IN ALUA
17699                                                                                  F2 := EVEN/ODD TARGET RBA
17700     1264       RB        ADD             ODD   OPB   ADD  LRZ  SP3B SF1             EVEN/ODD SOURCE RBA?
17702                                                                                  SAVE LEFT TARGET BYTE
17703     1265                 ADD             CF1   RF          CAD       RF              F1 := EVEN/ODD SOURCE BYTE
17705                                                                                  SYNCHRONIZE TARGET ADDRESS
```

```
 RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17707          ***************************************************************************
17708          *                                                                         *
17709          *    EDI2 (start/restart subprogram from new RBA)                         *
17710          *                                                                         *
17711          *    Set all subprogram pointers, etc for execution of the first          *
17712          *    subprogram instruction, or after an interrupt - to                   *
17713          *    restart execution at the new subprogram RBA.                         *
17714          *                                                                         *
17715          *    EDI4 (next subprogram instruction)                                   *
17716          *                                                                         *
17717          *    Decode the next sequential byte of the edit subprogram.              *
17718          *    Fetch the next byte and extracts the left nibble for decoding        *
17719          *    through a branch table.  If the opcode is %17, then decode           *
17720          *    the right nibble for the opcode.                                     *
17721          *                                                                         *
17722          ***************************************************************************
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17724           *
17725           *
17726           *     RESTART EDIT SUBPROGRAM AT CURRENT RBA
17727           *
17728           *
17729   1266  EDI2      SM   ADD       RG                   JSL  SPGS           UNC   SM FOR SOURCE/TARGET UPPER BOUND TESTS
17731                                                                                  SET UP SUBPROGRAM POINTERS AND FLAGS
17732           *
17733           *
17734           *     NEXT SUBPROGRAM INSTRUCTION DECODING
17735           *
17736           *
17737   1267  EDI4            ADD                          JSB  NSPG           UNC   -
17739                                                                                  GET SUBPROGRAM BYTE CONTAINING OPCODE
17740   1268                  ADD                  0004    ADDL                       -
17742                                                                                  SHIFT COUNT TO GET OPCODE
17743   1269      00F0 SP1B ANDL                   UBB     REPC                       EXTRACT FIRST NIBBLE OPCODE
17745                                                                                  REPEAT SHIFTING TO OBTAIN OPCODE
17746   126A      UBA       ADD  LSR  SP4A       UBB     CAD                 ZERO   SHIFT AND SAVE OPCODE
17748                                                                                  DONE WHEN REPEAT COUNT IS ZERO
17749   126B      UBA       ADD                  EDT1    ADDL LBL                   TABLE INDEX := FIRST NIBBLE OPCODE/2
17751                                                                                  TABLE BASE
17752   126C      UBA  UBB  ADD            WRX3         ADD                        INDEXED TABLE ADDRESS
17754                                                                                  -
17755   126D                ADD            RAR          ADD                        TRANSFER TABLE ADDRESS TO RAR
17757                                                                                  -
17758   126E                ADD                         ADD                        -
17760                                                                                  -
17761   126F                ADD            RSB          ADD                        RANDOM TABLE BRANCH
17763                                                                                  -
17764           *
17765           *
17766           *       LEFT NIBBLE OPCODE BRANCH TABLE
17767           *
17768           *
17769   1270  EDT1      SP4A JSB  MC         EVEN    SP4A JSB  MA             ODD   MOVE CHARACTERS
17771                                                                                  MOVE ALPHABETICS
17772   1271                JSB  MN         UNC          ADD                        MOVE NUMERICS
17774                                                                                  MOVE NUMERICS SUPPRESSED
17775   1272      SP4A JSB  MN         EVEN    SP4A JSB  IC             ODD   MOVE NUMERICS WITH FLOATING INSERTION
17777                                                                                  INSERT CHARACTER
17778   1273      SP4A JSB  IC         EVEN    SP4A JSB  ICI            ODD   INSERT CHARACTER SUPPRESSED
17780                                                                                  INSERT CHARACTERS IMMEDIATE
17781   1274      SP4A JSB  ICI        EVEN    SP4A JSB  BRS            ODD   INSERT CHARACTERS SUPPRESSED IMMEDIATE
17783                                                                                  BRANCH IF SIGNIFICANCE TRIGGER IS TRUE
17784   1275                JSB  SUFT       UNC          ADD                        SUBTRACT FROM TARGET POINTER
17786                                                                                  SUBTRACT FROM SOURCE POINTER
17787   1276                JSB  ICP        UNC          ADD                        INSERT CHARACTER PUNCTUATION
17789                                                                                  INSERT CHARACTER PUNCTUATION SUPPRESSED
17790   1277      SP4A JSB  IS         EVEN    SP4A JSL  EDI5           ODD   INSERT CHARACTERS DEPENDING ON SIGN
17792                                                                                  SECOND NIBBLE OPCODE IF NON OF THE ABOVE
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17794           ************************************************************************
17795           *                                                                      *
17796           *    SPGS (Subprogram setup)                                           *
17797           *                                                                      *
17798           *    Sets up the subprogram pointers to point to the first byte        *
17799           *    in the EDIT subprogram which may be either PB or DB relative.     *
17800           *                                                                      *
17801           *    NSPG (Next subprogram instruction)                                *
17802           *                                                                      *
17803           *    Fetchs the next subprogram instruction byte and increments        *
17804           *    the subprogram pointers.                                          *
17805           *                                                                      *
17806           ************************************************************************
17807           *
17808   1278    SPGS      CIR  JSB  *+5       ODD   RD        ADD  LSR       PSHR       PB/DB SUBPROGRAM?
17810                                                                                    SUBPROGRAM RBA/2; SAVE RETURN ADDRESS
17811   1279          UBB  PB   ADD       RH              BNKP ADD       BKX3           PB RELATIVE EFFECTIVE ADDRESS
17813                                                                                    PB RELATIVE BANK
17814   127A          UBA  PB   UBNE                PL    UBA  UBNE           POPR       BOUNDS TEST EA >= PB
17816                                                                                    BOUNDS TEST PL >= EA; RESTORE RETURN ADDRESS
17817   127B               PB   ADD       XR3       PL        ADD       XR3            SUBPROGRAM LOWER LIMIT
17819                                                                                    SUBPROGRAM UPPER LIMIT
17820   127C          RH        ADD       XR1   ROB3          ADD                 RSB    FETCH FROM PB RELATIVE EFFECTIVE ADDRESS
17822                                                                                    RETURN
17823   127D          RD        ADD  LSR                  BNKD ADD       BKX3           SUBPROGRAM RBA/2
17825                                                                                    DB RELATIVE BANK
17826   127E          UBA  DB   ADD       RH    ROD             JSB  *+2            NF5B   DB RELATIVE EFFECTIVE ADDRESS
17828                                                                                    SKIP BOUNDS TESTING IF SPLIT STACK MODE
17829   127F          UBA  DL   JSBS CEBC       NCRY  RG   UBA  JSBS CEBC          NCRY   BOUNDS TEST EA >= DL
17831                                                                                    BOUNDS TEST SM >= EA
17832   1280               DL   ADD       XR3                   ADD           POPR       SUBPROGRAM LOWER LIMIT
17834                                                                                    RESTORE RETURN ADDRESS
17835   1281          RH        ADD       XR1   ROB3  RG        ADD       XR3            SAVE SUBPROGRAM ADDRESS AND FETCH BYTE
17837                                                                                    SUBPROGRAM UPPER LIMIT
17838   1282          RE        ADD             ROD             ADD                 RSB    REFETCH SOURCE WORD
17840                                                                                    RETURN
17841           *
17842           *
17843   1283    NSPG RD         INC       RD    ODD   RD        ADD                 EVEN   INCREMENT SUBPROGRAM RBA; LEFT/RIGHT BYTE?
17845                                                                                    LEFT/RIGHT SUBPROGRAM BYTE?
17846   1284               XR1  INC       XR1   ROB3        OPB  ADD  RRZ  SP1B       RSB    FETCH NEXT SUBPROGRAM INSTRUCTION
17848                                                                                    RIGHT SUBPROGRAM BYTE; RETURN
17849   1285               SREG ADD       RH                OPB  ADD  LRZ  SP1B            SUBPROGRAM EFFECTIVE ADDRESS
17851                                                                                    LEFT SUBPROGRAM BYTE
17852   1286               CIR  ADD             EVEN        XR3  ADD                      PB/DB SUBPROGRAM?
17854                                                                                    SUBPROGRAM UPPER LIMIT
17855   1287          RH   XR3  BNDE            RSB   UBB   RH   BNDE                     BOUNDS TEST LOWER LIMIT
17857                                                                                    BOUNDS TEST UPPER LIMIT
17858                                                                                    REMOVE TICB OF 2522 . . .                  (2526)
17859   1288          RH   XR3  BNDE                  RREG  RH   BNDE                     BOUNDS TEST LOWER PB LIMIT
17861                                                                                    BOUNDS TEST UPPER PB LIMIT
17862                                                                                    ALSO REMOVED RSB OF 2522 TO NEXT LINE (2526)
17863   1289                    ADD                         ADD                 RSB    RETURN MORE SLOWLY THAN BEFORE . . .       (2526)
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17866         ************************************************************************
17867         *                                                                      *
17868         *    MC (Move Characters)                                              *
17869         *                                                                      *
17870         *    1. Extract the immediate operand count from the right nibble      *
17871         *       of the instruction.                                            *
17872         *    2. If the immediate operand count is zero, then fetch the         *
17873         *       extended operand count in the next subprogram instruction      *
17874         *       byte.                                                          *
17875         *    3. Move count bytes from the source to the target.                *
17876         *    4. Next subprogram instruction.                                   *
17877         *                                                                      *
17878         ************************************************************************
17879         *
17880  128A   MC            ADD                  000F SP1B ANDL   .    SP1B          -
17882                                                                                 IMMEDIATE OPERAND COUNT
17883  128B                 ADD                       UBB  JSL  NSPG          ZERO  -
17885                                                                                 EXTENDED OPERAND COUNT
17886  128C                 ADD                       SP1B ADD                      -
17888                                                                                 NUMBER OF CHARACTERS TO MOVE
17889  128D                 ADD                       UBB  JSL  EDI3          ZERO  -
17891                                                                                 LOOP UNTIL COUNT = 0
17892  128E                 ADD                            JSL  NSRC          UNC   -
17894                                                                                 FETCH NEXT SOURCE BYTE
17895  128F                 ADD                            JSL  NTRG          UNC   -
17897                                                                                 STORE NEXT TARGET BYTE
17898  1290                 JSB  *-3       UNC   FFFF SP1B ADDL      SP1B          NEXT ITERATION
17900                                                                                 DECREMENT THE COUNT
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17902           ************************************************************************
17903           *                                                                      *
17904           *    MA (Move Alphabetics)                                             *
17905           *                                                                      *
17906           *    1. Extract immediate operand count from the right nibble of       *
17907           *       the instruction.                                               *
17908           *    2. If the immediate operand count is zero, then fetch the         *
17909           *       extended operand count in the next subprogram instruction      *
17910           *       byte.                                                          *
17911           *    3. Move count bytes from the source to the target, checking       *
17912           *       that each byte is a valid uppercase letter, lowercase          *
17913           *       letter, or blank.  If an invalid character is found, then      *
17914           *       an illegal ascii digit trap is signaled.                       *
17915           *    4. Next subprogram instruction.                                   *
17916           *                                                                      *
17917           ************************************************************************
17918           *
17919     1291  MA   0020      ADDL      SPOA       000F SP1B ANDL      SP1B            BLANK FOR COMPARISON
17921                                                                                    IMMEDIATE OPERAND COUNT
17922     1292                ADD                            UBB  JSL  NSPG      ZERO  -
17924                                                                                    EXTENDED OPERAND COUNT
17925     1293                ADD                            SP1B ADD                  -
17927                                                                                    NUMBER OF ALPHABETICS TO MOVE
17928     1294                ADD                            UBB  JSL  EDI3      ZERO  -
17930                                                                                    LOOP UNTIL COUNT = 0
17931     1295                ADD                                 JSL  NSRC      UNC   -
17933                                                                                    FETCH NEXT SOURCE BYTE
17934     1296      0041      ADDL                  005A          ADDL                    "A" FOR UPPERCASE RANGE TEST
17936                                                                                    "Z" FOR LOWERCASE RANGE TEST
17937     1297      RH   UBA  SUB                   UBB  RH       JSBS *+4       TICB  RANGE TEST CHAR >= "A"
17939                                                                                    RANGE TEST "Z" >= CHAR
17940     1298      0061      ADDL                  007A          ADDL                    "a" FOR UPPERCASE RANGE TEST
17942                                                                                    "z" FOR LOWERCASE RANGE TEST
17943     1299      RH   UBA  SUB                   UBB  RH       JSBS *+2       TICB  RANGE TEST CHAR >= "a"
17945                                                                                    RANGE TEST "z" >= CHAR
17946     129A      SPOA RH   JSBS *+3       NZRO                ADD                     BLANKS ARE ALSO CONSIDERED ALPHABETIC
17948                                                                                    -
17949     129B                ADD                                 JSL  NTRG      UNC   -
17951                                                                                    STORE NEXT TARGET BYTE
17952     129C                JSB  *-8       UNC   FFFF SP1B ADDL      SP1B            NEXT ITERATION
17954                                                                                    DECREMENT COUNT
17955     129D                ADD                                 JSL  ETRP      UNC   -
17957                                                                                    INVALID ASCII DIGIT TRAP
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

17959           ***************************************************************************
17960           *                                                                         *
17961           *    MN/MNS/MFL                                                           *
17962           *       (Move Numerics,                                                   *
17963           *        Move Numerics Suppressed,                                        *
17964           *        Move with FLoating insertion)                                    *
17965           *                                                                         *
17966           *    1. Extract the immediate operand count from right nibble of          *
17967           *       the instruction.                                                  *
17968           *    2. If the immediate operand count is zero, then fetch the            *
17969           *       extended operand count in the next subprogram instruction         *
17970           *       byte.                                                             *
17971           *    3. If the significance trigger is on, then skip to step ???.         *
17972           *    4. Move leading blanks from the source to the target with            *
17973           *       zero filling if Move Numerics, or filling by the fill char        *
17974           *       if Move Numerics Suppressed or Move with FLoating                 *
17975           *       insertion.                                                        *
17976           *    5. Move leading zeros from the source to the target, with            *
17977           *       filling by the fill char if Move Numerics Suppressed or           *
17978           *       Move with FLoating Insertion.                                     *
17979           *    6. Insert the float char if Move with FLoating insertion.            *
17980           *                                                                         *
17981           ***************************************************************************
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	SKIP	COMMENT
17983		$NOWARN														
17984	129E	MN	0020		ADDL		XR5		000F	SP1B	ANDL		SP1B			BLANK FOR LEADING TEST
17986																IMMEDIATE OPERAND COUNT
17987	129F		0030		ADDL		XR6			UBB	JSL	NSPG			ZERO	ZERO FOR LEADING TEST
17989																EXTEND OPERAND COUNT
17990	12A0				ADD					SP1B	ADD					-
17992																NUMBER OF NUMERICS TO MOVE
17993	12A1				JSB	MN1		F3A		UBB	JSL	EDI3			ZERO	SIGNIFICANCE TRIGGER ALREADY SET?
17995																LOOP UNTIL COUNT = 0
17996	12A2				ADD						JSL	NSRC			UNC	-
17998																FETCH NEXT SOURCE BYTE
17999	12A3		RH	XR5	JSBS	*+6		NZRO	RA	SP4A	ADD	LSR			EVEN	LEADING BLANK?
18001																SUPPRESSED?
18002	12A4			XR6	ADD		RH			XR0	ADD		RH			ZERO FILL LEADING BLANK FOR MOVE-NUMERICS
18004																USE THE FILL CHARACTER IF SUPPRESSED
18005	12A5				ADD						JSL	NTRG			UNC	-
18007																STORE NEXT TARGET BYTE
18008	12A6				JSB	*-5		UNC	FFFF	SP1B	ADDL		SP1B			NEXT ITERATION
18010																DECREMENT COUNT
18011	12A7				ADD					UBB	JSL	EDI3			ZERO	-
18013																LOOP UNTIL COUNT = 0
18014	12A8				ADD						JSL	NSRC			UNC	-
18016																FETCH NEXT SOURCE BYTE
18017	12A9		RH	XR6	JSBS	*+4		NZRO	RA	SP4A	ADD	LSR			EVEN	LEADING ZERO?
18019																SUPPRESSED?
18020	12AA				ADD					XR0	ADD		RH			-
18022																USE FILL CHARACTER IF SUPPRESSED
18023	12AB				ADD						JSL	NTRG			UNC	-
18025																STORE NEXT TARGET BYTE
18026	12AC				JSB	*-5		UNC	FFFF	SP1B	ADDL		SP1B			NEXT ITERATION
18028																DECREMENT COUNT
18029	12AD		0004		ADDL						ADD					MFL OPCODE
18031																-
18032	12AE		UBA	SP4A	JSBS	*+6		NZRO			ADD					MFL?
18034																-
18035	12AF		RH		ADD		SP0A			XR1	ADD		RH			SAVE SOURCE CHARACTER
18037																FLOAT CHARACTER
18038	12B0				ADD						JSL	NTRG			UNC	-
18040																STORE FLOATING CHARACTER AT NEXT TARGET BYTE
18041	12B1		SP0A		ADD		RH				JSB	*+3			UNC	RESTORE SOURCE CHARACTER
18043																CONTINUE TRANSFER
18044		$WARN														

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18046         ************************************************************************
18047         *                                                                      *
18048         *    MN/MNS/MFL (continued)                                            *
18049         *                                                                      *
18050         *    Move the remaining bytes from the source to the target            *
18051         *    validating each ascii digit.  Signal an illegal ascii digit       *
18052         *    trap if the byte is not in the range ('0' -> '9').  Set the       *
18053         *    significance trigger since a non-blank character has been         *
18054         *    encountered in the source.                                        *
18055         *                                                                      *
18056         ************************************************************************
18057         *
18058  12B2                ADD                          UBB  JSL  EDI3           ZERO  -
18060                                                                                   LOOP UNTIL COUNT = 0
18061  12B3  MN1           ADD                               JSL  NSRC           UNC   -
18063                                                                                   FETCH NEXT SOURCE BYTE
18064  12B4                ADD            SF3A  0039         ADDL                      SET SIGNIFICANCE TRIGGER
18066                                                                                   "9" FOR COMPARISON TEST
18067  12B5        RH  XR6 JSBS *+3       NCRY  UBB  RH      JSBS *+3          NCRY  NUMERIC RANGE TEST CHAR >= "0"
18069                                                                                   NUMERIC RANGE TEST "9" >= CHAR
18070  12B6                ADD                               JSL  NTRG           UNC   -
18072                                                                                   STORE NEXT TARGET BYTE
18073  12B7                JSB  *-5       UNC   FFFF SP1B    ADDL      SP1B            NEXT ITERATION
18075                                                                                   DECREMENT COUNT
18076  12B8                ADD                               JSL  ETRP           UNC   -
18078                                                                                   INVALID ASCII DIGIT TRAP
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18080           ************************************************************************
18081           *                                                                      *
18082           *    IC/ICS (Insert Character / Insert Character Suppressed)           *
18083           *                                                                      *
18084           *    1. Extract the immediate operand count from the right nibble      *
18085           *       of the instruction byte.                                       *
18086           *    2. If the immediate operand count is zero, then fetch the         *
18087           *       extended operand count in the next subprogram instruction      *
18088           *       byte.                                                          *
18089           *    3. Fetch the insertion character from the next subprogram         *
18090           *       instruction byte.  If the significance trigger                 *
18091           *       is off and the instruction is Insert Character Suppressed,     *
18092           *       then use the fill character as the insertion character.        *
18093           *    4. Move the insertion character to the target count times.        *
18094           *                                                                      *
18095           ************************************************************************
18096           *
18097    12B9   IC            ADD                    000F SP1B ANDL      SP1B            -
18099                                                                                    IMMEDIATE OPERAND COUNT
18100    12BA                 ADD                         UBB  JSL  NSPG          ZERO  -
18102                                                                                    EXTENDED OPERAND COUNT
18103    12BB                 ADD                         SP1B ADD      SP2B            -
18105                                                                                    NUMBER OF CHARACTERS TO INSERT
18106    12BC                 ADD                              JSL  NSPG          UNC   -
18108                                                                                    FETCH INSERT CHARACTER
18109    12BD          SP4A JSB  *+2       ODD         XR0  ADD      RH                INSERT CHARACTER SUPPRESSED (IC/ICS)?
18111                                                                                    FILL CHARACTER FOR ICS
18112    12BE                 ADD            NF3A          ADD                          USE FILL CHARACTER IF SIG. TRIGGER IS 0
18114                                                                                    -
18115    12BF          SP1B ADD       RH                  SP2B ADD                      USE INSERT CHARACTER
18117                                                                                    LOOP COUNT
18118    12C0                 ADD                         UBB  JSL  EDI3          ZERO  -
18120                                                                                    LOOP UNTIL COUNT = 0
18121    12C1                 ADD                              JSL  NTRG          UNC   -
18123                                                                                    STORE NEXT TARGET BYTE
18124    12C2                 JSB  *-2       UNC   FFFF SP2B ADDL      SP2B            NEXT ITERATION
18126                                                                                    DECREMENT COUNT
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18128           ************************************************************************
18129           *                                                                      *
18130           *    ICI/ICSI                                                          *
18131           *            (Insert Characters Immediate,                             *
18132           *             Insert Characters Suppressed Immediate)                  *
18133           *                                                                      *
18134           *    1. Extract the immediate operand count from the right nibble      *
18135           *       of the instruction byte.                                       *
18136           *    2. If the immediate operand count is zero, then fetch the         *
18137           *       extended operand count in the next subprogram instruction      *
18138           *       byte.                                                          *
18139           *    3. Move the next count bytes from the subprogram to the           *
18140           *       target.  If Insert Characters Suppressed Immediate and         *
18141           *       the significance trigger is off, then use the fill             *
18142           *       character for each byte to be moved.                           *
18143           *                                                                      *
18144           ************************************************************************
18145           *
18146    12C3   ICI            ADD                   000F SP1B ANDL      SP1B            -
18148                                                                                    IMMEDIATE OPERAND COUNT
18149    12C4                  ADD                        UBB  JSL  NSPG          ZERO    -
18151                                                                                    EXTENDED OPERAND COUNT
18152    12C5                  ADD                        SP1B ADD       SP2B            -
18154                                                                                    NUMBER OF CHARACTERS TO INSERT
18155    12C6                  ADD                        UBB  JSL  EDI3          ZERO    -
18157                                                                                    LOOP UNTIL COUNT = 0
18158    12C7                  ADD                             JSL  NSPG          UNC     -
18160                                                                                    FETCH NEXT SUBPROGRAM IMMEDIATE BYTE
18161    12C8           SP4A JSB  *+2       ODD        XR0  ADD       RH                INSERT CHARACTER SUPPRESSED IMMEDIATE?
18163                                                                                    FILL CHARACTER FOR ICSI
18164    12C9                  ADD            NF3A             ADD                       USE FILL CHAR IF SIG. TRIGGER IS 0
18166                                                                                    -
18167    12CA           SP1B ADD       RH                      ADD                       USE INSERT IMMEDIATE CHARACTER
18169
18170    12CB                  ADD                             JSL  NTRG          UNC     -
18172                                                                                    STORE NEXT TARGET BYTE
18173    12CC                  JSB  *-6       UNC   FFFF SP2B ADDL      SP2B            NEXT ITERATION
18175                                                                                    DECREMENT COUNT
```

```
 RECORD   C.S          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18177            ************************************************************************
18178            *                                                                      *
18179            *    BRS  (BRanch if Significance trigger is set)                      *
18180            *                                                                      *
18181            *    1. Extract the immediate operand displacement from the right      *
18182            *       nibble of the instruction byte.                                *
18183            *    2. If the immediate operand displacement is zero, then fetch      *
18184            *       the extended operand displacment in the next subprogram        *
18185            *       instruction byte.                                              *
18186            *    3. If the significance trigger is not set, then do the            *
18187            *       next subprogram instruction.                                   *
18188            *    4. Determine the sign of the displacement and either add or       *
18189            *       subtract the displacement to or from the subprogram RBA.       *
18190            *    5. Restart the subprogram at the new subprogram RBA.              *
18191            *                                                                      *
18192            ************************************************************************
18193            *
18194     12CD   BRS  0080      ADDL      SP4A       000F SP1B ANDL      SP1B            DISPLACEMENT SIGN MASK
18196                                                                                    IMMEDIATE DISPLACEMENT OPERAND
18197     12CE                 ADD                        UBB  JSL  NSPG          ZERO  -
18199                                                                                    EXTENDED DISPLACEMENT OPERAND
18200     12CF                 ADD                        SP1B ADD                      -
18202                                                                                    DISPLACEMENT OPERAND
18203     12D0                 JSB  *+4       NF3A  UBB  SP4A AND                 ZERO  SIGNIFICANCE TRIGGER?
18205                                                                                    POSITIVE/NEGATIVE DISPLACEMENT
18206     12D1                 ADD                  FF00 SP1B IORL      SP1B            -
18208                                                                                    EXTEND SIGN BITS IF NEGATIVE
18209     12D2       RD        CAD                             ADD                     OLD SUBPROGRAM RBA
18211                                                                                    -
18212     12D3       UBA  SP1B ADD       RD                   JSL  EDI6          UNC   NEW SUBPROGRAM RBA
18214                                                                                    RESTART SUBPROGRAM AT NEW RBA
18215     12D4                 ADD                        JSL  EDI3          UNC   -
18217                                                                                    NEXT SUBPROGRAM INSTRUCTION
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18219           ************************************************************************
18220           *                                                                      *
18221           *    SUFS/SUFT                                                         *
18222           *             (SUbtract From Source pointer,                           *
18223           *              SUbtract From Target pointer)                           *
18224           *                                                                      *
18225           *    1. Extract the immediate operand displacement from the right      *
18226           *       nibble of the instruction byte.                                *
18227           *    2. If the immediate operand displacement is zero, then fetch      *
18228           *       the extended operand displacment in the next subprogram        *
18229           *       instruction.                                                   *
18230           *    3. If the extended displacement operand is zero, then do          *
18231           *       the next subprogram instruction.                               *
18232           *    4. Use SM+1 for upper stack limit to delay bounds testing at      *
18233           *       SM until the word is actually fetched or stored.               *
18234           *    5. Determine the sign of the displacement and extend the          *
18235           *       sign bits.                                                     *
18236           *                                                                      *
18237           *    SUFT (continued here)                                             *
18238           *                                                                      *
18239           *    6. Write the current target word and bounds test to flush         *
18240           *       the left byte from the buffer if needed.                       *
18241           *    7. Subtract the signed displacement from the target pointer       *
18242           *       RBA.                                                           *
18243           *    8. Compute the new target effective address, bounds test, and     *
18244           *       set the flags for target operations to be performed in ALUB.   *
18245           *    9. Next subprogram instruction.                                   *
18246           *                                                                      *
18247           ************************************************************************
```

RECORD NO	C.S. ADDR	ALU A LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	ALU B SPEC	ALU B SKIP	COMMENT
18249	12D5	SUFT	0080		ADDL		SP4A		000F	SP1B	ANDL		SP1B			DISPLACEMENT SIGN MASK
18251																IMMEDIATE DISPLACEMENT OPERAND
18252	12D6			SP4A	ADD		SP0A			UBB	JSL	NSPG			ZERO	SAVE SUBINSTRUCTION
18254																EXTENDED DISPLACEMENT OPERAND
18255	12D7				ADD					SP1B	JSL	EDI3			ZERO	-
18257																DISPLACEMENT OPERAND; DO NOTHING IF ZERO
18258	12D8			SM	INC		RG		UBB	SP4A	AND				ZERO	USE SM+1 TO TEST NEW RBAS BEFORE USAGE
18260																POSITIVE/NEGATIVE DISPLACEMENT?
18261	12D9		SP0A		JSB	SUFS		ODD	FF00	SP1B	IORL		SP1B			SUFT/SUFS?
18263																EXTEND DISPLACEMENT SIGN BITS IF NEGATIVE
18264	12DA				JSB	*+4		NF2	RF		INC			ROD		FINISH WRITING TARGET WORD IF LEFT BYTE
18266																FETCH CURRENT TARGET WORD
18267	12DB		RG	UBB	BNDG				SP3B		ADD					BOUNDS TEST TARGET WORD < SM+1
18269																LAST TARGET BYTE
18270	12DC		UBB		ADD	RLZ				OPB	ADD	RRZ				LEFT BYTE
18272																OLD RIGHT BYTE
18273	12DD				ADD				UBA	UBB	IOR			DATA		-
18275																WRITE TARGET WORD
18276	12DE		RC	SP1B	SUB	LSR			RC	SP1B	SUB		RC			NEW TARGET RBA/2
18278																NEW TARGET RBA
18279	12DF		UBA	DB	ADD		RH	ROD			JSBI	*+2			NF5B	NEW TARGET EFFECTIVE ADDRESS
18281																SKIP BOUNDS TEST IF SPLIT STACK
18282	12E0		UBA	DL	JSBS	CEBC		NCRY	RG	UBA	JSBS	CEBC			NCRY	BOUNDS TEST EA >= DL
18284																BOUNDS TEST SM >= EA
18285	12E1		RE		ADD			ROD	RH		ADD			ROD		REFETCH SOURCE OPERAND
18287																FETCH TARGET OPERAND
18288	12E2		RH		CAD		RF		RC		CSR			HBF2		SYNCHRONIZE TARGET OPERATIONS
18290																F2 <- EVEN/ODD SOURCE RBA
18291	12E3			XR1	ADD			ROB3		OPB	ADD	LRZ	SP3B			REFETCH TARGET OPERAND
18293																SAVE LEFT BYTE OF TARGET WORD
18294	12E4			SM	ADD		RG				JSL	EDI3			UNC	RESTORE SM FOR SOURCE/TARGET UPPER BOUND
18296																NEXT SUBPROGRAM INSTRUCTION

```
 RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18298            ************************************************************************
18299            *                                                                      *
18300            *    SUFS (continued here)                                             *
18301            *                                                                      *
18302            *    6. Subtract the signed displacement fro the source RBA            *
18303            *       pointer.                                                       *
18304            *    7. Compute the new source effective address, bounds test,         *
18305            *       and set the flags for the source operations to be              *
18306            *       performed in ALUA.                                             *
18307            *    8. Next subprogram instruction.                                   *
18308            *                                                                      *
18309            ************************************************************************
18310            *
18311    12E5    SUFS RB   SP1B SUB  LSR            RB   SP1B SUB       RB             NEW SOURCE RBA/2
18313                                                                                   NEW SOURCE RBA
18314    12E6         UBA  DB   ADD       RH   ROD            JSB  *+2            NF5B  NEW SOURCE EFFECTIVE ADDRESS
18316                                                                                   SKIP BOUNDS TESTS IF SPLIT STACK MODE
18317    12E7         UBA  DL   JSBS CEBC      NCRY RG   UBA  JSBS CEBC           NCRY  BOUNDS TEST EA >= DL
18319                                                                                   BOUNDS TEST SM >= EA.
18320    12E8         RB        ADD            ODD  RH        ADD       RE   SF1       EVEN / ODD SOURCE RBA?
18322                                                                                   F1 <- ODD SOURCE RBA DEFAULT
18323    12E9              SM   ADD       RG   CF1            JSL  EDI3           UNC   F1 <- EVEN SOURCE RBA; RESTORE SM LIMIT
18325                                                                                   NEXT SUBPROGRAM INSTRUCTION
```

```
RECORD  C S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18327           ************************************************************************
18328           *                                                                      *
18329           *     ICP/ICPS                                                         *
18330           *              (Insert Character Punctuation,                          *
18331           *               Insert Character Punctuation Suppressed)               *
18332           *                                                                      *
18333           *     1.  Extract the immediate operand character from the right       *
18334           *         nibble of the subprogram instruction.                        *
18335           *     2. The character to be inserted is obtained by adding %40        *
18336           *        to the immediate operand.                                     *
18337           *     3. If the instruction is Insert Character Punctuation            *
18338           *        Suppressed and the significance trigger is false, then        *
18339           *        the character to be inserted is the fill char.                *
18340           *     4. Store the insert character in the next target byte.           *
18341           *     5. Next subprogram instruction.                                  *
18342           *                                                                      *
18343           ************************************************************************
18344           *
18345    12EA   ICP            ADD                    000F SP1B ANDL                    -
18347                                                                                    IMMEDIATE INSERT CHARACTER OPERAND
18348    12EB            SP4A JSB  *+2        EVEN   0020 UBB  ADDL      SP1B          INSERT CHARACTER PUNCTUATION SUPPRESSED?
18350                                                                                    ADD A BLANK TO THE INSERT CHARACTER
18351    12EC                 ADD             NF3A        XR0  ADD       RH            USE THE FILL CHARACTER IF SUPPRESSED
18353                                                                                    FOR ICPS USE THE FILL CHARACTER DEFAULT
18354    12ED            SP1B ADD       RH                     ADD                     USE THE INSERT CHARACTER
18356                                                                                    -
18357    12EE                 ADD                                JSL  NTRG          UNC  -
18359                                                                                    STORE THE NEXT TARGET BYTE
18360    12EF                 ADD                                JSL  EDI3          UNC  -
18362                                                                                    NEXT SUBPROGRAM INSTRUCTION
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18364           ***************************************************************
18365           *                                                             *
18366           *    IS (Insert characters depending on Sign)                 *
18367           *                                                             *
18368           *    1. Extract the immediate operand count from the right nibble *
18369           *       of the instruction byte.                              *
18370           *    2. If the count is zero, then next subprogram instruction. *
18371           *    3. If the condition code is CCG or CCE, then move the next *
18372           *       count characters from the subprogram to the target and *
18373           *       add the count to the subprogram RBA to skip the second *
18374           *       string in the subprogram.  Restart the subprogram at the *
18375           *       new subprogram RBA.                                   *
18376           *    4. If the condition code is CCL, then add the count to the *
18377           *       subprogram RBA to skip over the first string in the   *
18378           *       subprogram and move the next count subprogram characters *
18379           *       to the target.                                        *
18380           *    5. Next subprogram instruction.                          *
18381           *                                                             *
18382           ***************************************************************
18383           *
18384    12F0   IS   0100 STA  ANDL      SPOA       000F SP1B ANDL      SP2B          CCL OR CCG/CCE?
18386                                                                              IMMEDIATE OPERAND COUNT
18387    12F1        UBB      ADD       SP4A            UBB  JSL  EDI3          ZERO  SAVE COUNT
18389                                                                              DONE IF COUNT IS ZERO
18390    12F2        SPOA     JSB  *+6       NZRO       UBB  JSB  *+5           ZERO  POSITIVE OR NEGATIVE?
18392                                                                              DONE WHEN COUNT = 0
18393    12F3                 ADD                                 JSL  NSPG      UNC   -
18395                                                                              FETCH NEXT SOURCE BYTE
18396    12F4                 ADD                            SP1B ADD       RH          -
18398                                                                              TRANSFER SUBPROGRAM BYTE TO TARGET
18399    12F5                 ADD                                 JSL  NTRG      UNC   -
18401                                                                              WRITE NEXT TARGET WORD
18402    12F6                 JSB  *-4       UNC   FFFF SP2B ADDL      SP2B          NEXT ITERATION
18404                                                                              DECREMENT COUNT
18405    12F7        RD   SP4A ADD       RD                  JSL  EDI6          ZERO  SKIP SECOND OPERAND STRING
18407                                                                              START AT NEW SUBPROGRAM RBA
18408    12F8        RD   SP4A ADD       RD                  JSL  SPGS          ZERO  SKIP FIRST OPERAND STRING
18410                                                                              UPDATE THE SUBPROGRAM POINTERS
18411    12F9                 ADD                            SP2B ADD                  -
18413                                                                              COUNT
18414    12FA                 ADD                            UBB  JSL  EDI3          ZERO  -
18416                                                                              DONE WHEN COUNT = 0
18417    12FB                 ADD                                 JSL  NSPG      UNC   -
18419                                                                              FETCH NEXT SUBPROGRAM BYTE
18420    12FC                 ADD                            SP1B ADD       RH          -
18422                                                                              TRANSFER SUBPROGRAM BYTE TO TARGET
18423    12FD                 ADD                                 JSL  NTRG      UNC   -
18425                                                                              STORE NEXT TARGET BYTE
18426    12FE                 JSB  *-4       UNC   FFFF SP2B ADDL      SP2B          NEXT ITERATION
18428                                                                              DECREMENT COUNT
18429    12FF                 ADD                                 ADD               NOP to keep system happy
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18432           ************************************************************************
18433           *    Duplicate A (DUP) Instruction                                     *
18434           ************************************************************************
18435           *
18436           %1300
18437           *
18438           $LUT INSTR=DUP:0 000 100 101 xxx xxx, SR=1, ENTRY=DUP
18439           *
18440   1300    DUP            ADD                          JSZ  PSHM          SR7   ; JSB to empty RG if SR = 7
18442   1301                   ADD            EPSH      RA  ADD       RH   CCA       Push 1; New (S) := (S)
18444   1302                   ADD            NEXT          ADD                      NEXT;
18446           *
18447           ************************************************************************
18448           *    DDUP (Double duplicate) Instruction                               *
18449           ************************************************************************
18450           *
18451           $LUT INSTR=DDUP:0 000 100 110 xxx xxx, SR=2, ENTRY=DDUP
18452           *
18453   1303    DDUP           JSZ  PSHM      SRG5          JSZ  PSM2          SR7   Empty 2 TOS if SR = 7 else empty 1 if SR = 6
18455   1304                   ADD            EPSH      RB  ADD       RH   CCA       Push; New (S-1) := (S-1), CCA
18457   1305                   JSZ  NEXT      EPSH      RA  ADD       RG   DCC       JSB for NEXT, push 1; New (S) := (S), DCC
```

```
RECORD   C S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18460          ************************************************************************
18461          *                                                                      *
18462          *    NSRC (fetch Next SouRCe byte)                                     *
18463          *                                                                      *
18464          *    1. Extract the left byte from the current source word and         *
18465          *       increment the source RBA.  Bounds test the source              *
18466          *       effective address.                                             *
18467          *    2. If the right byte flag is off then set the right byte          *
18468          *       flag on and return.                                            *
18469          *    3. Extract the right byte from the current source word,           *
18470          *       bounds test the source effective address, and clear the        *
18471          *       right byte flag.                                               *
18472          *    4. Increment the source effective address and fetch.              *
18473          *    5. Return.                                                        *
18474          *                                                                      *
18475          *    NTRG (store Next TaRGet byte)                                     *
18476          *                                                                      *
18477          *    1. Increment the target RBA and save the byte in case it is       *
18478          *       the left byte.                                                 *
18479          *    2. If the right byte flag is off, then set the right byte         *
18480          *       flag and return.                                               *
18481          *    3. Increment the target effective address, combine the left       *
18482          *       byte that was previously saved with the right byte and         *
18483          *       store the next target word.                                    *
18484          *    4. Bounds test the target effective address, clear the right      *
18485          *       byte flag.                                                     *
18486          *    5. Return.                                                        *
18487          *                                                                      *
18488          ************************************************************************
18489          *
18490          *
18491   1306   NSRC      OPA  ADD  LRZ  RH         RG   RE   BNDE                F1    EXTRACT LEFT SOURCE BYTE
18493                                                                                   LEFT OR RIGHT SOURCE BYTE?  (1002)
18494   1307       RB        INC       RB                   ADD            SF1   RSB   INCREMENT SOURCE RBA
18496                                                                                   REMOVED BOUNDS TO ABOVE LINE          (1002 )
18497   1308            OPA  ADD  RRZ  RH         RG   RE   BNDE           CF1         EXTRACT RIGHT SOURCE BYTE
18499                                                                                   -
18500   1309       RE        INC       RE   ROD            ADD                RSB   INCREMENT SOURCE EFFECTIVE ADDRESS AND FETCH
18502                                                                                   RETURN
18503          "
18504          "
18505   130A   NTRG RC        INC       RC   F2   RH        ADD       SP3B         INCREMENT TARGET RBA
18507                                                                                   SAVE LEFT BYTE
18508   130B                   ADD            RSB  SP3B      ADD  RLZ       SF2     RETURN IF LEFT BYTE
18510                                                                                   PREV LEFT BYTE
18511   130C       RF        INC       RF        UBB  RH    IOR            DATA    INCREMENT TARGET EFFECTIVE ADDRESS
18513                                                                                  WRITE TARGET WORD
18514                                                                                  TICB TO SLOW DOWN RSB TO AVOID S KILL(1000 )
18515   130D       RG   UBA  BNDE           CF2              ADD                     (1010)
18517   130E                 ADD                             ADD                RSB   BOUNDS TEST TARGET EFFECTIVE ADDRESS <= SM
18519                                                                                  RETURN
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18521           ************************************************************************
18522           *                                                                      *
18523           *    EDI5 (extended subprogram instruction decoding)                   *
18524           *                                                                      *
18525           *    1.  Extract the subprogram opcode contained in the right          *
18526           *        nibble of the subprogram instruction.                         *
18527           *    2.  Use the subprogram opcode as an index into the branch         *
18528           *        table.                                                        *
18529           *    3.  Range check the valid subprogram opcodes.                     *
18530           *    4.  Jump into the brach table to jump to the correct              *
18531           *        subprogram instruction.                                       *
18532           *                                                                      *
18533           ************************************************************************
18534           *
18535    130F   EDI5 000F SP1B ANDL      SP4A                ADD                       MASK FOR SUB-OPCODE
18537                                                                                  -
18538    1310        UBA       ADD  LSR             EDT2    ADDL LBL                  BRANCH TABLE INDEX
18540                                                                                  BRANCH TABLE BASE
18541    1311        UBA  UBB  ADD            WRX3  0009    ADDL                      INDEXED BRANCH TABLE ADDRESS
18543                                                                                  MAXIMUM VALID SUB-OPCODE
18544    1312                  ADD            RAR   UBB  SP4A JSBS EDI3          NCRY  TRANSFER TABLE ADDRESS TO RAR
18546                                                                                  SKIP BRANCH TABLE IF NOT VALID
18547    1313                  ADD                        ADD                       -
18549                                                                                  -
18550    1314                  ADD            RSB         ADD                       RANDOM TABLE BRANCH
18552                                                                                  -
18553           *
18554           *
18555           *    RIGHT NIBBLE OPCODE BRANCH TABLE
18556           *
18557           *
18558    1315   EDT2       SP4A JSB  TE        EVEN       SP4A JSB  ENDF        ODD   TERMINATE EDIT
18560                                                                                  END FLOATING INSERTION
18561    1316              SP4A JSB  SST1      EVEN       SP4A JSB  SST0        ODD   SET SIGNIFICANCE TRIGGER TO 1
18563                                                                                  SET SIGNIFICANCE TRIGGER TO 0
18564    1317              SP4A JSB  MDWO      EVEN       SP4A JSB  SFC         ODD   MOVE DIGIT WITH OVERPUNCH
18566                                                                                  SET FILL CHARACTER
18567    1318              SP4A JSB  SFLC      EVEN       SP4A JSB  DFLC        ODD   SET FLOAT CHARACTER DEPENDING ON SIGN
18569                                                                                  DEFINE FLOAT CHARACTER DEPENDING ON SIGN
18570    1319              SP4A JSB  SETC      EVEN       SP4A JSB  DBNZ        ODD   SET LOOP COUNT
18572                                                                                  DECREMENT LOOP COUNT AND BRANCH IF NON-ZERO
```

```
 RECORD   C S        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18574            *************************************************************************
18575            *                                                                       *
18576            *    EDI6 (next subprogram instruction at new RBA)                      *
18577            *                                                                       *
18578            *    Set up the subprogram pointers and flags for the new               *
18579            *    subprogram RBA.  Continue with the interrupt test.                 *
18580            *                                                                       *
18581            *    EDI3 (test for interrupts between subprogram instructions)         *
18582            *                                                                       *
18583            *    If interrupts are not pending, then do the next subprogram         *
18584            *    instruction.  Otherwise flush the last target byte from the        *
18585            *    buffer if needed.  Add two words to the interrupt marker           *
18586            *    containing the significance trigger, loop count, float char,       *
18587            *    and fill char.  Leave a -1 on top of the stack so the              *
18588            *    instruction will be resumed after the interrupt.                   *
18589            *                                                                       *
18590            *    Go to the interrupt handler.                                       *
18591            *                                                                       *
18592            *************************************************************************
18593            *
18594     131A   EDI6          ADD                                 JSL  SPGS          UNC   -
18596                                                                                        RESTART SUBPROGRAM AT NEW RBA
18597     131B   EDI3          JSB  *+2       TEST                 ADD                      INTERRUPTS PENDING?
18599                                                                                        -
18600     131C                 ADD                                 JSL  EDI4          UNC   NEXT SUBPROGRAM INSTRUCTION
18602                                                                                        -
18603            *
18604            *
18605            *    INTERRUPT PENDING, FINISH LAST TARGET WORD, THEN INTERRUPT
18606            *
18607            *
18608     131D                 JSB  *+4       NF2  RF              INC                ROD       FINISH WRITING TARGET WORD IF LEFT BYTE
18610                                                                                        FETCH CURRENT TARGET WORD
18611     131E       RG   UBB  BNDE                                 SP3B             ADD       BOUNDS TEST LAST TARGET WORD
18613                                                                                        LAST TARGET BYTE
18614     131F       UBB       ADD  RLZ                                 OPB  ADD  RRZ        LEFT BYTE
18616                                                                                        OLD RIGHT BYTE
18617     1320                 ADD                                 UBA  UBB  IOR           DATA      -
18619                                                                                        WRITE LAST TARGET WORD
18620            *
18621            *
18622            *    SET UP INTERRUPT RETURN STACK MARKER
18623            *
18624            *
18625     1321            XR0  ADD  RRZ  RH   NF3A  8000      ADDL                       SAVE LOOP COUNT
18627                                                                                        SIGNIFICANCE TRIGGER MASK
18628     1322       UBA  UBB  IOR       RH                        XR0  ADD  RLZ       EPSH      SAVE LOOP COUNT AND SIGNIFICANCE TRIGGER
18630                                                                                        FILL CHARACTER
18631     1323                 ADD                                 UBB  XR1  IOR      RA   EPSH      -
18633                                                                                        SAVE FILL AND FLOAT CHARACTERS
18634     1324                 JSZ  IRDN      UNC                  CAD      RG                OFF TO INTERRUPT HANDLER
18636                                                                                        RESTART FLAG
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18638           ************************************************************************
18639           *                                                                      *
18640           *    ETRP (Edit TRaP handling for invalid data)                        *
18641           *                                                                      *
18642           *    Signal an invalid ascii digit trap.                               *
18643           *                                                                      *
18644           *    TE (Terminate Edit)                                               *
18645           *                                                                      *
18646           *    Flush the last target byte if needed. Pop the four TOS            *
18647           *    registers.  Test for an invalid ascii digit trap.                 *
18648           *                                                                      *
18649           *    Next instruction.                                                 *
18650           *                                                                      *
18651           ************************************************************************
18652           *
18653    1325   ETRP           ADD                              ADD            SF4B
18655                                                                                    INVALID ASCII DIGIT TRAP
18656    1326   TE             JSB  *+4       NF2   RF          INC            ROD
18658                                                                                    FETCH LAST TARGET WORD
18659    1327          RG  UBB  BNDE                  SP3B      ADD
18661                                                                                    LAST TARGET BYTE FOR LEFT BYTE
18662    1328          UBB      ADD  RLZ                  OPB   ADD  RRZ
18664                                                                                    OLD RIGHT BYTE
18665    1329                   ADD                       UBA UBB IOR            DATA
18667                                                                                    WRITE LAST TARGET WORD
18668    132A                   ADD       RA                   ADD            EPP4
18670                                                                                    POP ALL TOS PARAMETERS
18671    132B                   ADD                            JSZ  TRPA           F4B
18673                                                                                    ILLEGAL ASCII DIGIT TRAP
18674    132C                   ADD                            ADD                 NEXT
18676                                                                                    NEXT INSTRUCTION
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18678           ************************************************************************
18679           *                                                                      *
18680           *    ENDF (END Floating insertion)                                     *
18681           *                                                                      *
18682           *    If the significance trigger hasn't been already been set,         *
18683           *    then insert the floating character at the next target byte.       *
18684           *                                                                      *
18685           *    Next subprogram instruction.                                      *
18686           *                                                                      *
18687           ************************************************************************
18688           *
18689    132D   ENDF          JSB  *+2       F3A        XR1  ADD       RH             NO ACTION TAKEN IF SIGNIFICANCE TRIGGER IS 0
18691                                                                                  FLOAT CHARACTER
18692    132E                ADD                            JSL  NTRG          UNC   -
18694                                                                                  INSERT FLOATING CHARACTER
18695    132F                ADD                            JSL  EDI3          UNC   -
18697                                                                                  NEXT SUBPROGRAM INSTRUCTION
```

```
 RECORD  C.S        ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18699          ************************************************************************
18700          *                                                                      *
18701          *    SST1 (Set the significance Trigger to 1)                          *
18702          *                                                                      *
18703          *    Do what it says, then do the next subprogram instruction.         *
18704          *                                                                      *
18705          *    SST0 (Set the Significance Trigger to 0)                          *
18706          *                                                                      *
18707          *    Do what is says, then do the next subprogram instruction.         *
18708          *                                                                      *
18709          ************************************************************************
18710          *
18711    1330  SST1           ADD            SF3A           JSL  EDI3          UNC   SET SIGNIFICANCE TRIGGER TO 1
18713                                                                                 NEXT SUBPROGRAM INSTRUCTION
18714    1331  SST0           ADD            CF3A           JSL  EDI3          UNC   SET SIGNIFICANCE TRIGGER TO 0
18716                                                                                 NEXT SUBPROGRAM INSTRUCTION
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18718          ********************************************************************
18719          *                                                                  *
18720          *    MDWO (Move Digit With Overpunch)                              *
18721          *                                                                  *
18722          *    1. Fetch the next source byte.                                *
18723          *    2. If the significance trigger is false, then zerofill the    *
18724          *       byte if the byte is a leading blank.                       *
18725          *    3. Trap if the byte is not a legal ascii digit ('0'-'9').     *
18726          *    4. Set the sign flag used for overpunching based on the       *
18727          *       condition code in the STATUS register.                     *
18728          *    5. Overpunch the byte and store it as the next target byte.   *
18729          *    6. Next subprogram instruction.                               *
18730          *                                                                  *
18731          ********************************************************************
18732     1332 MDWO 0020       ADDL      SP4A                 JSL  NSRC          UNC   BLANK FOR COMPARISON
18734                                                                                    FETCH NEXT SOURCE BYTE
18735     1333                 JSB  *+2       F3A   RH   SP4A SUB                NZRO  NO BLANK FILL IF SIGNIFICANCE TRIGGER SET
18737                                                                                    ZERO FILL REQUIRED IF SOURCE BYTE IS BLANK
18738     1334                 ADD                  0030      ADDL      RH              -
18740                                                                                    ZERO FILL THE LEADING BLANK
18741     1335      0030       ADDL      SP4A       0039      ADDL                      "0" FOR ASCII DIGIT TEST              (2307)
18743                                                                                    "9" FOR ASCII DIGIT TEST
18744     1336      RH   UBA   JSBS ETRP      NCRY  UBB  RH   JSBS ETRP          NCRY  INVALID ASCII DIGIT TEST              (2307)
18746                                                                                    INVALID ASCII DIGIT TEST
18747     1337      RH   SP4A  SUB       SP4A       0100      ADDL                      SP4A=DIGIT-30H (ZERO IF ASCII '0')    (2307)
18749                                                                                    CCL MASK FOR STATUS REGISTER
18750     1338      UBB  STA   AND             ZERO            ADD           SF2       POSITIVE OR NEGATIVE OVERPUNCH?
18752                                                                                    POSITIVE OVERPUNCH
18753     1339                 ADD               CF2            JSL  OVPN          UNC   NEGATIVE OVERPUNCH
18755                                                                                    OVERPUNCH SOURCE BYTE
18756     133A      RC         CSR               HBF2           JSL  NTRG          UNC   RESTORE TARGET RBA FLAG
18758                                                                                    STORE NEXT TARGET BYTE
18759     133B                 JSB  EDI3      F3A                ADD                     NEXT SUBPROGRAM INSTRUCTION IF SIG FLAG SET
18761                                                                                    -                                     (2307)
18762     133C                 JSB  EDI3      UNC        SP4A JSB  SST1          NZRO  SET SIGNIFICANCE FLAG IF CURRENT DIGIT IS
18764                                                                                    NONZERO--ELSE DO NEXT SUBINSTRUCTION  (2307)
```

```
 RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18766            ***************************************************************************
18767            *                                                                         *
18768            *    SFC (Set the Fill Character)                                         *
18769            *                                                                         *
18770            *    1. Fetch the next subprogram byte.                                   *
18771            *    2. Set the fill char to that byte.                                   *
18772            *    3. Next subprogram instruction.                                      *
18773            *                                                                         *
18774            ***************************************************************************
18775            *
18776     133D   SFC            ADD                              JSL  NSPG           UNC  -
18778                                                                                      FILL CHAR IS IN NEXT SUBPROGRAM BYTE
18779     133E                  ADD                         SP1B JSL  EDI3 XR0       UNC  -
18781                                                                                      NEXT SUBPROGRAM INSTRUCTION; SET FILL CHAR
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18783           ************************************************************************
18784           *                                                                      *
18785           *    SFLC (Set the FLoating Character)                                 *
18786           *                                                                      *
18787           *    1. Fetch the next subprogram byte containing the positive and     *
18788           *       and negative sign values.                                      *
18789           *    2. Determine the condition code contained in the STATUS           *
18790           *       register and extract the left  nibble if CCG/CCE or the        *
18791           *       right nibble if CCL.                                           *
18792           *    3. Add %40 to the nibble to obtain the ascii sign and set         *
18793           *       the floating character to this value.                          *
18794           *    4. Next subprogram instruction.                                   *
18795           *                                                                      *
18796           *    DFLC (Define the FLoating Character)                              *
18797           *                                                                      *
18798           *    1. Fetch the next subprogram byte containing the positive         *
18799           *       floating character.                                            *
18800           *    2. Fetch the following subprogram byte containing the negative    *
18801           *       floating character.                                            *
18802           *    3. Determine the condition code contained in the STATUS           *
18803           *       register and set the floating character to the positive        *
18804           *       floating character if CCG/CCE or the negative floating         *
18805           *       character if CCL.                                              *
18806           *    4. Next subprogram instruction.                                   *
18807           *                                                                      *
18808           ************************************************************************
18809           *
18810   133F    SFLC 0100 STA  ANDL      SP0A                JSL  NSPG           UNC   CCL OR CCG/CCE?
18812                                                                                     FETCH BYTE CONTAINING FLOATING CHAR NIBBLES
18813   1340         SP0A      JSB  *+4        NZRO  000F SP1B ANDL      SP2B          POSITIVE OR NEGATIVE?
18815                                                                                     FETCH RIGHT NIBBLE IF NEGATIVE
18816   1341                   ADD                   00F0 SP1B ANDL                    -
18818                                                                                     EXTRACT LEFT NIBBLE IN PLACE
18819   1342         0004      ADDL                  UBB       REPC                    SHIFT COUNT TO EXTRACT THE LEFT NIBBLE
18821                                                                                     REPEAT THE FOLLOWING LINE
18822   1343         UBA       CAD             ZERO  UBB       ADD  LSR  SP2B          DONE WHEN COUNT = 0
18824                                                                                     SHIFT LEFT NIBBLE TO RIGHT NIBBLE
18825   1344                   ADD                   0020 SP2B ADDL      XR1           -
18827                                                                                     FLOAT CHAR := NIBBLE + BLANK
18828   1345                   ADD                        JSL  EDI3           UNC       -
18830                                                                                     NEXT SUBPROGRAM INSTRUCTION
18831   1346   DFLC 0100 STA  ANDL      SP4A                JSL  NSPG           UNC   CCL OR CCG/CCE?
18833                                                                                     FETCH BYTE CONTAINING POSITIVE FLOAT CHAR
18834   1347                   ADD                   SP1B JSL  NSPG XR1          UNC   -
18836                                                                                     FETCH BYTE CONTAINING NEGATIVE FLOAT CHAR
18837   1348                   ADD                   SP4A ADD                    ZERO  -
18839                                                                                     DOES STATUS REGISTER INDICATE NEGATIVE(CCL)
18840   1349                   ADD                   SP1B ADD       XR1                -
18842                                                                                     NEGATIVE FLOAT CHAR
18843   134A                   ADD                        JSL  EDI3           UNC       -
18845                                                                                     NEXT SUBPROGRAM INSTRUCTION
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18847           **************************************************************************
18848           *                                                                        *
18849           *     SETC (Set the loop Count)                                          *
18850           *                                                                        *
18851           *     1. Fetch the next subprogram byte and assign it to the             *
18852           *        loop count.                                                     *
18853           *     2. Next subprogram instruction.                                    *
18854           *                                                                        *
18855           **************************************************************************
18856           *
18857    134B   SETC            ADD                              JSL  NSPG            UNC   -
18859                                                                                        FETCH SUBPROGRAM BYTE CONTAINING LOOP COUNT
18860    134C              SP1B ADD       XR0                    JSL  EDI3            UNC   STORE LOOP COUNT
18862                                                                                        NEXT SUBPROGRAM INSTRUCTION
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18864           ************************************************************************
18865           *                                                                      *
18866           *    DBNZ (Decrement and Branch if Non-Zero)                           *
18867           *                                                                      *
18868           *    1. Fetch the next subprogram byte containing the branch           *
18869           *       displacement.                                                  *
18870           *    2. Decrement the loop count.                                      *
18871           *    3. If the loop count is zero, then do the next subprogram         *
18872           *       instruction.                                                   *
18873           *    4. Extend the sign of the displacement.                           *
18874           *    5. Subtract the displacement from the subprogram RBA.  An         *
18875           *       extra byte displacement is also subtracted since the           *
18876           *       displacement is relative to the address of the byte            *
18877           *       containing the displacement, but the RBA now points to         *
18878           *       the byte after the displacement.                               *
18879           *    6. Next subprogram instruction after updating the subprogram      *
18880           *       pointers, flags, etc.                                          *
18881           *                                                                      *
18882           ************************************************************************
18883           *
18884    134D   DBNZ      XR0   ADD       SP0A                JSL  NSPG          UNC   LOOP COUNT
18886                                                                                    FETCH BRANCH DISPLACEMENT
18887    134E         SP0A      CAD       XR0       0080 SP1B ANDL                     DECREMENT LOOP COUNT
18889                                                                                    MASK SUBPROGRAM DISPLACEMENT SIGN BIT
18890    134F         UBA       JSB  *+4       ZERO  UBB       ADD                ZERO  NO ACTION IF LOOP COUNT IS NOW ZERO
18892                                                                                    POSITIVE OR NEGATIVE DISPLACEMENT
18893    1350                   ADD                 FF00 SP1B IORL      SP1B           -
18895                                                                                    EXTEND SIGN BIT IF NEGATIVE
18896    1351                   ADD                           ADD                      -
18898                                                                                    -
18899    1352         RD   SP1B CAD       RD                  JSL  EDI6          UNC   NEW SUBPROGRAM RBA
18901                                                                                    RESTART SUBPROGRAM AT NEW RBA
18902    1353                   ADD                           JSL  EDI3          UNC   -
18904                                                                                    NEXT SUBPROGRAM INSTRUCTION
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18906           ************************************************************************
18907           *          DPPX is the entry point for the PSEB, PSDB, DISP and        *
18908           *          XCHD instructions.                                          *
18909           ************************************************************************
18910           *
18911           $LUT INSTR=DISP/PSE&DB/XCHD:0 011 000 000 11, DSPL=4, ENTRY=DPPX, F2
18912           *
18913     1354  DPPX 0580       ADDL      SP4A                 JSZ  TRP6 BKX3       NPRV  SP4A := addr of QI lor mask for #DISP;
18915                                                                                       BKX3 := 0, trap if not privileged
18916     1355            DSPL JSB  XCHD       ZERO       UBA  ADD  LRZ        ROX3       JSB if XCHD instruction; Read QI (UBB = 5)
18918     1356            UBB  ADD             ROX3  FFEE      ADDL                      Read QI; UBB := -18
18920     1357            CIR  JSB  DISP       EVEN  UBB  OPB  ADD               ROX3       JSB if DISP instruction; Read (QI-18)
18922     1358            OPA  ADD             ROB3       UBA  ADD  LSR        SF4B ODD   Read (QI); Skip if PSEB instruction, set F4
18924     1359  PSDB 1100 CPX2 ANDL      SP4A                 OPB  INC          DATA NEXT  SP4A := UBA (for #DISP) := ICS & DISP flags
18926                                                                                       (on ICS if in DISPATCHER);
18927                                                                                       increment (QI-18), NEXT if PSDB instr
18928     135A  PSEB UBB  DSPL SUB             NCRY       SREG JSZ  PSEH            ZERO  Skip if (QI-18) < 2 (DSPL = 3);
18930                                                                                       Trap if (QI-18) = 0
18931     135B                 JSZI NEXT       CCA   UBB        CAD               DATA       Set CCG and NEXT if (QI-18) >= 2;
18933                                                                                       Decrement (QI-18)
18934     135C            CPX2 ADD  SWAB       EVEN       SP4A ADD               CCA  NZRO  Skip if not in DISPATCHER;
18936                                                                                       Skip if on ICS, set CCE if not
18937     135D                 ADD             DATA       OPB  JSZ  INT9            NEG   (QI) := 0 if in DISPATCHER;
18939                                                                                       Start DISPATCHER if not on ICS and
18940                                                                                       (QI).(0:1) F4 causes JSB to #IXT4 from
18941                                                                                       #INT9. (QI) is left in OPA for #IXT4
18942     135E                 INC             CCA                ADD               NEXT  Set CCG; NEXT
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18945          ************************************************************************
18946          *          Dispatch (DISP) Instruction                                 *
18947          ************************************************************************
18948          *
18949    135F  DISP      OPA  ADD            WRX3  UBA  OPB  IOR             CCA  ZERO  Write at (QI);
18951                                                                                   Skip if not on ICS and (QI-18) = 0, set CCE
18952    1360            SP4A ADD  RLZ       DATA             INC             CCA  NEXT  (QI) := !8000; Set CCG, NEXT if on ICS or
18954                                                                                     DISABLED
18955    1361            OPA  JSZ  INT9      ROX3             ADD             SF4B       Start DISPATCHER if not on ICS and ENABLED,
18957                                                                                     Reread (QI) for #IXT4;
18958                                                                                     F4 causes JSB to #IXT4 from #INT9
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18960            ************************************************************************
18961            *    XCHD (Exchange DB) Instruction - The entry point for XCHD         *
18962            *    is #DPPX, for which the SR preadjust is 0.  The instruction       *
18963            *    checks for two or more TOS registers full, and fetches them       *
18964            *    from memory if not.  The split bank flag is set or cleared        *
18965            *    according to the new value of DB-BANK.                            *
18966            ************************************************************************
18967            *
18968     1362   XCHD       RB    JSZ  PULM       SRL2        BNKD JSZ  PUL2 SP1B       SRZ   Pull two TOS if SR = 0 else pull one TOS if
18970                                                                                           SR = 1, UBA := new bank #; SP1B := DB-BANK
18971     1363         3FFF       ADDL                  UBA   BNKS XOR                  ZERO  UBA := pre-mask for CCPX;
18973                                                                                           Skip if new DB-BANK = S-BANK
18974     1364         1000       ADDL                  UBA         INC             CCPX UNC  UBA := pre-mask for CCPX;
18976                                                                                           Set FSS (UBB = !4000), skip
18977     1365               SP1B ADD        RB          UBA   UBB  IOR             CCPX       (S-1) := old DB-BANK; Clear FSS if
18979                                                                                           DB-BANK = S-BANK (UBB = !5000)
18980     1366               DB   ADD        RA          RA    RB   XFRR       DB             (S) := old DB, DB := new DB from (S)
18982     1367                    ADD             NEXT  0OFF SREG ANDL       BNKD           NEXT; BNKD := new DB-BANK from (S-1)
```

```
 RECORD   C.S.        ************ ALU A ************  ************** ALU B ***************
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

18985            ***********************************************************************
18986            *                                                                     *
18987            *     Set Disable/Enable Interrupts Bit (SED) instruction             *
18988            *                                                                     *
18989            *     This instruction will reset/set the interrupts enabled bit      *
18990            *     in the status register (bit 1) depending upon CIR.(15:1).       *
18991            *                                                                     *
18992            ***********************************************************************
18993            *
18994            $LUT INSTR=SED:0 011 000 000 10x xxx, DSPL=4, ENTRY=SED
18995            *
18996  1368  SED       CIR  JSB  SED1      EVEN           JSZ  TRP6          NPRV  JSB if DISABLE; Trap if not privileged
18998  1369            CPX2 JSB  SED2      ODD   4000     ADDL      RH             JSB if DINTFF; UBB := RH := mask to set ints
19000  136A      UBB   STA  IOR       STA                 JSZC NEXT          UNC   STA := STA with interrupts set; UBB := mask
19002                                                                                 to clear interrupts; JSB for NEXT
19003  136B  SED1 UBB   STA  AND       STA                 JSZ  NEXT          UNC   STA := STA with interrupts cleared; NEXT
19005  136C  SED2 RH    STA  IOR       STA                 JSZ  DIMS          UNC   STA := STA with interrupts set;  Handle
19007                                                                                   deferred interrupts
```

```
 RECORD  C S.          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19009          ********************************************************************
19010          *                                                                  *
19011          *          Execute (XEQ) Instruction                               *
19012          *                                                                  *
19013          ********************************************************************
19014          *
19015          $LUT INSTR=XEQ:0 011 000 001 10x xxx, DSPL=4, ENTRY=XEQ
19016          *
19017   136D   XEQ        SM   INC            WRS   SR   SM   REPC      SP3B DCSR SRZ   Write TOS registers at SM + 1; Repeat if
19019                                                                                    SR <> 0, dec SR, SP3B := UBB := S
19020   136E               QDWN ADD            DATA       UBB  ADD       SM   DCSR SRL2 Empty TOS register; Repeat until SR in rank1
19022                                                                                    is < 2, dec SR, SM := S (if SR = 0 at
19023                                                                                    start of instruction, store into SM may be
19024                                                                                    NOPed)
19025   136F         UBB        ADD                    FFFF P    ADDL      P            UBA=SM ;P=P-1                          (2324)
19027   1370         UBA  DSPL  SUB        SP4A        Z         ADD            CLSR   SP4A := UBA := addr of instruction (S - K);
19029                                                                                    UBB := Z, clear SR                     (2324)
19030   1371         UBA  DB    BNDE                   SP3B UBB  JSZC STOV           POS  Bounds check addr >= DB; STOV if S > Z
19032   1372                    ADD                         BNKS ADD       BNKP           ; BNKP := S-BANK                     (2324)
19034   1373               SP4A ADD            RONP        BNKP ADD                      Read instruction into NIR; P := P - 1 so
19036                                                                                    that instruction after XEQ follows 'XEQed'
19037                                                                                    instruction, SREG := P-BANK            (2324)
19038   1374                    JSZ  NOP        UNC         UBB  ADD       BNKP          Wait two clocks for NEXT; Restore BNKP(2324)
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19041           ********************************************************************
19042           *                                                                  *
19043           *    ADSS is the entry point for both the ADDS and SUBS            *
19044           *    instructions.  F2 is set for the SUBS instuction and not      *
19045           *    for ADDS.  DSPL is set to 8 so as to access the immediate     *
19046           *    operand in CIR.                                               *
19047           *                                                                  *
19048           ********************************************************************
19049           *
19050           $LUT INSTR=ADDS:0 011 101 0xx xxx xxx, DSPL=8, ENTRY=ADSS
19051           $LUT INSTR=SUBS:0 011 101 1xx xxx xxx, DSPL=8, F2, ENTRY=ADSS
19052           *
19053    1375   ADSS       SM   INC              WRS   SR   SM   REPC       SP3B DCSR SRZ   Write at SM + 1; SP3B := S, REPC if SR > 0
19055    1376              QDWN ADD              DATA       UBB  ADD        SM   DCSR SRL2  Write QDWN; Repeat until SR in rank1 < 2,
19057                                                                                       dec SR, SM := S (if SR = 0 at start of
19058                                                                                       instruction store into SM may be NOPed)
19059    1377              DSPL ADD              NZRO  SP3B       CAD        RH              UBA := immediate operand, skip if <> 0;
19061                                                                                       RH := S - 1
19062    1378              UBB  JSBI ADS1        ROS   SP3B UBA   ADSB       SP1B CLSR       Read TOS and jump if operand in TOS;
19064                                                                                       SP1B := S +/- DSPL, clear SR
19065    1379         UBB  Q    JSZS STUN        NEG   Z    UBB  JSZC STOV             NEG  STUN if S < Q; STOV if Z < S and within 32K
19067    137A                   JSZ  NEXT        UNC        SP1B ADD        SM              Jump to NEXT; SM := S
19069    137B   ADS1 RH   OPA  ADSB       SP4A         SP3B Q    JSZC STUN             NEG  SP4A := S-1 +/- TOS; STUN if old S <= Q
19071    137C         UBA  Q    JSZS STUN        NEG   Z    UBA  JSZC STOV             NEG  STUN if new S < Q and within 32K;
19073                                                                                       STOV if Z < new S and within 32K
19074    137D                   JSZ  NEXT        UNC        SP4A ADD        SM              Jump to NEXT; SM := S
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19077          ********************************************************************
19078          *                                                                  *
19079          *    Set Registers (SETR) Instruction                              *
19080          *                                                                  *
19081          ********************************************************************
19082          *
19083          $LUT INSTR=SETR:0 010 111 1xx xxx xxx, Q, ENTRY=SETR
19084          *
19085   137E   SETR      SM   INC            WRS   SR   SM   REPC      SM   DCSR SRZ   SAVE INSTRUCTION PARAMETERS BY WRITING TO
19087                                                                                    MEMORY
19088                                                                                  SM:=SR + SM, REPEAT IF SR <> 0, DCSR
19089   137F           QDWN ADD            DATA             ADD                DCSR SRL2  WRITE QDWN; REPEAT UNTIL SR IN RANK1 < 2
19091   1380           CIR  ADD  RLZ  SP4A HBF2  Z          ADD       RE   CLSR        SP4A.(0:8):=CIR, F2:=CIR(8);
19093                                                                                  RE:=UBB:=Z, CLSR
19094   1381      UBB  SM   JSZS STOV      NEG   F000 UBA  ANDL                 ZERO  STOV IF Z < S;
19096                                                                                  SKIP IF NO PRIVILEDGED REGISTERS TO SET
19097   1382           SM   ADD       RH   ROBS             JSZ  TRP6           NPRV  READ FIRST PARAMETER; TRAP IF NPRV
19099   1383           SP4A ADD  LSL  RG   HBF2       BNKS JSB  STSB XR1        F2    RG.(0:7):=CIR.(9:7), F2:=CIR(9);
19101                                                                                  XR1:=BNKS, JSB IF SET S-BANK
19102   1384           STA  ADD       XR6                  DB   ADD       XR2         XR6:=STA; XR2:=DB
19104   1385           RG   ADD  LSL  RG   HBF2       BNKD JSB  STDB XR3        F2    F2:=CIR(10); JSB IF SET DB AND DB-BANK
19106   1386           UBA  ADD  LSL       HBF2       DL   JSB  STDL XR4        F2    F2:=CIR(11); XR4:=DL, JSB IF SET DL
19108   1387      RG   RG   ADD  LSL  RG   HBF2  Z         JSB  SETZ XR5        F2    F2:=CIR(12); XR5:=Z, JSB IF SET Z
19110   1388           UBA  ADD  LSL  RG   HBF2       SM   JSB  SSTA XR9       F2    F2:=CIR(13); XR9:=SM, JSB IF SET STATUS
19112   1389      X         ADD       XR7                  BNKD JSB  SETX SP3B       F2    XR7:=X; SP3B:=BNKD, JSB IF SETX
19114   138A      RG   RG   JSB  SETQ      NEG            Q    ADD       RF               JSB TO SET Q IF CIR(14); RF:=Q
19116   138B           CIR  JSB  STSM      ODD            RH   ADD       SM               JSB IF CIR(15); SM:=NEW SM
19118   138C      UBB  Q    JSBS UNFL      NEG   Z        UBB  JSBS OVFL           NEG   JSB FOR UNFL IF S < Q AND WITHIN 32K;
19120                                                                                  JSB FOR OVFL IF Z < S AND WITHIN 32K
19121                                                                                    (OVERFLOW TEST HAS PRIORITY)
19122   138D           DB   ADD                       SP3B XR1  XOR                NZRO  UBA:=DB, SKIP IF BNKD <> BNKS
19124   138E      UBA  DL   SUB                       Z    DB   SUB            SF5B TICB  ; SET F5B IF NOT SPLIT BANKS, SKIP IF
19126                                                                                    (BNKD = BNKS) AND (DB >= DL) AND (Z >= DB)
19127                                                                                    (SKIP IF SPLIT STACK, TICB TEST REQUIRES
19128                                                                                    THAT Q OR S RELATIVE ADDRESSING BE
19129                                                                                    SPECIFIED IN LUT ENTRY)
19130   138F           STA  ADD  LSL       HBF2       Q    ADD                UNC   F2:=INTERRUPT BIT;
19132                                                                                  UBB:=Q, SKIP IF SPLIT STACK
19133   1390      UBB  DL   JSBS UNFL      NEG   UBB  DB   JSBS UNFL           NEG   JSB FOR UNFL IF Q < DL AND WITHIN 32K;
19135                                                                                  JSB FOR UNFL IF Q < DB AND WITNIN 32K AND
19136                                                                                    NOT SPLIT STACK
19137                                                                                  FIXED SO THAT IT JUMPS                (2635)
19138   1391      RE   DL   SUB                       Z    DL   SUB                  JSB FOR STACK UNDERLFOW IF NOT (Z >= DL);
19140                                                                                  JSB FOR STACK UNDERFLOW IF (Z - DL) > 32K
19141   1392      3FFF      ADDL                      OOFF XR1  ANDL       BNKS  F5B   UBA:=MASK TO SET FSS MINUS ONE;
19143                                                                                  BNKS:=NEW VALUE, SKIP IF NOT SPLIT BANKS
19144   1393      1000      ADDL                           UBA  INC            CCPX UNC   UBA:=%1000;
19146                                                                                  UBB:=%4000, SET FSS IF SPLIT BANKS, SKIP
19147   1394           CPX2 ADD                 EVEN  UBA  UBB  IOR            CCPX       SKIP IF NO DEFERRED INTERPUPTS;
19149                                                                                  UBB:=%5000, CLEAR FSS IF NOT SPLIT BANKS
19150   1395                JSZ  DIMS      F2                 ADD                NEXT  Handle deferred interrupt if interrupts on;
19152                                                                                    NEXT
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19154   1396  STSB RH   Q    JSZC STUN      NEG   RH         CAD        RH   ROS        STUN IF SM <= Q; RH:=NEW SM, READ TOS
19156   1397               ADD                                OPB  ADD   XR1       RSB   ; XR1:=NEW BNKS, RSB
19158   1398  STDB RH        CAD            ROBS              OPB  ADD   DB              READ TOS; DB:=PARAMETER FROM TOS
19160   1399       UBA  Q    CAD            POS   UBA         CAD        RH   ROS        SKIP IF NEW SM > Q; RH:=NEW SM, READ TOS
19162   139A            RG   JSB  RBKD      UNC   00FF OPB   ANDL       BNKD       RSB   UBA := shifted CIR, JSB if new SM <= Q;
19164                                                                                    BNKD := parameter from TOS, return
19165   139B  STDL RH   Q    JSBC RBKD      NEG         DB    ADD                        JSB IF SM <= Q; UBB:=DB
19167   139C       RH        CAD       RH   ROBS  UBB   OPB   ADD        DL          RSB   RH:=NEW SM, READ TOS;
19169                                                                                    DL:=DB + TOS PARAMETER, RSB
19170   139D  SETZ RH   Q    JSBC RSDL      NEG         DB    ADD                        JSB IF SM <= Q; UBB:=DB
19172   139E       RH        CAD       RH   ROBS  UBB   OPB   ADD        Z               RH:=NEW SM, READ TOS;
19174                                                                                    Z:=DB + TOS PARAMETER
19175   139F            RG   ADD                           UBB   ADD        RE          RSB   UBA:=SHIFTED CIR; RE:=NEW Z, RSB
19177   13A0  SSTA RH   Q    JSBC RESZ      NEG         OPB   ADD        SP1B            JSB IF SM <= Q; SP1B:=NEW STA
19179   13A1            STA  ADD            POS   2F00  UBB   ANDL       SP2B            SKIP IF PRIVILEDGED; SP2B:=NON-PRIVILEDGED
19181                                                                                      BITS OF NEW STA
19182   13A2            SP1B ADD       STA  UNC   D0FF  UBA   ANDL                       STA:=NEW STATUS;
19184                                                                                    UBB:=PRIVILEDGED BITS OF OLD STA
19185   13A3       UBB  SP2B IOR       STA        RH          CAD        RH   ROS  RSB   SET NON-PRIVILEDGED BITS OF STA;
19187                                                                                    RH:=NEW SM, READ TOS, RSB
19188   13A4  SETX RH   Q    JSBC RESX      NEG         OPB   ADD                        JSB IF SM <= Q;
19190                                                                                    UBB:=PARAMETER FOR TOS
19191   13A5            UBB  ADD       X         RH          CAD        RH   ROS  RSB   X:=NEW X; RH:=NEW SM, READ TOS, RSB
19193   13A6  SETQ RH        CAD       RH   ROBS        DB    ADD                        RH:=NEW SM, READ TOS; UBB:=DB
19195   13A7       RH   Q    JSBC RESQ      NEG   UBB   OPB   ADD        Q           RSB   JSB IF SM <= Q; Q:=DB + PARAMETER, RSB
19197   13A8  STSM RH   RF   JSBC RESQ      NEG         DB    ADD                        JSB IF SM <= Q; UBB:=DB
19199   13A9                 ADD                       UBB   OPB   ADD        SM          RSB   ; SM:=DB + PARAMETER FROM TOS, RSB
19201           *
19202           ********************************************************************
19203           *    The following code restores the environment prior to the      *
19204           *    start of the SETR instruction in the event of a stack         *
19205           *    underflow or overflow.  All instruction parameters in the     *
19206           *    TOS registers are saved by writing them to memory.  The       *
19207           *    original values of all registers are stored in the course     *
19208           *    of executing the instruction.  If stack underflow is detected *
19209           *    while reading a parameter from the TOS in memory, all         *
19210           *    registers which may have been set up to that point are        *
19211           *    restored.  All registers are restored if stack overflow or    *
19212           *    underflow is detected as a result of setting Z, Q, or S.      *
19213           ********************************************************************
19214           *
19215   13AA  OVFL                ADD                                ADD             SF4B        ; SF4B TO INDICATE OVERFLOWS
19217   13AB  UNFL                ADD                           XR9  ADD        SM               BEGIN RESTORING REGISTERS, RESTORE SM
19219   13AC  RESQ                ADD                           RF   ADD        Q                ; RESTORE Q
19221   13AD  RESX       XR7      ADD            X                   ADD                          RESTORE X;
19223   13AE  RESZ       XR6      ADD            STA           XR5   ADD        Z                RESTORE STA; RESTORE Z
19225   13AF  RSDL                ADD                           XR4   ADD        DL               ; RESTORE DL
19227   13B0  RBKD                ADD                           XR3   ADD        BNKD             ; RESTORE BNKD
19229   13B1                      JSZ  STOV      UNC           XR2   JSZ  STUN  DB          NF4B  JSB FOR STACK OVERFLOW;
19231                                                                                    RESTORE DB, JSB IF STACK UNDERFLOW
```

```
 RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19233          ************************************************************************
19234          *                                                                      *
19235          *    Push Registers (PSHR) Instruction                                 *
19236          *                                                                      *
19237          ************************************************************************
19238          *
19239          $LUT INSTR=PSHR:0 010 100 1xx xxx xxx, DSPL=8, ENTRY=PSHR
19240          *
19241    13B2  PSHR       CIR  CSR            HBF2  SR    SM   ADD       RH              F2:=CIR(15); RH:=(SR+SM)
19243    13B3             UBA  CSR       SP4A HBF2        UBB  JSBS PSH1 SP3B      F2    F2:=CIR(14); SP3B:=-(S), JSB IF CIR(15)
19245    13B4             UBA  CSR       SP4A HBF2        Q    JSB  PSH1 RH        F2    F2:=CIR(13); RH:=Q, JSB IF CIR(14)
19247    13B5        X    -    ADD                         UBA  CSR       SP2B HBF2 NF2   UBA:=X; F2:=CIR(12), SKIP IF NOT CIR(13)
19249    13B6             STA  ADD       SP4A       UBA  DB   JSB  PSH1 RH        NF4B  SP4A:=STA; RH:=X+DB, MEDJSB IF CIR(13)
19251    13B7             SP2B CSR       SP4A HBF2  UBA  DB   JSB  PSH1 RH        F2    F2:=CIR(11); RH:=STA+DB, JSB IF CIR(12)
19253    13B8             UBA  CSR       SP4A HBF2  Z         JSB  PSH1 RH        F2    F2:=CIR(10); RH:=Z, JSB IF CIR(11)
19255    13B9             DSPL JSB  PSHF      ZERO       DL   JSB  PSH1 RH        F2    JSB IF PSHR 0; RH:=DL, JSB IF CIR(10)
19257    13BA             SP4A CSR            HBF2  0003 SP4A ANDL                ZERO  F2:=CIR(9); SKIP IF CIR.(8:2)=0
19259    13BB             DB   JSB  PHDB      F2                JSZ  TRP6        NPRV  UBA:=DB, JSB IF CIR(9); TRAP IF NPRV
19261    13BC             CIR  JSB  PSH1      BIT8  UBA  BNKS ADD       RH              JSB IF CIR(8); RH:=DB + BNKS
19263    13BD                  ADD                               ADD                NEXT  ; NEXT
19265    13BE  PSH1       SM   INC            WRS   RH   DB   SUB       RH    INCN SR7   SEND WRITE ADDR; RH:=RH-DB, INCN, SR7?
19267    13BF             SP4A ADD                  Z    UBA  SUB             INSR RSB   UBA:=SP4A;
19269                                                                                    UBB:=Z - NEW SM, INSR AND RSB IF SR < 6
19270    13C0             RH   ADD            DATA       UBB  CSL             LBF5       PUSH QDWN (RH WAS RG BEFORE INCN);
19272                                                                                    F5B:=NEW SM > Z
19273    13C1             SP4A ADD            RSB        SM   JSBI PSTO SM        F5B   UBA:=SP4A, RSB; SM:=SM+1, STOV IF Z < SM
19275          *
19276          ************************************************************************
19277          *    If Stack Overflow is detected, the following code restores        *
19278          *    the stack and pointers to the conditions which existed at         *
19279          *    the start of the PSHR instruction.  If the original value of      *
19280          *    S is less than the current value of SM, i.e. ((SM+1)-S)>=0,       *
19281          *    then there are locations at or below the original S in the        *
19282          *    TOS registers.  These are pushed into memory, and SM is set       *
19283          *    to the original value of S.                                       *
19284          ************************************************************************
19285          *
19286    13C2  PSTO       SM   ADD       RH         SP3B SM   REPC      SP1B DCSR POS   RH:=SM; SP1B:=UBB:=-(OLD S)+SM, REPC IF>0
19288    13C3             QDWN ADD                  UBB       INC              DCSR ZERO  PUSH TOS AT OR BELOW OLD S INTO MEMORY
19290    13C4                  JSZ  STOV      DATA  RH   SP1B SUB       SM   CLSR      JSB STOV; SM:=SM-((-OLD S)+SM)=OLD S, SR:=0
19292    13C5  PHDB       SM   INC            WRS        BNKD ADD       RH   INCN SRG5  SEND WRITE ADDR; RH:=BNKD, INCN, SR > 5?
19294    13C6             QDWN ADD            DATA  UBA        ADD       RH   INSR      PUSH QDWN; RH:=UBB:=SM+1, INSR IF SR<=4
19296    13C7             QDWN ADD            DATA  UBB        INC       SM   INCN SRG5  PUSH QDWN; SM:=SM+2, SKIP IF SR > 5
19298    13C8             DB   ADD       RA         RH         CAD       SM   INSR RSB   RA:=UBA:=DB; RESTORE SM IF SR<=4, RSB
19300    13C9             DB   ADD            RSB   Z    SM   JSBS PSTO             NEG   UBA:=DB, RSB; STOV IF SM > Z
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19303           ************************************************************************
19304           *    The following code fills the TOS registers for diagnostic        *
19305           *    purposes when the instruction calls for zero registers to        *
19306           *    be pushed onto the TOS.                                          *
19307           ************************************************************************
19308           *
19309   13CA    PSHF       SM   ADD                ROS   SR        ADD       CTR              READ TOS IN MEMORY; CTR:=SR FOR REGN
19311   13CB               JSZ  NEXT               SR7   UBA  Q    JSZC NEXT          NCRY  JSB FOR NEXT IF SR=7 OR NOT (SM > Q)
19313   13CC               OPA  ADD       REGN     RREG  CAD       SM   INSR              PULL STACK OPERAND INTO REGISTER;
19315                                                                                      SM:=UBB;=SM - 1, INSR
19316   13CD               UBB  JSB  *-2           ROS   ADD       ICTR                   READ NEW TOS IN MEMORY, LOOP BACK;
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19319          ************************************************************************
19320          *    Double Exchange (DXCH) Instruction                                *
19321          ************************************************************************
19322          *
19323          $LUT INSTR=DXCH:0 000 001 110 xxx xxx, ENTRY=DXCH, SR=4
19324          *
19325    13CE  DXCH            ADD                       RB   ADD       RD             ; (S-3) := (S-1)
19327    13CF                  ADD                       RD   ADD       RB   CCA       ; (S-1) := (S-3), CCA on msw
19329    13D0                  ADD                       RA   ADD       RC             ; (S-2) := (S)
19331    13D1                  ADD            NEXT       RC   ADD       RA   DCC      NEXT; (S) := (S-2), DCC on lsw
19333          *
19334          ************************************************************************
19335          *    Exchange A and B (XCH) Instruction                                *
19336          ************************************************************************
19337          *
19338          $LUT INSTR=XCH:0 000 011 010 xxx xxx, ENTRY=XCH, SR=2
19339          *
19340    13D2  XCH       RA    ADD       RB                RB   ADD                    (S-1) := (S); UBB := (S-1)
19342    13D3  XCH1      UBB   ADD       RA   CCA                ADD            NEXT  (S) := (S-1), CCA; NEXT
19344          *
19345          ************************************************************************
19346          *    Exchange A and X (XAX) Instruction                                *
19347          ************************************************************************
19348          *
19349          $LUT INSTR=XAX:0 000 011 101 xxx xxx, ENTRY=XAX, SR=1
19350          *
19351    13D4  XAX       RA    ADD       X                 ADD                    X := (S)
19353    13D5        X         ADD       RA   CCA          ADD            NEXT  (S) := X, CCA; NEXT
19355          *
19356          ************************************************************************
19357          *    Rotate A,B,C (CAB) Instruction                                    *
19358          ************************************************************************
19359          *
19360          $LUT INSTR=CAB:0 000 101 110 xxx xxx, ENTRY=CAB, SR=3
19361          *
19362    13D6  CAB             ADD                       RA   ADD       RB             ; (S-1) := (S)
19364    13D7                  ADD            RC         RC   JSB   XCH1          UNC   (S-2) := (S-1); UBB := (S-2), jump to finish
19366          *
19367          ************************************************************************
19368          *    Exchange B and X (XBX) Instruction                                *
19369          ************************************************************************
19370          *
19371          $LUT INSTR=XBX:0 000 111 101 xxx xxx, ENTRY=XBX, SR=2
19372          *
19373    13D8  XBX       RB    ADD       X                 ADD                    X := (S-1)
19375    13D9        X         ADD       RB                ADD            NEXT  (S-1) := X; NEXT
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19378           ********************************************************************
19379           *    TASL (Triple arithmetic shift left) Instruction               *
19380           ********************************************************************
19381           *
19382           $LUT INSTR=TASL:0 001 001 000 XX,ENTRY=TASL,SR=3,DSPL=6
19383           $LUT INSTR=TASL X:0 001 101 000 XX,ENTRY=TASL,SR=3,DSPL=6,X
19384           *
19385    13DA   TASL            ADD         SP4A                ADD                    SP4A:=0;
19387    13DB          XC   DSPL JSB  TAS1        UNC             ADD       SP3B         UBA:=SHIFT COUNT, JSB; SP3B:=0
19389    13DC               UBA  QASL                           UBB  LINK          DCTR CTR0  SHIFT UNTIL CTR IS DECREMETED TO 0
19391    13DD               SREG ADD        RC   CCA            SREG ADD                   STORE (S-2), CCA; UBB:=(S-1)
19393    13DE   TAS2 SPOA       ADD                             UBB  ADD       RB   DCC     UBA:=(S); STORE (S-1), DCC
19395    13DF                   ADD                 NEXT        UBA  ADD       RA   DCC     NEXT; STORE (S), DCC
19397    13E0   TAS1       RA   ADD         SPOA        003F   UBA  ANDL      CTR          SPOA:=(S); CTR:=SHIFT COUNT MOD 64
19399    13E1               RC   ADD                 RSB         RB   REPC          DCTR CTR0  UBA:=(S-2), RSB;
19401                                                                                      UBB:=(S-1), REPEAT IF CTR <> 0
19402           *
19403           ********************************************************************
19404           *    TASR (Triple arithmetic shift right) Instruction              *
19405           ********************************************************************
19406           *
19407           $LUT INSTR=TASR:0 001 001 001 XX,ENTRY=TASR,SR=3,DSPL=6
19408           $LUT INSTR=TASR X:0 001 101 001 XXX,ENTRY=TASR,SR=3,DSPL=6,X
19409           *
19410    13E2   TASR XC   DSPL JSB  TAS1        UNC             ADD                    UBA:=SHIFT COUNT, JMP;
19412    13E3               UBA  QASR                           UBB  LINK          DCTR CTR0  SHIFT UNTIL CTR IS DECREMENTED TO 0
19414    13E4               SREG ADD        RC   CCA            SREG JSB  TAS2          UNC   STORE (S-2); UBB:=(S-1), JSB TO FINISH
19416           *
19417           ********************************************************************
19418           *    TNSL (Triple normalizing shift left) Instruction              *
19419           ********************************************************************
19420           *
19421           $LUT INSTR=TNSL:0 001 001 110 XX,ENTRY=TNSL,SR=3
19422           $LUT INSTR=TNSL X:0 001 101 110 XX,ENTRY=TNSL,SR=3,X
19423           *
19424    13E5   TNSL       RA   ADD         SPOA        03FF RC   ANDL      SP1B         SPOA:=(S); SP1B:=UBB:=(S-2).(6:10)
19426    13E6          RA   RB   IOR                   NZRO  FE00 UBB  ADDL      SP2B       CRRY  SKIP IF (S).(S-2) <> 0;
19428                                                                                      SP2B:=(S-2).(0:7):=%177, SKIP IF (S-2).(6:1)
19429    13E7               SP1B JSB  TNS0        ZERO            ADD       SP3B SF1       JSB IF (S).(S-1).(S-2).(6:42) = 0;
19431                                                                                      SP3B:=0, SF1 IF NOT (S-2).(6:1)
19432    13E8                    ADD         SP4A        0100       ADDL      RH           SP4A:=0; RH:=MASK TO TEST BIT 7
19434    13E9               SP2B ADD                 NF1         RB   REPC                UBB:=(S-2); UBB:=(S-1), REPEAT IF F1
19436    13EA          RH   UBA  QASL                CRRY        UBB  LINK      RB   ICTR  REPEAT AND INCTREMENT CTR UNTIL (S-2).(6:1)
19438                                                                                      UBA:=NORMALIZED (S-2);
19439                                                                                      RB:=NORMALIZED (S-1), ICTR
19440    13EB          0200 SREG IORL                          CTRS ADD                   SET BIT 6 IN (S-2); UBB:=SHIFT COUNT
19442    13EC   TNS1 XC   UBB  ADD         X                   UBA  ADD       RC   CCA   X:=X (CONDITIONALLY) + SHIFT COUNT;
19444                                                                                      (S-2):=NORMALIZED (S-2), CCA
19445    13ED          SPOA       ADD         RA                      ADD                NEXT  (S):=NORMALIZED (S); NEXT
19447    13EE   TNS0               JSB  TNS1        UNC     002A       ADDL               JUMP BACK; UBB:=42
```

```
RECORD   C S          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19450           ************************************************************************
19451           *    Test TOS (TEST) Instruction                                       *
19452           ************************************************************************
19453           *
19454           $LUT INSTR=TEST:0 000 010 101 xxx xxx, ENTRY=TEST, SR=1
19455           *
19456    13EF   TEST           ADD                        RA   JSB  XCH1          UNC   ; UBB := (S), jump to finish
19458           *
19459           ************************************************************************
19460           *    Test double word on TOS (DTST) Instruction                        *
19461           ************************************************************************
19462           *
19463           $LUT INSTR=DTST:0 000 010 111 xxx xxx, ENTRY=DTST, SR=2
19464           *
19465    13F0   DTST      RB   JSZ  DTS1      UNC   RA   RA   LINK          CCRY      UBA := msw + lsw(0); Clear carry bit in STA
19467           *
19468           ************************************************************************
19469           *    Test byte on TOS (BTST) Instruction                               *
19470           ************************************************************************
19471           *
19472           $LUT INSTR=BTST:0 000 011 001 xxx xxx, ENTRY=BTST, SR=1
19473           *
19474    13F1   BTST           JSZ  NEXT      UNC        RA   ADD           CCB       Jump to NEXT; Set CCB on TOS
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19477          ********************************************************************
19478          *          LINKED LIST SEARCH INSTRUCTION                          *
19479          ********************************************************************
19480          *
19481          $LUT INSTR=RSW/LLSH/I-O:0 010 000 011 00x xxx, ENTRY=LLSH
19482          *
19483    13F2  LLSH       CIR  CSR       SP4A        0002       ADDL      SP1B             SP4A := CIR &csr(1); SP1B := 2 (for I/O)
19485    13F3             STA  JSZ  TRP6       POS         UBA  JSLZ RSW            POS   JMP if I/O or RSW instr else mode violation?
19487    13F4                  JSZ  PULM       SRL2        SP1B JSZI PUL2           SRZ   Pull 2 TOS if SR = 0 else pull 1 if SR = 1
19489                                                                                      ; UBB := 3 (only for when SR >= 3 on entry)
19490    13F5       SR   UBB  JSZC PULM       NCRY  SR    SP1B JSZC PUL2           NCRY  Fill TOS to at least 4
19492    13F6       X         INC       X         00FF  RB   ANDL      BKX3             X := X + 1 to negate 1st check; BKX3 := bank
19494    13F7       RD   RA   ADD             ROX3              ADD                      Read target word
19496    13F8  LLS1 X         CAD       X       NZRO  RA         ADD       SP1B ROX3      X:=X - 1, SKIP IF <> 0; READ NEW LINK BANK
19498    13F9                 JSZC NEXT       CCA   RA         INC            ROX3      CCL IF X=0, JSB FOR NEXT; READ NEW LINK ADDR
19500    13FA       RC   OPA  JSBC LLS2       NCRY        OPB  ADD       BKX3             TERMINATE IF NOT (TEST WORD > TARGET);
19502                                                                                      BKX3:=LINK BANK
19503    13FB            OPA  JSB  LLS3       TEST        OPB  ADD       RA               UBA:=TARGET, JSB IF INTERRUPTS; RA:=ADDR
19505    13FC       RD   UBB  JSB  LLS1       ROX3  UBA   BKX3 XFRS      RB               READ NEW TARGET WORD, LOOP BACK;
19507                                                                                      RREG:=TARGET, RB:=NEW LINK BANK
19508    13FD  LLS2            INC            CCA   RREG       LINK                NEXT  INCR TARGET, CCG IF ALL 1'S ELSE CCE, NEXT
19510    13FE  LLS3            JSZ  IRDN       UNC         SP1B ADD       RA               JSB TO INTERRUPT HANDLER; RESTORE RA
19512    13FF                  ADD                                ADD                      NOP to keep system happy                (2555)
```

```
RECORD   C S                ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19515            ************************************************************************
19516            *                                                                      *
19517            *         DEC is the entry point for the decimal instruction set       *
19518            *                                                                      *
19519            *         The last four bits of the op-code are used to index the      *
19520            *         decimal jump table.                                          *
19521            *                                                                      *
19522            ************************************************************************
19523            *
19524            $LUT INSTR=EDIS:0 010 000 11x xxx xxx, DSPL=4, ENTRY=DEC
19525            *
19526            %1400
19527            *
19528     1400   DECT 0070 CIR  ANDL                          JSL  DMPY          UNC   1. DOUBLE  LOGICAL MULTIPY;
19530                                                                                      CHECK CIR ( 9:3 )
19531     1401        0060 CIR  ANDL                          JSL  CVAD          UNC   2. ASCII TO DECIMAL CONVERSION;
19533                                                                                      CHECK CIR ( 9:2 )
19534     1402             ADD                                JSL  CVDA          UNC   3. DECIMAL TO ASCII CONVERSION;
19536
19537     1403        0060 CIR  ANDL       XR2                JSL  CVBD XR2      UNC   4. BINARY TO DECIMAL CONVERSION;
19539                                                                                      CHECK CIR ( 9:2 )
19540     1404        0060 CIR  ANDL       XR4                JSL  CVDB XR2      UNC   5. DECIMAL TO BINARY CONVERSION;
19542                                                                                      CHECK CIR ( 9:2 )
19543     1405        0040 CIR  ANDL       XR6                JSL  SLD  XR2      UNC   6. DECIMAL LEFT SHIFT;
19545                                                                                      CHECK CIR ( 9:1 )
19546     1406        0040 CIR  ANDL       XR6                JSL  SLD  XR2      UNC   7. DECIMAL NORMALIZING LEFT SHIFT;
19548                                                                                      CHECK CIR ( 9:1 )
19549     1407        0040 CIR  ANDL       XR6                JSL  SRD  XR2      UNC   8. DECIMAL RIGHT SHIFT;
19551                                                                                      CHECK CIR ( 9:1 )
19552     1408        0040 CIR  ANDL       XR6                JSL  ADDD XR2      UNC   9. DECIMAL ADD;
19554                                                                                      CHECK CIR ( 9:1 )
19555     1409        0040 CIR  ANDL       XR6                JSL  ADDD XR2      UNC   A. DECIMAL COMPARE;
19557                                                                                      CHECK CIR ( 9:1 )
19558     140A        0040 CIR  ANDL       XR6                JSL  ADDD XR2      UNC   B. DECIMAL SUBTRACT;
19560                                                                                      CHECK CIR ( 9:1 )
19561     140B        0040 CIR  ANDL       XR6                JSL  MPYD XR2      UNC   C. DECIMAL MULTIPLY;
19563                                                                                      CHECK CIR ( 9:1 )
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19565            "
19566            "
19567            "
19568            "
19569            "        START OF DECIMAL INSTRUCTION SET ---
19570            "
19571            "         0  1  2  3  4  5  6  7  8  9  10  11  12   13  14  15
19572            "        I-------------------------------------------------------I
19573            "        I 0  0  1  0  0  0  0  1  1  X   X   X   -   DSPL     - I
19574            "        I-------------------------------------------------------I
19575            "
19576            "         1.  ADJUST TOS REGISTERS; ONLY 3 TOS REGISTERS ARE VALID.
19577            "         2.  CALCULATE ENTRY JUMP TARGET.
19578            "         3.  ALL EXTENDED REGISTERS USED IN THE DECIMAL INSTRUCTION
19579            "             SET ARE WITH REGISTER FLAG SET ( SRF ); & CLEAR ( CRF )
19580            "             ON EXIT OF THE INSTRUCTION.
19581            "         4.  MOST OF THE INSTRUCTION SET MAKE USE OF XRA1 - XRA9,
19582            "             & XRB0 - XRB8 AS TEMPORARY DATA BUFFERS.  THEY ARE
19583            "             TWO IDENTICAL COPIES.  THESE BUFFERS ARE INITIALLY
19584            "             ZEROED BEFORE FETCHING DATA.
19585            "
19586            "
19587   140C  DEC       DSPL JSZ  TRP        ZERO  FFF3      ANDL      CTX           TRAP IF DSPL := 0; MASK, CLEAR CTX
19589   140D           INC  LSL        CF2   RREG UBA  JSZ  TRP       POS   2, CF2; TRAP IF DSPL >= D
19591   140E     SR    UBA  JSZC ETO3       CRRY  UBA       JSZC PULM SP3B      SRZ   EMPTY TOS TO 3; PULM IF SR := 0
19593   140F           JSZ  PULM       NF2                   JSZ  PUL2      SRL2  PULM IF NF2; PUL2 IF SR < 2
19595   1410     DECT DSPL ADDL LBL              SP3B        CSR       RF    SRF   CALCULATE JUMP TARGET; RF := 8000, SRF
19597   1411     UBA       CAD             WRD               ADD       XR0         ADJUSTED JUMP TABLE ADDR; XRB0 := 0
19599   1412               ADD        XR0  RAR               ADD       XR1   CLO   YRA, XRA0 := 0; XRB1 := 0, CLEAR OVERLFOW
19601   1413               ADD        XR1  CF3A        RC     ADD       XR26        XRA1 := 0, CF3A, SAVE RC ( FOR MPYD )
19603   1414               ADD        XR10                    ADD       XR9         XRA10 := 0; XRB9 := 0
19605   1415               STA  ADD             HBF2  FFE3      ADDL      SP1B  RSB   SET F2 IF PRIV MODE;
19607                                                                                SP1B := -( VALID DIGIT COUNT ), TO JMP TABLE
```

```
 RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19609           *
19610           *
19611           *
19612           *          ASCII TO DECIMAL CONVERSION ---
19613           *
19614           *          ON ENTRY :
19615           *
19616           *                     SR  := 3
19617           *                     RA  := SOURCE DIGIT COUNT
19618           *                     RB  := SOURCE BYTE ADDRESS ( DB REL )
19619           *                     RC  := TARGET DIGIT COUNT
19620           *                     RD  := TARGET BYTE ADDRESS ( DB REL )
19621           *                            ( VALID AFTER PULM )
19622           *
19623           *             TARGET ADDR := ASCII SOURCE CONVERTED TO PACKED
19624           *             DECIMAL.  ASCII SOURCE MUST BE LEADING BLANKS ( %40 )
19625           *             OR DECIMAL DIGITS.  THE RIGHTMOST DIGIT MAY BE SIGNED.
19626           *             BOTH DIGIT COUNTS MUST BE IN THE RANGE 0 <= N <= 28.
19627           *             IF EITHER DIGIT COUNT IS ZERO ONLY THE SDEC IS
19628           *             PERFORMED.
19629           *
19630           *
19631     1416  CVAD        RB   ADD        RG                        JSZ  PULM           UNC   SAVE RB; PULM FOR RD
19633     1417        RC    SP1B JSB  AD17       CRRY  RA   SP1B JSB  AD17           CRRY  TRAP IF RC > 28; TRAP IF RA > 28
19635     1418              RA   JSB  SD24       ZERO       RC   JSB  SD24           ZERO  DEC STACK & NEXT IF RA OR RC := 0
19637     1419              DB   ADD            NFSS  Z    DB   SUB            CF1   NCRY  SKIP IF NOT SPLIT BANKS; SKIP IF DB > Z
19639     141A                   ADD            CF1   UBA  DL   SUB            CTF1        ; F1 SET IF NOT SPLIT STACK:
19641                                                                                        ( (NFSS) AND (Z>=DB) AND (DB>=DL) )
19642     141B              RC   ADD        RH                        JSL  CKCD           UNC   RH:=NIBBLE COUNT; GO CHECK TARGET ADDRESS
19644     141C        RA         CAD                  0003            ADDL      CTR       NF5B  UBA:=RA-1;
19646                                                                                              CTR:=3 (DEC DIGS TO FILL WD), SKIP IF F5B=0
19647     141D        UBA  RA    ADD        RH                        ADD                 SF4B  RH:=NIBBLE COUNT (2RA-1); F4B:=F5B
19649     141E        SPOA SP1B  ADD        RF   ROD                  JSL  CKAB           UNC   OPA:=TARGET SIGN WD, RF:=TARGET ADDRESS;
19651                                                                                              CHECK SOURCE ADDRESS
19652     141F              OPA  ADD  RRZ       CF3A       RC   JSBS CAD1 XR0        F4B   UBA:=POSS. EXTRA BYTE IN TARG SIGN WD;
19654                                                                                              XRB0:=-(DEC CNT), JUMP IF TARG SIGN WD FULL
19655     1420        UBA  UBA  ADD  LSL  RH   UNC   0001       ADDL      CTR               RH(6:8):=EXTRA BYTE, SKIP;
19657                                                                                              CTR:=1 (DEC DIGITS TO FILL WD) IF EXTRA BYTE
```

```
 RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19659           *
19660           *
19661           *
19662           *         CAD1-4 SET UP RE ( 4:8 ) WITH THE EXTRA BYTE IN THE
19663           *         TARGET SIGN WORD ( IF ANY ), LOAD RE ( 0:4 ) WITH THE
19664           *         APPROPRIATE SIGN PATTERN, AND SET THE CONDITON CODE.
19665           *
19666           *
19667     1421  CAD1 UBA  UBA  SUB        RH   UNC   SP3B SP4A ADD        SP3B ROD  F5B   RH:=0 IF NO EXTRA BYTE, SKIP;
19669                                                                                      OPB:=SOURCE SIGN WD, SP3B:=ADDR, WORD FULL?
19670     1422        RREG SREG ADD  LSL  RH           UBB        CAD        SP3B ROD  UNC   RH(4:8):=EXTRA BYTE;
19672                                                                                      IF NOT FULL, DEC ADDR & GET FULL WD, SKIP
19673     1423             RA   SUB        XR0  CCA         OPB  ADD  RRZ  SP2B SF4B UNC   XRA0:=~(ASCII COUNT), COND CODE:= "<0"
19675                                                                                      SP2B(8:8):=SIGN BYTE IF WD FULL, F4B:=1,SKIP
19676     1424                  ADD        SP0A             SREG ADD  LRZ  SP2B CF4B       SP0A:=0 (CCE INDIC.);
19678                                                                                      SP2B(8:8):=SIGN BYTE IF WD NOT FULL, F4B:=0
19679     1425        D000 RH   IORL       RE               CTRS ADD  LSR        LBF5      RE:=MINUS SIGN (AND POSSIBLE EXTRA BYTE)
19681                                                                                      F5B:=(RE LACKS > 1 DECIMAL DIGITS)
19682     1426        FF83 SP2B ADDL       RH         FFC6       ADDL       SP2B           RH:=SIGN BYTE - %175; SP2B:=-%72
19684     1427        0002 UBA  ADDL       RG               UBA  JSL  CAD7                ZERO  RG:=SIGN BYTE - %173; JUMP IF DIGIT IS -0
19686     1428        RREG      ADD              CCA  CFFF  RE   ANDL       RE             COND CODE:= ">0"; RE(0:4):=PLUS SIGN
19688     1429        0028 RG   ADDL                          RG   JSL  CAD7 RH    ZERO  UBA:=SIGN BYTE - %123; RH:=SP4A, JUMP IF +0
19690     142A        0009 UBA  ADDL                          UBA  JSB  AD14 RH    POS   UBA:=SIGN BYTE - %112; RH:=UBA, JUMP IF ERR
19692     142B        0009 UBA  ADDL                          UBA  JSL  CAD3 RH    POS   UBA:=SIGN BYTE - %101; RH:=UBA, JUMP IF NEG
19694     142C        0007 UBA  ADDL                          UBA  JSL  CAD4 RH    POS   UBA:=SIGN BYTE - %72; RH:=UBA, JUMP IF POS
19696     142D             UBA  ADD                           UBA  JSB  AD14 RH    POS   UBA:=SIGN BYTE - %72; RH:=UBA, JUMP IF ERROR
19698     142E        000A UBA  ADDL                    F000 RE   IORL       RE             UBA:=SIGN BYTE - %60; RE(0:4):=F (UNSIGNED)
19700     142F        0010 UBA  ADDL                          UBA  JSL  CAD7 RH    POS   UBA:=SIGN BYTE - %40; RH:=UBA, JUMP/UNSIGNED
19702     1430        0004      ADDL                          UBA  JSB  AD14 RH    NZRO  UBA:=4, RH:=UBA, JMP IF INV DIGIT (NOT BLNK)
19704     1431  CAD2       RE   ADD  SWAB                     UBA  JSL  CADC RG    UNC   CBA0 => A0CB; RG:=4, JUMP TO BLANK LOOP
19706     1432  CAD3            CAD              CCA  1000  RE   IORL       RE             COND CODE:= "<0"; RE(0:4):=MINUS SIGN
19708     1433  CAD4 0004       ADDL       RG               RH   INC                       INIT RG FOR LOOP; ADJUST DIGIT RESIDUE
19710     1434                  ADD                           UBB  JSL  CAD6 RH    UNC   ; UBB,RH:=DECIMAL PATTERN, JUMP TO LOOP
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19713           *
19714           *         CAD5 & CAD6 COMPRISE THE MAIN " INNER LOOP " OF THE CVAD
19715           *         ROUITNE.  EACH ASCII DIGIT IS PLACED INTO THE LOWER 8
19716           *         BITS OF RH, CHECKED FOR VALIDITY, STRIPPED OF EXTRA BITS,
19717           *         EFFECTIVELY SHIFTED RIGHT 4 BIT POSITIONS INTO RE IN
19718           *         ORDER TO OBTAIN ITS DECIMAL REPRESENTATION, AND WRITTEN
19719           *         TO THE APPROPRIATE TARGET LOCATION WHEN RE IS FULL.  IF
19720           *         ASCII RUNOUT OCCURS, OR AN ASCII BYTE < %60 IS DETECTED
19721           *         ( BLANK ), THE CADA, CADB, CADC LOOP IS ENTERED.
19722           *
19723           *
19724     1435  CAD5            JSB  CADA      NF2        OPB  ADD  LRZ  RH   CF4B F4B   JUMP FOR ZERO FILL IF ASCII RUNOUT;
19726                                                                                    RH(8:8):=OPB(0:8), F4B:=0, SKIP IF F4B
19727     1436        0010      ADDL      SP4A            SREG ADD  RRZ  RH   SF4B UNC   SP4A:=0010 (USED FOR BLANK CHECK AT CADB);
19729                                                                                    RH(8:8):=OPB(8:8) IF R BYTE NEXT,F4B:=1,SKIP
19730     1437        FFD0      ADDL                 SP3B       CAD       SP3B ROD       UBA:=-%60; IF L BYTE NOW, GET NEXT WD
19732     1438        RH   SP2B JSB  AD14      CRRY  RH   UBA  JSB  CADB RH         NCRY  TRAP IF RH >= %72; RH:=RH-%60, JUMP IF < 0
19734     1439  CAD6  UBB  RE   ADD  SWAB                       ADD                       000D CBA0 => ADCB;
19736     143A        UBA  UBA  ADD  LSL            UBA  UBA  LINK            DCTR        SHIFT UBA,UBA LEFT 2 BITS; DEC DIGIT COUNT
19738     143B        UBA  UBA  ADD  LSL  RE       UBB  UBB  LINK                        RE:=DCBA (DECIMAL DIGIT HAS BEEN SHIFTED IN)
19740     143C        SP0A RH   IOR       SP0A            XR0  JSBI CAD8 XR0       ZERO  CAPTURE NONZERO BITS IN SP0A;
19742                                                                                    INCREMENT DECIMAL DIGIT COUNT, JUMP IF DONE
19743     143D             XR0  INC       XR0  HBF2       CTRS ADD  LSR       LBF5 NF5B  INC ASCII DIGIT CNT, F2:=0 IF ASCII RUNOUT
19745                                                                                    F5B:=(RE LACKS > 1 DIGITS), SKIP IF RE FULL
19746     143E             RF   ADD            WRD             JSL  CAD5            UNC   YRA:=TARG ADDR; LOOP IF RE NOT FULL
19748     143F             RE   ADD            DATA            ADD       RE   SF5B       WRITE RE TO MEM IF FULL; RE:=0, F5B:=1
19750     1440        RF        CAD       RF   ROD        RG   JSL  CAD5 CTR         UNC   DEC TARG ADDR, OPA:=WD AT TARG ADDR;
19752                                                                                    CTR:=4 (NIBBLES TO GO), JUMP TO LOOP
19753     1441  CAD7  0004      ADDL      RG              RH   JSL  CAD6            UNC   INIT RG FOR LOOP; UBB:=RH, JUMP INTO LOOP
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19756           *
19757           *
19758           *
19759           *         CADA, CADB, & CADC COMPRISE THE ASCII RUNOUT/ LEADING
19760           *         BLANKS INNER LOOP.  IN THE CASE OF ASCII RUNOUT
19761           *         ( RA < RC ), ZERO FILL IS PLACED IN THE TARGET FIELD.
19762           *         IN THE CASE OF BLANKS, EACH SUBSEQUENT ASCII BYTE IS
19763           *         CHECKED TO MAKE SURE THAT IT ALSO IS A BLANK, AND
19764           *         ZERO FILL IS PLACED IN THE TARGET FIELD.
19765           *
19766           *
19767    1442   CADA          JSB  CADC      NF2        OPB  ADD  LRZ  RH   CF4B F4B   JUMP FOR ZERO FILL IF ASCII RUNOUT;
19769                                                                                  RH(8:8):=OPB(0:8), F4B:=0, SKIP IF F4B
19770    1443          RE    ADD  SWAB                 SREG ADD  RRZ  RH   SF4B UNC   CBA0 => A0CB;
19772                                                                                  RH(8:8):=OPB(8:8) IF R BYTE NEXT,F4B:=1,SKIP
19773    1444                ADD                SP3B       CAD       SP3B ROD       ; IF LEFT BYTE NOW, GET NEXT WORD
19775    1445   CADB    RE   ADD  SWAB           RH   SP4A JSB  AD14              NZRO  CBA0 => A0CB; TRAP IF NOT BLANK
19777    1446   CADC UBA UBA  ADD  LSL            UBA  UBA  LINK                   DCTR  SHIFT UBA,UBA LEFT TWO BITS; DEC DIGIT COUNT
19779    1447        UBA UBA  ADD  LSL  RE         UBB  UBB  LINK                         RE:= 0CBA (ZERO HAS BEEN SHIFTED IN)
19781    1448            XR0  INC       XR0  HBF2       XR0  JSBI CAD8 XR0         ZERO  INC ASCII DIGIT CNT, F2:=0 IF ASCII RUNOUT;
19783                                                                                  INCREMENT DECIMAL DIGIT COUNT, JUMP IF DONE
19784    1449        FFE0      ADDL       SP4A             CTRS ADD  LSR       LBF5 NF5B  SP4A:=-%40 (BLANK);
19786                                                                                  F5B:=(RE LACKS > 1 DIGITS), SKIP IF RE FULL
19787    144A            RF    ADD             WRD              JSL  CADA            UNC   YRA:=TARG ADDR; LOOP IF RE NOT FULL
19789    144B            RE    ADD             DATA             ADD        RE   SF5B       WRITE RE TO MEMORY IF FULL; RE:=0  F5B:=1
19791    144C        RF        CAD        RF   ROD         RG   JSL  CADA CTR       UNC   DEC TARG ADDR, OPA:=WORD AT TARGET ADDRESS;
19793                                                                                  CTR:=4 (NIBBLES TO GO), JUMP TO LOOP
19794           "
19795           "
19796           "         CAD8 & CAD9 WRITE THE LAST 1 - 4 DECIMAL DIGITS
19797           "         ( IN RE ), TO THE APPROPRIATE TARGET ADDRESS.
19798           "
19799           "
19800    144D   CAD8      RE   ADD  SWAB                 CTRS JSL  CAD9            EVEN  CBA0 => A0CB; JUMP IF UNFILLED NIBBLES EVEN
19802    144E        UBA  UBA  ADD  LSL            UBA  UBA  LINK             DCTR        SHIFT UBA,UBA LEFT 2 BITS; DEC DIGIT COUNT
19804    144F        UBA  UBA  ADD  LSL  RE         UBB  UBB  LINK                         RE:=0CBA (ZERO HAS BEEN SHIFTED IN)
19806    1450   CAD9      OPA  ADD  LRZ                  RF   ADD                  WRD  CTR0  UBA(8:8):=POSSIBLE EXTRA TARGET BYTE;
19808                                                                                  YRB:=TARGET ADDRESS, SKIP IF RE FULL
19809    1451        SP0A      ADD             NZRO  UBA  RE   ADD  SWAB       DATA UNC   SKIP IF RESULT NONZERO;
19811                                                                                  IF RE NOT FULL, WRITE EX. BYTE & DIGS TO MEM
19812    1452                  ADD             CCA        RE   ADD            DATA        COND CODE:= "=0" IF RESULT IS ZERO;
19814                                                                                  WRITE RE TO MEMORY IF FULL
```

```
RECORD   C.S.      ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19816           *
19817           *
19818           *
19819           *         SD24 --- WILL POP THE STACK TWO OR FOUR TIMES.
19820           *
19821           *                  IF CIR ( 11:1 ) := 0   ; DECREMENT BY 2, OR
19822           *                                  := 1   ; DECREMENT BY 4.
19823           *
19824           *
19825     1453  SD24 0010 CIR  ANDL                        ADD            EPP2       CHECK CIR ( 11:1 ); POP2
19827     1454                 ADD                   UBA   ADD            CRF  ZERO  ; SKIP IF CIR ( 11:1 ) := 0
19829     1455                 JSZ  NEXT      UNC           ADD            EPP2       JUMP FOR NEXT; POP TWO MORE IF SDEC=1
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19832           *
19833           *         ROUTINE TO DO BOUNDS CHECKING ON TARGET FIELD ----
19834           *
19835           *         ON ENTRY :
19836           *
19837           *                   RD := TARGET BYTE ADDRESS ( DB REL )
19838           *                   RH := TARGET DIGIT COUNT
19839           *                   F1 := 0, IF SPLIT STACK ( ELSE := 1 )
19840           *                   F3 := 0
19841           *
19842           *         ON EXIT  :
19843           *
19844           *                  SPOA := TARGET MSW ADDRESS
19845           *                  SP1B := TARGET WORD COUNT - 1
19846           *                    F1 := F3
19847           *
19848   1456  CKCD      RD   ADD  LSR  RH   NF1        RD   ADD          LBF5       RH:=DB REL WORD ADDR; F5B:=LSB(BYTE ADDR)
19850   1457            SM   ADD             SF3A       RH   ADD  LSR  SP1B LBF5 NF5B  UBA:=SM, F3A:=1 IF F1=1;
19852                                                                                 SP1B:=BYTE CNT - 1, SET F5B IF SIGN WD FULL
19853   1458      UBA   DB   SUB             NF1        UBB  INC        SP1B LBF5       UBA:=SM-DB, SKIP IF SPLIT STACK;
19855                                                                                 INC SP1B IF LSB(BYTE ADDR)=1
19856   1459      RH    UBA  CAD             CTF1            ADD                      F1:=(PTR > (SM-DB)) AND (NOT SPLIT STACK)
19858   145A            RD   ADD  LSR        F1HB       SP1B ADD  LSR  SP1B         UBA:=PTR+32K IF F1; SP3B:=WORD COUNT - 1
19860   145B      UBA   DB   ADD       SP0A        UBB  SM   RSUB                     SP0A:=MSW ADDR; UBB:=SM-SP3B
19862                                                                                  TICB TO SLOW DOWN RSB                (1000 )
19863   145C      UBA   DL   BNDE                  UBB  UBA  BNDE                     (1010)
19865   145D                 ADD                      ADD                 RSB        BNDV IF MSW ADDR < DL IN USER MODE;
19867                                                                                 BNDV IF LSW ADDR > SM IN USER MODE
19868           *
19869           *
19870           *         ROUTINE TO DO BOUNDS CHECKING ON SOURCE FIELD ---
19871           *
19872           *         ON ENTRY :
19873           *
19874           *                   RG := SOURCE BYTE ADDRESS ( DB REL )
19875           *                   RH := SOURCE DIGIT COUNT
19876           *                   F3 := 0, IF SPLIT STACK ( ESLE := 1 )
19877           *
19878           *         ON EXIT  :
19879           *
19880           *                  SP4A := SOURCE MSW ADDRESS
19881           *                  SP3B := SOURCE WORD COUNT - 1
19882           *
19883   145E  CKAB      RG   ADD  LSR  RH             RG   ADD          LBF5       RH:=DB REL WORD ADDR; F5B:=LSB(BYTE ADDR)
19885   145F            SM   ADD                      RH   ADD  LSR  SP3B LBF5 NF5B  UBA:=SM;
19887                                                                                 SP3B:=BYTE CNT - 1, SET F5B IF SIGN WD FULL
19888   1460      UBA   DB   SUB             NF3A       UBB  INC        SP3B LBF5       UBA:=SM-DB, SKIP IF SPLIT STACK;
19890                                                                                 INC SP3B AND CORRECT F5B IF LSB(BYTE ADDR)=1
19891   1461      RH    UBA  CAD             CTF1            ADD                      F1:=(PTR > (SM-DB)) AND (NOT SPLIT STACK)
19893   1462            RG   ADD  LSR        F1HB  SP3B      ADD  LSR  SP3B         UBA:=PTR+32K IF F1; SP3B:=WORD COUNT - 1
19895   1463      UBA   DB   ADD       SP4A        UBB  SM   RSUB                     SP4A:=MSW ADDR; UBB:=SM - SP3B
19897                                                                                  TICB TO SLOW DOWN RSB                (1000 )
19898   1464      UBA   DL   BNDE                  UBB  UBA  BNDE                     (1010)
19900   1465                 ADD                      ADD                 RSB        BNDV IF MSW ADDR < DL IN USER MODE;
19902                                                                                 BNDV IF LSW ADDR > SM IN USER MODE
```

```
 RECORD   C.S          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19904           $NOWARN
19905           *
19906           *
19907           *          DECIMAL TO ASCII CONVERSION ---
19908           *
19909           *          ON ENTRY :
19910           *
19911           *                     SR := 3
19912           *                     RA := SOURCE BYTE ADDRESS ( DB REL )
19913           *                     RB := SOURCE DIGIT COUNT
19914           *                     RC := TARGET BYTE ADDRESS ( DB REL )
19915           *
19916           *             TARGET ADDR := PACKED DECIMAL CONVERTED TO ASCII.
19917           *             OP CODE BIT 9 & 10  AFFECT THE SIGN OF THE RESULT
19918           *             AS FOLLOWS :-
19919           *
19920           *                BIT 9  BIT 10      TARGET SIGN
19921           *
19922           *                  0       0        SAME AS SOURCE
19923           *                  0       1        NEG IF SOURCE IS NEG, ELSE UNSIGNED
19924           *                  1       X        UNSIGN
19925           *
19926           *             AN UNSIGNED RESULT IS CONSIDERED POSITIVE WHEN SETTING
19927           *             THE CONDITION CODE, RESULTING THE CCG OR CCE.  LEADING
19928           *             ZEROS ARE CONVERTED TO ZEROS.  THE SOURCE IS ASSUMED
19929           *             TO HAVE THE SAME NUMBER OF DIGITS AS THE TARGET.  IF
19930           *             THE DIGIT COUNT IS ZERO, ONLY SDEC IS PERFORMED.
19931           *
19932           *
19933    1466   CVDA       RB    JSB  SD13      ZERO  RB   SP1B JSB  DA17          CRRY  DEC STACK & EXECUTE NEXT INSTR IF RB=0;
19935                                                                                     TRAP IF RB>=29
19936    1467              DB    ADD            NFSS  Z    DB   SUB            CF1   NCRY  UBA:=DB, SKIP IF NOT SPLIT BANKS;
19938                                                                                     F1:=0, SKIP IF Z<DB
19939    1468              RB    SUB            CF1   UBA  DL   SUB            CTF1        UBA:=-RB; F1 SET IF NOT SPLIT STACK;
19941                                                                                     ( (NFSS) AND (Z>=DB) AND (DB>=DL) )
19942    1469        RB          CAD                  RC   UBA  XFRR      RD              UBA:=RB+1; RD:=RC (TARGET BYTE ADDRESS)
19944    146A        RB    UBA   ADD       RH               SREG JSL  CKCD SP2B      UNC   RH:=TARGET NIBBLE COUNT (2RB-1);
19946                                                                                     SP2B:=-RB (DIGIT CNT), GO CHECK TARG ADDRS
19947    146B        SPOA        ADD            ROBD       RA   ADD       RG              OPB:=FRST TARG WD; RG:=RA (SOURCE BYTE ADDR)
19949    146C              RB    ADD       RH               JSL  CKAB CTX       UNC   RH:=RB(SOURCE NIBBLE CNT), COND CODE:= ">0";
19951                                                                                     CTX:=0, GO CHECK SOURCE ADDRESSES
19952    146D                    ADD       RD               SP4A INC       XR0  ROAD    RD:=0(CCE IND); OPA:=SEC SRCE WD, XRB0:=ADDR
19954    146E        E700        ADDL      XR3        3030       ADDL      XR3           XRA3:=E700 (FOR USE AFTER DA1-DA3 LOOP);
19956                                                                                     XRB3:=3030 (FOR USE IN DA1-DA3 LOOP)
19957    146F              SP4A  ADD             ROBD       UBB  ADD       RE              OPB:=FRST SRCE WD; RE:=3030(EXTA ASCII BITS)
19959    1470                    ADD                  RC   OPB  XFRR            CF2   EVEN  ; F2:=0("L ASCII BYTE"), SKIP/TARG ADDR EVEN
19961    1471              RC    ADD            EVEN       SREG ADD  LLZ       SF2         SKIP IF TARGET ADDRESS EVEN;
19963                                                                                     IF TARG ADDR ODD, F2:=1 ("RIGHT ASCII BYTE")
19964    1472        0030  UBB   IORL           RE         0004       ADDL      CTR         IF TARG ADDR ODD, RE(0.8):=EXTRA BYTE;
19966                                                                                     CTR:=4 (DIGITS FROM SOURCE WORD TO GO)
19967    1473        0001        ADDL           RG               RA   ADD             CF1   EVEN  RG:=1(FOR CTR CHECKS); F1:=0,SRCE ADDR EVEN?
19969    1474        FFF6        ADDL           XR0              OPB  ADD  RLZ  RH         UNC   XRA0:=-10 (USED FOR DIGIT VALIDITY CHECKS);
19971                                                                                     IF SOURCE ADDRESS ODD, RH(0.8):=OPB(8.8)
19972    1475        SPOA        ADD            WRD        SREG ADD       RH        UNC   YRA:=TARG ADDR, IF SRCE ADDR EVEN, RH =OPB
19974    1476              RB    JSB  DA1        ODD   0002       ADDL      CTR           JUMP IF DIGIT COUNT ODD;
19976                                                                                     IF SOURCE ADDRESS ODD, CTR:=2 (DIGITS TO GO)
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

19977   1477       RG   INC                      RH   ADD  SWAB RH   DCTR
19978   1478       RG   JSBI DA2       MEDJ  UBA  CTRS SUB              CTF1        UBA:=RG+1=2, JUMP (MEDIUM, SO UBA:=DIGIT);
19980                                                                               IF DIG COUNT EVEN, RH:=SWAB(RH), CTR:=CTR-1
19981   1479       000F RH   ANDL                   UBA  CTRS SUB                   UBA(12:4):=FRST DIG; F1:=1 IF LAST DIG IN WD
19983           *
19984           *
19985           *          DA1-3 COMPRISE THE MAIN " INNER LOOP " OF THE CVDA
19986           *          ROUTINE.  EACH DECIMAL SOURCE DIGIT IS LOADED OR SHIFTED
19987           *          INTO THE UPPER 4 BITS OF RH, SHIFTED LEFT 4 BIT POSITIONS
19988           *          INTO SP4A, CHECKED FOR VALIDITY, AND OR'ED INTO RE
19989           *          ( WHICH CONTAINS THE EXTRA ASCII BITS ) TO OBTAIN ITS
19990           *          ASCII REPRESENTATION.  WHEN RE IS FULL, IT IS WRITTEN
19991           *          TO THE APPROPRIATE TARGET LOCATION.
19992           *
19993           *
19994   147A    DA1                ADD  LSL              RH   RH   LINK                 UBA,UBB:=RH SHIFTED LEFT 2 BITS
19996   147B       UBA  UBA  ADD  LSL  SP4A       UBB  UBB  LINK      RH             UBA,SP4A(12:4):=DEC DIG; RH(0:12):=RH(4:12)
19998   147C    DA2 UBA  XRO  JSB  DA15      CRRY  UBA  RD   IOR       RD   DCTR      TRAP IF DIGIT >= 10;
20000                                                                               CAPTURE NONZERO BITS IN RD, CTR:=CTR-1
20001   147D       RREG RE   ADD  SWAB RE   NF2            SP2B JSBI DA4  SP2B      ZERO  RE:=SWAB(RE+DIGIT), SKIP IF LEFT BYTE;
20003                                                                               SP2B:=SP2B+1 (NEG DIG CNT), JUMP IF LAST DIG
20004   147E       UBA  ADD  SWAB       DATA       CTRS ADD       RF   SF2  NF2   IF RIGHT BYTE, WRITE ASCII WORD TO TARGET;
20006                                                                               RF:=CTR, F2:=1, SKIP IF NOT RIGHT BYTE
20007   147F            ADD                         XR3  ADD       RE   CF2       ; IF RIGHT BYTE, RE:=3030, F2:=0
20009   1480    DA3 RG   RF   SUB            CTF1            JSL  DA1            NF1  F1:=1 IF NEXT DIGIT LAST IN SOURCE WORD;
20011                                                                               JUMP IF MORE DIGITS LEFT IN SOURCE WORD
20012   1481       OPA  ADD       RH   CF1       XRO  INC       XRO  ROAD        RH:=NEXT SOURCE WORD;
20014                                                                               XRBO:=XRBO+1(SRCE ADDR),OPA:=SRCE WD AFT NXT
20015   1482            JSB  DA1       UNC   0004       ADDL      CTR             JUMP BACK TO LOOP; CTR:=4 (DIGITS TO GO)
20017           *
20018           *
20019           *          DA4-7 GENERATE THE APPROPRIATE ( SIGNED ) REPRESENTATION
20020           *          FOR THE LOW ORDER ASCII, WRITE THE LAST 1-2 ASCII DIGITS
20021           *          ( WHICH IS IN RE ), TO THE APPROPRIATE TARGET ADDRESS,
20022           *          AND SET THE CONDITION CODE.
20023           *
20024           *
20025   1483    DA4 SP0A SP1B ADD            ROBD  F000 RH   ANDL      RH             OPB:=DATA AT LAST TARG LOC; RH(0:4):=SIGN
20027   1484       RB        ADD            CCA   F000       ADDL
20028   1485       0040 CIR  ANDL                 UBB  RH   JSBS DA6            ZERO  UBA(9):=CIR(9); JUMP IF SIGN=F (UNSIGNED)
20030   1486            UBA  JSB  DA6       NZRO  D000 RH   SUBL      RF             JUMP IF CIR(9):=1 (UNSIGNED); RF:=D000-RH
20032   1487       1900 RE   ADDL      RE                    CAD            CCA     ASSUME NEG, ADJUST RE; COND CODE:= "<0"
20034   1488       0020 CIR  ANDL                 RF   JSL  DA5            ZERO  UBA(10):=CIR(10); JUMP IF SIGN=D (MINUS)
20036   1489       RE   XR3  ADD       RE   CCA        UBA  JSL  DA6            NZRO  ASSUME POS/UNSGND, ADJ RE, COND CODE:= ">0";
20038                                                                               JUMP IF CIR(10):=1 (UNSIGNED)
20039   148A       1000 UBA  ADDL      RE         3100       ADDL                    ASSUME POS/SIGNED, ADJUST RE; UBB:=3100
20041   148B    DA5     SP4A JSB  DA6       NZRO  UBB  RE   SUB                NCRY  JUMP IF LAST DIGIT NOT ZERO; SKIP IF -0
20043   148C       3400 RE   ADDL      RE         3B00 RE   ADDL      RE             ADJUST RE IF -0; ADJUST RE IF +0
20045   148D    DA6             ADD                         JSL  DA7  SP2B       F2    ; JUMP IF LAST TARGET WORD FULL, SP2B:=0
20047   148E       RE   ADD  LRZ  RE                  OPB  ADD  RLZ  SP2B            SAVE ASCII BYTE IN RE(8:8), F1:=NF2;
20049                                                                               GET EXTRA BYTE IN SP2B(0:8)
20050   148F    DA7             ADD                         RD   ADD            NZRO  ; SKIP IF RESULT NOT ZERO
20052   1490       RE   SP2B ADD  SWAB       DATA             ADD            CCA     WRITE DATA TO TARGET ADDRESS;
20054                                                                               COND CODE:= "=0" IF RESULT IS ZERO
20055           $WARN
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20057            *
20058            *
20059            *
20060            *
20061            *          SD13 --- WILL POP THE STACK ONCE OR THREE TIMES.
20062            *
20063            *                   IF CIR ( 11:1 ) := 0   , DECREMENT BY 1, OR
20064            *                                   := 1   , DECREMENT BY 3.
20065            *
20066            *
20067     1491   SD13 0010 CIR  ANDL                          ADD             EPOP        CHECK CIR ( 11:1 ); POP 1
20069     1492                 ADD                      UBA  ADD             CRF  ZERO   ; CRF, SKIP IF CIR ( 11:1 ) := 0
20071     1493                 JSZ  NEXT      UNC           ADD             EPP2        DO NEXT; POP 2
20073            *
20074            *
20075            *
20076            *          TRAP ROUTINES FOR CVAD ---
20077            *
20078            *          1. CF1 IF INVALID ASCII DIGIT.
20079            *          2. SF1 IF INVALID DECIMAL OPERAND LENGTH.
20080            *
20081            *
20082     1494   AD14           ADD            CF1           JSL  BDER            UNC   CF1; JSB BDER FOR INVALID ASCII DIGIT
20084     1495   AD17           ADD            SF1           JSL  BDER            UNC   SF1; JSB BDER FOR INVALID DECIMAL OPER LEN
20086            *
20087            *
20088            *
20089            *          TRAP ROUTINES FOR CVDA ---
20090            *
20091            *          1. CF1 IF INVALID DECIMAL DIGIT.
20092            *          2. SF1 IF INVALID DECIMAL OPERAND LENGTH.
20093            *
20094            *          IF USER TRAP IS DISABLED, SET OVERFLOW & JSB SD13.
20095            *
20096            *
20097     1496   DA15           ADD                           ADD             CF1  UNC    ; CF1, SKIP
20099     1497   DA17 2000 STA  ANDL                          ADD             SF1         CHECK STA ( 2:1 ), USER TRAP; SF1
20101     1498                 UBA  JSB  *+2          ZERO    ADD             CRF         JSB IF USER TRAP DISABLED; CRF
20103     1499                      JSZ  TRPD        UNC     JSZ  TRPR            F1    TRAP TARGETS
20105     149A        0800 STA  IORL      STA                JSB  SD13            UNC   SET OVERFLOW; DO STACK ADJUSTMENT
```

```
 RECORD   C S          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20108            *
20109            *
20110            *
20111            *          CVBD ( CONVERT 4 BINARY WORDS ) ---
20112            *
20113            *          ON ENTRY :
20114            *
20115            *             BINARY WORDS ARE STORED IN XRA6 - XRA9.
20116            *
20117            *          ON EXIT  :
20118            *
20119            *             DECIMAL DATA ARE STORED INTO TWO SETS AT
20120            *             XRA5 - XRA9, & XRB4 - XRB8.
20121            *
20122            *          1. FOUR BINARY WORDS CAN BE CONVERTED TO A
20123            *             MAXIMUM OF 20 DECIMAL DIGITS.
20124            *          2. THE ALGORITHM IS SAME AS CONVERTING 3 BINARY
20125            *             WORDS TO DECIMAL DIGITS.
20126            *
20127            *
20128     149B   BD4          XR8  ADD                          ADD             PSHR        SETUP TO DIVIDE BY 100M
20130     149C                XR8  ADD              UBA         ADD       SP2B              DIVIDEND ( SP2B, SP0A, SP3B )
20132     149D                XR7  ADD       SP0A       UBA    JSL  D100 SP3B       UNC    DO DIVISION
20134     149E                UBB  ADD       SP0A       UBA    ADD       SP2B              REMAINDER ( SP0A, SP2B )
20136     149F                XR9  ADD                  SP3B    ADD       XR0               QUOTIENT := ( XRB0, XRA0 );
20138     14A0                SP4A ADD       XR0        UBA    JSL  D100 SP3B       UNC    . DO DIVISION
20140     14A1                UBA  ADD       SP0A       001E    ADDL      CTR             REMAINDER := ( SP0A, SP2B ); REPEAT LOOP
20142     14A2                SP4A ADD       RG              SP2B JSL  BTOD SP3B       UNC    CONVERT REMAINDER TO DECIMAL ( LSD'S )
20144     14A3          UBA   XR0  XFRS      SP0A            UBB  ADD       BKX5            SETUP FOR NEXT DIVISION
20146     14A4          RREG  UBB  XFRS      XR9             RG   ADD       SP3B
20147     14A5          RREG        ADD       XR8             XR0  JSL  D100 SP2B       NCRY   DIVIDEND := ( SP2B, SP0A, SP3B )
20149     14A6                UBA  ADD       SP0A       001E    ADDL      CTR             REMAINDER := ( SP0A, SP2B )
20151                                                                                      REPEAT LOOP COUNT
20152     14A7                SP4A ADD       RG              SP2B JSL  BTOD SP3B       UNC    CONVERT REMAINDER TO DECIMAL DIGITS
20154     14A8                UBA  ADD       XR6             UBB  ADD       XR6
20155     14A9                UBB  ADD       XR7        000E    ADDL      CTR
20156     14AA                RG   ADD       SP0A            JSL  BTOD SP3B       UNC    CONVERT THE QUOTIENT TO DECIMAL ( MSD'S )
20158     14AB                XR6  ADD                      UBB  ADD       XR4  POPR
20159     14AC          UBB   XR8  XFRS      SP4A            UBA  ADD       XR5
20160     14AD          RREG  XR13 XFRR      XR5             UBA  ADD       XR7       NF4B
20161     14AE                SREG CSR            HBF2            ADD                 RSB    SET F2 IF SIGN MASK := 000F; RETURN IF F4
20163            *
20164            *
20165     14AF                     ADD                  BKX5 JSL  BD3S XR8        UNC    STORE OUT DATA
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20168           *
20169           *
20170           *
20171           *          CVBD ( CONVERT 5 BINARY WORDS ) ---
20172           *
20173           *          ON ENTRY :
20174           *
20175           *             BINARY WORDS ARE STORED IN XRA5 - XRA9.
20176           *
20177           *          ON EXIT  :
20178           *
20179           *             DECIMAL DATA ARE STORED INTO TWO SETS AT
20180           *             XRA3 - XRA9, & XRB2 - XRB8.
20181           *
20182           *          1. FIVE BINARY WORDS CAN BE CONVERTED TO A
20183           *             MAXIMUM OF 25 DECIMAL DIGITS.
20184           *          2. REFER TO BD3 FOR ALGORITHM.
20185           *
20186           *
20187    14B0   BD5        XR5  ADD                          ADD
20188    14B1               XR7  ADD                    UBA   ADD        SP2B
20189    14B2               XR6  ADD       SPOA         UBA   JSL  D100  SP3B       UNC
20190    14B3               UBB  ADD       SPOA         UBA   ADD        SP2B
20191    14B4               XR8  ADD                SP3B       ADD        RG
20192    14B5               SP4A ADD       XR7          UBA   JSL  D100  SP3B       UNC
20193    14B6               UBB  ADD       SPOA         UBA   ADD        SP2B
20194    14B7               XR9  ADD                          ADD
20195    14B8               SP4A ADD       XR8          UBA   JSL  D100  SP3B       UNC
20196    14B9               UBA  ADD       SPOA    001E       ADDL       CTR
20197    14BA               SP4A ADD       XR9          SP2B  JSL  BTOD  SP3B       UNC   CONVERT THE REMAINDER TO BE THE LSD'S
20199    14BB          UBA  RG   XFRS      XR6          UBB   ADD        BKX4 SF4B
20200    14BC          RREG      ADD       RF                 JSL  BD4               UNC   LSD'S := ( RF, BKX4 ); CONVERT THE 4 WORDS
20202    14BD                    ADD                    XR5   ADD        XR3
20203    14BE               UBA  ADD       XR4                ADD
20204    14BF                    ADD                    SP4A  JSB  BD6R  XR5        F5B   RETURN IF FROM BD6
20206           *
20207           *          SAVE DECIMAL DIGITS
20208           *
20209    14C0          UBB  XR5  XFRR      XR6          BKX4  ADD        RG   CF4B
20210    14C1               SREG ADD       XR3          XR6   ADD        XR4
20211    14C2               UBB  ADD       XR5          UBA   ADD        XR2
20212    14C3               RF   ADD       XR8          BKX5  ADD        XR6
20213    14C4               UBB  ADD       XR7          UBA   ADD        XR7
20214    14C5               RG   ADD       XR9          RG    JSL  BD3S  XR8        UNC   ; STORE OUT DATA
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20217           *
20218           *
20219           *
20220           *          CVBD ( CONVERT 6 BINARY WORDS ) ---
20221           *
20222           *          ON ENTRY :
20223           *
20224           *             BINARY WORDS ARE STORED IN XRA4 - XRA9.
20225           *
20226           *          ON EXIT  :
20227           *
20228           *             DECIMAL DATA ARE STORED INTO TWO SETS AT
20229           *             XRA2 - XRA9, & XRB1 - XRB8.
20230           *
20231           *          1. SIX BINARY WORDS CAN BE CONVERTED TO A
20232           *             MAXIMUM OF 29 DECIMAL DIGITS.
20233           *          2. REFER BD3 FOR ALGORITHM.
20234           *
20235           *
20236    14C6   BD6        XR4  ADD                              ADD
20237    14C7               XR6  ADD                         UBA  ADD        SP2B SF5B        ; SET F5 FOR BD5 RETURN
20239    14C8               XR5  ADD       SPOA              UBA  JSL  D100  SP3B       UNC
20240    14C9               UBB  ADD       SPOA         UBA       ADD        SP2B
20241    14CA               XR7  ADD                    SP3B       ADD
20242    14CB               UBB  ADD       XR5               UBA  ADD        SP3B
20243    14CC               SP4A ADD       XR6                    JSL  D100       UNC
20244    14CD               UBB  ADD       SPOA              UBA  ADD        SP2B
20245    14CE               XR8  ADD                              ADD
20246    14CF               SP4A ADD       XR7               UBA  JSL  D100  SP3B       UNC
20247    14D0               UBB  ADD       SPOA              UBA  ADD        SP2B
20248    14D1               XR9  ADD                              ADD
20249    14D2               SP4A ADD       XR8               UBA  JSL  D100  SP3B       UNC
20250    14D3               UBA  ADD       SPOA         001E      ADDL       CTR
20251    14D4               SP4A ADD       XR9               SP2B JSL  BTOD  SP3B       UNC   CONVERT THE REMAINDER TO BE THE LSD'S
20253    14D5               UBA  ADD       RH                UBB  JSB  BD5   SP1B       UNC   LSD'S := ( RH, SP1B ); JSB TO CONVERT
20255                                                                                          THE 5 BINARY WORDS
20256           *
20257           *          SAVE DECIMAL DATA
20258           *
20259    14D6   BD6R       XR8  ADD                         0080 XR31 ADDL       CTR
20260    14D7          UBA  XR6  XFRS      REGN                   XR6  ADD        XR2  ICTR
20261    14D8          RREG UBB  XFRS      REGN                   UBA  ADD        XR1  ICTR
20262    14D9          RREG      ADD       REGN                   BKX5 ADD        XR4  ICTR
20263    14DA               UBB  ADD       REGN                   UBA  ADD        XR3  ICTR
20264    14DB               RF   ADD       REGN                   BKX4 ADD        XR6  ICTR
20265    14DC               UBB  ADD       REGN              UBA       ADD        XR5  ICTR
20266    14DD               RH   ADD       REGN                   SP1B ADD        XR8  ICTR
20267    14DE               UBB  ADD       REGN                   UBA  ADD        XR7
20268           *
20269    14DF                    ADD                              RA   JSB  BDTR            ODD   JSB IF CONVERTING 7 BINARY WORDS
20271    14E0                    ADD                                   JSL  BD3S            UNC   ; STORE OUT DATA
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20274           *
20275           *
20276           *
20277           *         CVBD ( CONVERT 7 BINARY WORDS ) ---
20278           *
20279           *         ON ENTRY :
20280           *
20281           *            BINARY WORDS ARE STORED IN XRA3 - XRA9.
20282           *
20283           *         ON EXIT  :
20284           *
20285           *            DECIMAL DATA ARE STORED INTO TWO SETS AT
20286           *            XRA2 - XRA9, & XRB1 - XRB8.
20287           *
20288           *         1. SEVEN BINARY WORDS CAN BE CONVERTED TO
20289           *            32 DECIMAL DIGITS.
20290           *         2. REFER BD3 FOR ALGORITHM.
20291           *
20292           *
20293    14E1   BD7       XR4  ADD       SP0A            UBA  JSL  D100 SP3B      UNC
20294    14E2              UBB  ADD       SP0A            UBA  ADD       SP2B
20295    14E3              XR6  ADD                  SP3B      ADD
20296    14E4              UBB  ADD       XR4             UBA  ADD       SP3B
20297    14E5              SP4A ADD       XR5                  JSL  D100           UNC
20298    14E6              UBB  ADD       SP0A            UBA  ADD       SP2B
20299    14E7              XR7  ADD                           ADD
20300    14E8              SP4A ADD       XR6             UBA  JSL  D100 SP3B      UNC
20301    14E9              UBB  ADD       SP0A            UBA  ADD       SP2B
20302    14EA              XR8  ADD                           ADD       XR31
20303    14EB              SP4A ADD       XR7             UBA  JSL  D100 SP3B      UNC
20304    14EC              UBB  ADD       SP0A            UBA  ADD       SP2B
20305    14ED              XR9  ADD                           ADD
20306    14EE              SP4A ADD       XR8             UBA  JSL  D100 SP3B      UNC
20307    14EF              UBA  ADD       SP0A       001E      ADDL      CTR
20308    14F0              SP4A ADD       XR9             SP2B JSL  BTOD SP3B      UNC
20309    14F1        UBA         ADD       XR10            UBB  JSB  BD6  XR15      NCRY
20310           *
20311           *         SAVE DECIMAL DATA IN BOTH EXTENDED REGISTERS
20312           *
20313    14F2   BDTR      XR10 ADD       XR8             XR15 ADD       RG
20314    14F3        0005      ADDL      RH              UBA  ADD       XR7
20315    14F4              REGN ADD                       RG   ADD       XR8  DCTR
20316    14F5        RH        CAD       RH    CTF1      UBA  JSB  *-1  REGN      F1
20317    14F6              RG   ADD       XR9                JSL  BD3S            UNC   STORE OUT DATA
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20320           *
20321           *
20322           *
20323           *         ROUTINE TO CONVERT TWO'S-COMPLEMENT BINARY WORDS
20324           *         TO PACKED-DECIMAL DIGITS.
20325           *
20326           *         ON ENTRY :
20327           *
20328           *                     SP0A :=  UPPER BINARY WORD
20329           *                     SP3B :=  LOWER BINARY WORD
20330           *                      CTR :=  # OF BINARY BIT - 1
20331           *
20332           *         ON EXIT  :
20333           *
20334           *                      UBA :=  UPPER DECIMAL WORD
20335           *                      UBB :=  LOWER DECIMAL WORD
20336           *
20337           *
20338    14F7   BTOD            ADD                          REPC                  ; REPEAT LOOP
20340    14F8          UBA  UBA  BTOD                 UBB  UBB  LINK          DCTR CTR0  CONVERT BINARY TO DECIMAL
20342    14F9          UBA       ADD                  UBB       ADD               RSB   DECIMAL RESULT ( UPPER, LOWER ); RETURN
20344           *
20345           *
20346           *
20347           *         ROUTINE TO DIVIDE BY 100M FOR CVBD
20348           *
20349           *           ( RRGA, RRGB, SP0A, SP3B ) / ( XRA15, XRB14 )
20350           *
20351           *         ON ENTRY :
20352           *
20353           *                     SP0A :=  UPPER BINARY WORD
20354           *                     SP3B :=  LOWER BINARY WORD
20355           *                    XRA15 :=  !05F5
20356           *                    XRB14 :=  !E100
20357           *
20358           *         ON EXIT  :
20359           *
20360           *                     SP3B :=  UPPER QUOTIENT
20361           *                     SP4A :=  LOWER QUOTIENT
20362           *
20363           *
20364    14FA   D100            ADD       SP4A CF1        SP2B REPN              20    CF1: SETUP REPEAT LOOP
20366    14FB          UBA  XR15 DVSB                 UBB  XR14 LINK          DCTR CTR0  DIVIDE ( RRGA, RRGB ) BY ( XRA15, XRB14 )
20368    14FC          UBA       ADD  LSR             UBB       LINK      SP2B      RSB   REMAINDER := ( UBA, UBB )
20370                                                                                    QUOTIENT  := ( SP3B, SP4A )
20371    14FD                     ADD                               ADD                 NOP to keep system happy                (2555)
20373    14FE                     ADD                               ADD                 NOP to keep system happy                (2555)
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20377           %1500
20378           *
20379           *
20380           *
20381           *          DECIMAL LEFT SHIFT & DECIMAL NORMALIZING LEFT SHIFT ---
20382           *
20383           *          ON ENTRY :
20384           *                       SR := 3
20385           *                        X := SHIFT COUNT
20386           *                       RA := SOURCE DIGIT COUNT
20387           *                       RB := SOURCE BYTE ADDRESS ( DB REL )
20388           *                       RC := TARGET DIGIT COUNT
20389           *                       RD := TARGET BYTE ADDRESS ( DB REL )
20390           *                             ( VALID AFTER PULM )
20391           *
20392           *             TARGET ADDR := SOURCE ADDR SHIFTED LEFT X ( 11:5 ).
20393           *             THE SIGN IS PRESERVED WITH THE EXCEPTIONS THAT -0
20394           *             BECOMES +0 AND ALL NON-NEGATIVE SIGNS RESULT IN
20395           *             POSITIVE SIGNS.  IF ALL SIGNIFICANT DIGITS DO NOT
20396           *             FIT INTO THE TARGET FIELD, THE RESULT IS LEFT-
20397           *             TRUNCATED AND CARRY ( STA (5:1) ) IS SET.   HIGH
20398           *             ORDER NON-SIGNIFICANT DIGITS ARE FILLED WITH ZEROS.
20399           *
20400           *
20401    1500   SLD              JSZ  PULM      UNC         UBA  JSZ  TRP  XR5        NZRO  PULM FOR RD; TRAP IF CIR ( 9:1 ) := 1
20403    1501        001F        ADDL      SP4A                   JSL  CKLN XR7        UNC   MASK; CHECK FOR VALID OPERAND LENGTHS
20405    1502        X    SP4A AND       X                      JSL  CKBA XR8        UNC   SHIFT COUNT; CHECK FOR SOURCE/TARGET ADDR
20407    1503        RH        ADD       XR11           RG      JSL  SPLT XR11       F2    SOURCE WD-ADDR; CHECK SPLIT STACK IF PRIV
20409    1504        X         ADD       XR15                   JSL  MASK XR9        UNC   XRA15 := X; GET SOURCE/TARGET MASKS
20411    1505             CIR  JSB  NSLD       ODD         RA   ADD       RG   CCRY       JSB IF NSLD; SAVE RA, CLEAR CARRY
20413    1506   NSL1      XR13 ADD                                JSL  LEN  XR10       UNC   ; DETERMINE ACC LENGTH
20415    1507             XR11 ADD       SP0A           RG      JSL  FCHB XR29       UNC   SOURCE WD-ADDR; GET SOURCE DATA
20417    1508             XR15 ADD       RH        0002 SP4A ADDL                    EVEN  SHIFT COUNT; SKIP IF SOURCE SIGN MASK:=0F00
20419    1509             XR13 ADD                 RREG UBA  ADD       RH   LBF5       TARGET SIGN MASK; UBB := SHIFT COUNT + 2
20421    150A                  ADD                 RREG UBA  XFRS            SF4B EVEN  ; SKIP IF TARGET SIGN MASK := 0F00
20423    150B                  ADD       XR14      RREG RH   RSUB      RH   LBF5       XRA14 := 0; UBB := SHIFT COUNT - 2
20425    150C                  CAD       RG                  JSL  SRDX            UNC   -1;
20427    150D        RH        JSB  SLD3      POS   RH        CSR       XR12 HBF2       JSB IF EFFECTIVE SHIFT COUNT IS POS;
20429                                                                                    SET F2 IF NEGATIVE
20430           *
20431           *
20432           *          EFFECTIVE SHIFT COUNT IS NEGATIVE
20433           *
20434           *
20435    150E                  JSB  SRD2      ZERO  RG   CTRS JSB  SRD1 CTR        F5B   DO SHIFT RIGHT 1 DIGIT IF F5;
20437                                                                                    ESLE DO SHIFT RIGHT 2 DIGITS ( SLOW JUMP )
20438    150F                  JSB  SLDN      UNC             ADD                       END OF SHIFTING
20440           *
20441           *
20442    1510   SL5D      REGN ADD       XR14           XR13 JSBI SLST CTR        UNC   5-DIGIT SHIFT
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20445           *
20446           *
20447           *
20448           *         DECIMAL NORMALIZING LEFT SHIFT ---
20449           *
20450           *         THE ENTRY POINT FOR NSLD IS SLD.
20451           *           TARGET ADDR := SOURCE ADDR SHIFTED LEFT X ( 11:5 ).
20452           *           IF ALL SIGNIFICANT DIGITS WOULD NOT FIT INTO THE
20453           *           TARGET ADDR FIELD, TARGET ADDR := SOURCE ADDR LEFT
20454           *           JUSTIFIED, X := X - ACTUAL SHIFT COUNT, AND CARRY
20455           *           IS SET.  IF TARGET ADDR WILL NOT HOLD SOURCE ADDR
20456           *           EVEN WITH A SHIFT COUNT OF ZERO, NO DATA IS MOVED
20457           *           AND DECIMAL OVERFLOW IS SET.  LEADING ZEROS WILL
20458           *           BE FILLED AND THE SIGN WILL BE PRESERVED AS SLD.
20459           *
20460           *
20461     1511  NSLD F3FF STA  ANDL      STA                 JSL  SCNB          UNC   CLEAR OVERFLOW & CARRY;
20463                                                                                  DO SCAN FOR LEADING ZEROS
20464     1512       RH   XR15 ADD       SP4A       RH       JSB  NSL1 RG       ZERO  # OF SHIFT; JSB IF ZERO LENGTH
20466     1513       RC   RH   JSBS EE13      NEG   RC   UBA  JSBS NSL1         POS   JSB IF TARGET FIELD IS NOT LARGE ENOUGH;
20468                                                                                  OK TO DO SHIFT
20469     1514       RC   RG   SUB       XR15       RG   SP4A RSUB          SCRY      EFFECTIVE SHIFT COUNT; SET CARRY
20471     1515       UBB  UBA  SUB       X                   JSB  NSL1         UNC   REMAINING SHIFT COUNT; DO SHIFT
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20474            *
20475            *
20476            *
20477            *         DECIMAL RIGHT LEFT ---
20478            *
20479            *         ON ENTRY :
20480            *
20481            *                     SR  := 3
20482            *                      X  := SHIFT COUNT
20483            *                     RA  := SOURCE DIGIT COUNT
20484            *                     RB  := SOURCE BYTE ADDRESS ( DB REL )
20485            *                     RC  := TARGET DIGIT COUNT
20486            *                     RD  := TARGET BYTE ADDRESS ( DB REL )
20487            *                            ( VALID AFTER PULM )
20488            *
20489            *            TARGET ADDR := SOURCE ADDR SHIFTED RIGHT X ( 11:5 ).
20490            *            THE SIGN IS PRESERVED IN THE SAME WAY AS SLD.  ZEROS
20491            *            ARE SHIFTED IN FROM THE LEFT AND LEADING NON-
20492            *            SIGNIFICANT DIGITS ARE FILLED WITH ZEROS.
20493            *
20494            *
20495     1516   SRD              JSZ  PULM      UNC        UBA  JSZ  TRP  XR5       NZRO  PULM FOR RD; JSB IF CIR ( 9:1 ) := 1
20497     1517        001F        ADDL      SP4A                  JSL  CKLN XR7       UNC   SHIFT COUNT MASK; CHK FOR VALID LENGTHS
20499     1518        X    SP4A AND       X                     JSL  CKBA XR8       UNC   SHIFT COUNT; CHK OPERAND ADDRESSES
20501     1519             RH   ADD       XR11           RG     JSL  SPLT XR11      F2    SOURCE ADDR; CHK FOR SPLIT STACK IF PRIV
20503     151A        X         ADD       XR15                  JSL  MASK XR9       UNC   SHIFT COUNT; GET OPERAND MASKS( MSW & SIGN )
20505     151B             XR13 ADD                        RA     JSL  LEN  RG        UNC   ; DETERMINE TARGET LENGTH
20507     151C             XR11 ADD       SP0A                  JSL  FCHB           UNC   SOURCE WD-ADDR; GET SOURCE DATA
20509     151D             XR15 ADD       RH         FFFE SP4A  ADDL                EVEN  SHIFT COUNT;
20511                                                                                      SKIP IF SOURCE SIGN MASK := 0F00
20512     151E             XR13 ADD                        RREG UBA  ADD       RH   LBF5  TARGET SIGN MASK; UBB := SHIFT COUNT - 2
20514     151F        0002      ADDL                       UBA       ADD                EVEN  2; SKIP IF TARGET SIGN MASK := 0F00
20516     1520             XR14 ADD       XR14           UBA  RH   ADD       RH   LBF5  XR14 := 0; UBB := SHIFT COUNT + 2
20518     1521        0005      ADDL      RG             0088 BKX3 SUBL      CTR        ; MSW REGN POINTER
20520     1522        RH        JSB  SRD3      POS       RH        CSR       XR12 HBF2  JSB IF SHIFT COUNT >= 0; SET F2 IF ODD
20522            *
20523            *         EFFECTIVE SHIFT COUNT IS NEGATIVE
20524            *
20525     1523        0089      ADDL                            RA   JSB  SLD1 SP1B      F5B   ; DO LEFT SHIFT 1 DIGIT IF F5
20527     1524             ADD                             UBA  BKX3 JSBS SLD2 CTR       NF5B  ; DO LEFT SHIFT 2 DIGITS IF NF5
20529     1525             ADD                                  JSB  SRDN              UNC   ; END OF SHIFTING
20531            *
20532            *
20533     1526   SR5D       JSB  SRST      UNC        0088 BKX3 SUBL      CTR                5-DIGIT SHIFT
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20536           *
20537           *
20538           *
20539           *
20540   1527  SRD3 RH         ADD  LSR  XR15 CF1         RA   JSB  SRD5 SP1B      F2    SHIFT COUNT; SRD5 IF SHIFT COUNT IS ODD
20542   1528                 JSB  SRD2       UNC         XR12 JSB  SRD7           EVEN  DO SHIFT RIGHT 2 DIGITS; JSB IF 00
20544   1529                 ADD                             JSB  SRD7           UNC   JSB SRD7
20546           *
20547           *
20548   152A  SRD5       XR15 JSB  SR3D       ODD         XR12 JSB  SRD1           EVEN  DO 3-DIGIT SHIFT; DO 1-DIGIT SHIFT
20550           *
20551           *
20552   152B       RG   RH   JSBS SR5D       ZERO            ADD                        5-DIGIT SHIFT
20554   152C  SRD7       XR15 ADD  LSR  RH   CF1   0089      ADDL      CTR              SHIFT COUNT; LSW REGN POINTER
20556           *
20557           *
20558   152D  SRD4 0089       ADDL       SP4A                BKX3 INC       RG               ; UBB := ACC WD LEN
20560   152E       RH         JSB  SRDN       ZERO  RH            CAD       RH   DCTR       DONE SHIFTING; DEC REGN POINTER
20562   152F       RG         CAD        RG   CTF1               CAD       ICTR ICTR       SET F1 IF DONE; -1, ICTR
20564   1530            REGN  ADD        REGN        UBB  CTRS  JSB  *-1  CTR       F1    STORE SHFITED DATA BACK ; LOOPBACK
20566   1531                  ADD                             SP4A JSB  *-4  CTR       UNC   DO ANOTHER 4-DIGIT SHIFT
20568           *
20569           *
20570   1532  SRDN 000F CIR  ANDL                  000C        ADDL                       MASK CIR ( 12:4 ); MASK
20572   1533       UBA  UBB  JSBS MPYS       ZERO  0005        ADDL                       FOR MPYD RETURN; MASK
20574   1534       RREG UBB  JSBS MPYS       ZERO  0089  BKX3  SUBL      CTR           FOR CVDB RETURN; MSW REGN POINTER
20576           *
20577           *          STORE OUT DATA
20578           *
20579   1535  SRST           ADD             CF1         XR11 JSL  STOA SP2B      UNC   STORE DATA TO TARGET,UBB := TARGET WD-ADDR
20581   1536                 ADD                             JSL  S024           UNC   DO SDEC
```

RECORD NO	C S ADDR	LABL	********** ALU A ********** RREG	SREG	FUNC	SFNC	STOR	SPSK	********** ALU B ********** RREG	SREG	FUNC	SFNC	STOR	SPEC	SKIP	COMMENT
20584		*														
20585		*														
20586		*														
20587		*														
20588	1537	SLD3	RH		ADD	LSR	XR15	CF2		XR29	JSB	SLD5	SP1B		F2	SHIFT COUNT; SLD5 IF SHIFT COUNT IS ODD
20590	1538				JSB	SLD2		UNC		XR12	JSB	SLD7			EVEN	JSB TO SHIFT LEFT 2 DIGITS; JSB IF 00
20592	1539				ADD						JSB	SLD7			UNC	
20594		*														
20595		*														
20596	153A	SLD5		XR15	JSB	SL3D		ODD		XR12	JSB	SLD1			EVEN	LEFT SHIFT 3-DIGIT; LEFT SHIFT 1-DIGIT
20598	153B		RH	RG	JSBS	SL5D		ZERO		XR13	ADD		CTR			DO SHIFT 5-DIGIT IF ZERO; REGN POINTER
20600	153C	SLD7		XR15	ADD	LSR	RH			XR13	ADD		CTR	CF4B		SHIFT COUNT;
20602	153D			UBA	JSB	*+7		ZERO			ADD					JSB TO SHIFT LEFT 3 DIGITS;
20604		*														
20605		*														
20606	153E	SLD4			ADD					BKX3	ADD		RG	ICTR		; ACC WD LENGTH, ICTR
20608	153F			REGN	ADD			F2		RH	JSB	SLDN			ZERO	SKIP IF FROM SL3D; SLDN IF DONE
20610	1540		UBA	XR14	IOR		XR14				ADD					SAVE LOST DIGIT;
20612	1541	SL3J	RG		CAD		RG	CTF1			ADD			DCTR		SET F1 IF DONE ALL SHIFTING; DEC CTR
20614	1542			REGN	ADD		REGN			CTRS	JSBI	*-1	CTR		F1	STORE SHIFTED WORDS; LOOPBACK IF NOT DONE
20616	1543		RH		CAD		RH	CF2		XR13	JSB	*-5	CTR		UNC	DEC INT(X/4); NEXT 4-DIGIT SHIFT
20618		*														
20619		*														
20620	1544	SLDN	000F	CIR	ANDL				000C		ADDL					MASK CIR (12:4); MASK
20622	1545		UBA	UBB	JSBS	MPYS		ZERO	0089	BKX3	SUBL		CTR			FOR MPYD; MSW REGN POINTER
20624		*														
20625		*														
20626	1546	SLST		XR14	ADD			ZERO			ADD					SKIP IF NOT DIGIT LOST;
20628	1547				ADD			SF3A		XR11	JSL	STOA	SP2B		UNC	F3 IF DIGIT LOST; GO STORE RESULT, TARG ADDR
20630	1548				ADD			NF3A	0400		ADDL					SKIP IF NO DIGIT LOST; MASK
20632	1549		UBB	STA	IOR		STA	CF3A			JSL	S024			UNC	SET CARRY IF DIGITS LOST; DO DEC & EXIT

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20635           *
20636           *
20637           *
20638           *          ON ENTRY :   DATA ARE STORED IN TWO SETS OF DATA
20639           *                       BUFFERS XRA1 - XRA9, & XRB0 - XRB8.
20640           *                       SP1B := DIGIT COUNT.
20641           *
20642           *          ON EXIT  :   SHIFTED RESULT ( SHIFTED LEFT 1 DIGIT )
20643           *                       WILL BE STORED IN XRA1 - XRA9 IF # OF
20644           *                       DIGITS >= 16; ELSE WILL BE STORED IN
20645           *                       XRA5 - XRA9 ONLY.
20646           *
20647           *
20648    154A   SLD1       XR9  ADD       SP4A       F800      ADDL      RG              SETUP FOR QASL; MASK
20650    154B              XR7  ADD       SP0A       F000      ADDL                      SETUP FOR QASL; MASK
20652    154C        UBB   REGN AND             ZERO       XR7  ADD       SP3B            SKIP IF DIGIT WILL BE LOST;
20654    154D              XR5  ADD             SF3A         XR5  REPN                03   SETUP REPEAT LOOP
20656    154E              UBA  QASL      XR5                 UBB  LINK          DCTR CTR0  DO LEFT SHIFT 1 DIGIT
20658    154F              UBB  ADD       XR6        000F UBA  ANDL      SP2B            RESULT; SAVE UPPER MOST DIGIT
20660    1550        SP0A       ADD       XR7        SP3B      ADD                       RESULT
20662    1551        UBB        ADD       XR8        FFF0 SP1B ADDL                POS  RESULT; SKIP IF # OF DIGITS >= 16
20664    1552              SP4A ADD       XR9                  XR4  ADD                 RSB  RESULT; RETURN
20666           *
20667           *                    DO SHIFT IF # OF DIGITS >= 16
20668           *
20669    1553              UBB  ADD       SP4A                 XR3  ADD       SP3B            SETUP FOR QASL
20671    1554              XR3  ADD       SP0A       RG        XR0  ADD           CTF1        SET F1 IF THE UPPERMOST BIT IS SET
20673    1555              XR1  ADD             F1HB           XR1  REPN                03   SETUP REPEAT LOOP
20675    1556              UBA  QASL      XR1                 UBB  LINK          DCTR CTR0  DO LEFT SHIFT 1 DIGIT
20677    1557              UBB  ADD       XR2                      ADD                       RESULT
20679    1558        SP0A       ADD       XR3        SP3B      ADD                       RESULT
20681    1559              UBB  ADD       XR4                 SP2B ADD                CRRY RESULT; SLOW RETURN
20683    155A        UBB   SP4A IOR       XR5                      ADD                 RSB  RESULT; RETURN
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20686           *
20687           *
20688           *
20689           *          DECIMAL OVERFLOW FOR NSLD
20690           *
20691           *
20692     155B  EE13 0400 STA  IORL      STA                  JSL  ER13           UNC   SET CARRY; JSB ER13
20694           *
20695           *
20696           *
20697           *          ON ENTRY :  CTR := REGN POINTER AT MSW
20698           *                      DATA ARE STORED IN 2 SETS OF DATA BUFFERS
20699           *                      XRA1 - XRA9, & XRB0 - XRB8.
20700           *
20701           *          ON EXIT  :  SHIFTED DATA ( LEFT SHIFT 2 DIGITS )
20702           *                      WILL BE STORED IN BOTH XRA1 - XRA9, &
20703           *                      XRB0 - XRB9.
20704           *
20705           *
20706     155C  SLD2      REGN ADD  LLZ  SPOA           BKX3 INC       RG   PSHR        MASK UPPER DIGITS; ACC WD-COUNT - 1,
20708     155D            REGN ADD  RLZ                 REGN ADD  LRZ                  SWAP BYTES
20710     155E       UBA  UBB  IOR       REGN                 INC  LSL       DCTR      STORE INTO REGN; 2, DEC COUNTER
20712     155F       RG        CAD       RG   CTF1  UBB  UBA  XFRS      REGN           DEC WORD-LENGTH; STORE INTO REGN
20714     1560       SPOA      ADD                 ZERO  RREG CTRS JSB  *-3  CTR   F1    SKIP IF NO DIGIT WILL BE LOST;
20716                                                                                    NEW REGN POINTER, LOOPBACK IF F1
20717     1561                 ADD                 SF3A                 JSB  MPYS    UNC   SET F3 IF DIGITS WILL BE LOST; RETURN
```

```
 RECORD   C.S.      ********** ALU A **********  ************** ALU B **************
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20720            *
20721            *
20722            *
20723            *         SHIFT LEFT 3 DIGITS --
20724            *
20725            *         1. FIRST, SHIFT THE DATA RIGHT 1 DIGIT
20726            *         2. THEN, SHIFT THE DATA LEFT 1 WORD
20727            *         3. SET F3 IF ANY DIGIT WILL BE LOST.
20728            *
20729            *
20730     1562   SL3D      XR15 ADD  LSR  RH                      ADD            PSHR       SHIFT COUNT ; ADJUST RAR
20732     1563                  JSB  SRD1      MEDJ  OFFF         ADDL      RG               DO RIGHT SHIFT 1 DIGIT; MASK
20734     1564        RG    REGN AND       XR14 ZERO       BKX3 INC       RG   SF2         MASK DIGIT LOST; INC WD COUNT, SF2
20736     1565              ADD            SF3A       RH   JSBI SL3J RH        UNC   SET F3 IF DIGIT LOST; JSB FOR 1-WD SHIFT
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20739             *
20740             *
20741             *
20742             *         SHIFT RIGHT 1 DIGIT ---
20743             *
20744             *         ON ENTRY :
20745             *
20746             *                    DATA ARE STORED IN TWO SETS OF DATA
20747             *                    BUFFERS XRA1 - XRA9, & XRB0 - XRB8.
20748             *                    SP1B := DIGIT COUNT
20749             *
20750             *         ON EXIT  :  SHIFTED RESULT ( SHIFTED RIGHT 1 DIGIT )
20751             *                    WILL BE STORED IN XRA1 - XRA9 IF # OF
20752             *                    DIGITS >= 17; ELSE WILL BE STORED IN
20753             *                    XRA5 - XRA9.
20754             *
20755             *
20756     1566    SRD1       XR9  ADD        SP4A                ADD                        SETUP QASR;
20758     1567               XR7  ADD        SP0A          XR7   ADD        SP3B
20759     1568               XR5  ADD                      XR5   REPN                 03     SETUP REPEAT LOOP
20761     1569               UBA  QASR       XR5           UBB   LINK           DCTR CTR0  DO RIGHT SHIFT 1 DIGIT
20763     156A          UBB  UBA  XFRR       XR6           UBA   ADD            HBF2       RESULT; SET F2 IF NEG
20765     156B          SP0A SREG XFRR       XR7  NF2      XR13  ADD        CTR             RESULT; SKIP IF POS;
20767     156C          0FFF SREG ANDL       XR5       SP3B      ADD                        MASK OFF THE ONES FROM QASR                (0118)
20769     156D          UBB       ADD        XR8       FFEF SP1B ADDL                POS    RESULT; SKIP IF # OF DIGITS >= 17
20771     156E               SP4A ADD        XR9           XR4   ADD                RSB    RESULT; RETURN
20773             *
20774             *         DO SHIFT IF # OF DIGITS >= 17
20775             *
20776     156F               UBB  ADD        SP4A          XR3   ADD        SP3B            SETUP FOR QASR
20778     1570               XR3  ADD        SP0A                ADD
20779     1571               XR1  ADD                      XR1   REPN                 03     SETUP REPEAT LOOP
20781     1572               UBA  QASR       XR1           UBB   LINK           DCTR CTR0  DO RIGHT SHIFT 1 DIGIT
20783     1573               UBB  ADD        XR2           XR13  ADD        CTR             RESULT
20785     1574          SP0A      ADD        XR3       SP3B      ADD                        RESULT
20787     1575               UBB  ADD        XR4                 ADD                NZRO   RESULT; SLOW RETURN
20789     1576               SP4A ADD        XR5                 ADD                RSB    RESULT; RETURN
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20792             *
20793             *
20794             *
20795             *         DECIMAL OVERFLOW FOR ADDD & SUBD
20796             *
20797             *
20798     1577    AE13           ADD                              JSL  ER13           UNC   ; JSB ER13
20800             *
20801             *
20802             *
20803             *         SHIFT RIGHT 2 DIGITS ---
20804             *
20805             *         ON ENTRY :
20806             *
20807             *                    CTR := REGN POINTER AT MSW
20808             *                    DATA ARE STORED IN 2 SETS OF DATA BUFFERS
20809             *                    XRA1 - XRA9, & XRB0 - XRB8.
20810             *
20811             *         ON EXIT  :  SHIFTED DATA ( RIGHT SHIFTED 2 DIGITS )
20812             *                    WILL BE STORED INTO BOTH XRA1 - XRA9, &
20813             *                    XRB0 - XRB9.
20814             *
20815             *
20816     1578    SRD2       REGN ADD       SP0A            BKX3 INC       RH   PSHR       MSW; ACC WD-COUNT - 1
20818     1579                    ADD                       REGN ADD  LRZ  SP1B ICTR       ; MOVE LEFT BYTE TO RIGHT BYTE, ICTR
20820     157A          SP0A      ADD  RLZ  SP0A       RH        CAD       RH   DCTR       MOVE RIGHT BYTE TO LEFT; DEC WD-COUNT & DCTR
20822     157B          REGN ADD       SP0A       UBA  SP1B IOR       REGN ICTR       NEXT DATA WORD; MERGE DATA & STORE INTO REGN
20824     157C          UBB  ADD       REGN       RH        JSB  *-3            POS   STORE INTO REGN ( A ); LOOPBACK IF NOT DONE
20826     157D               ADD                              JSB  MPYS           UNC   RETURN SUBROUTINE
```

```
RECORD  C.S.      ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20829           *
20830           *
20831           *
20832           *         SHIFT RIGHT 3 DIGITS ---
20833           *
20834           *         1. FIRST, SHIFT THE DATA LEFT 1 DIGIT
20835           *         2. THEN, SHIFT THE DATA RIGHT 1 WORD.
20836           *
20837           *
20838    157E   SR3D      XR15 ADD  LSR                      ADD          PSHR       # OF SHIFTS ( WORD ):
20840    157F                   JSB  SLD1      MEDJ      UBA  INC      RH         DO LEFT SHIFT 1 DIGIT
20842    1580                   JSB  SRD4      UNC   0089     ADDL     CTR        DO RIGHT SHIFT 1 WORD; MSW REGN POINTER
20844    1581   MPYS 0005       ADDL      RG                  ADD          POPR NZRO  MASK FOR SRD5; POPR
20846    1582                   ADD                           ADD               RSB   RETURN
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20849            *
20850            *
20851            *
20852            *          DECIMAL ADD, COMPARE, & SUBTRACT ---
20853            *
20854            *          ON ENTRY :
20855            *
20856            *                     SR := 3
20857            *                     RA := OPERAND-1 DIGIT COUNT
20858            *                     RB := OPERAND-1 BYTE ADDRESS ( DB REL )
20859            *                     RC := OPERAND-2 DIGIT COUNT
20860            *                     RD := OPERAND-2 BYTE ADDRESS ( DB REL )
20861            *                           ( VALID AFTER PULM )
20862            *
20863            *          OPERAND-2 ADDR := OPERAND-2 + OPERAND-1.  THE LEFT
20864            *          TRUNCATED RESULT IS STORED AND DECIMAL OVERFLOW OCCURS
20865            *          IF ALL SIGNIFICANT DIGITS OF THE RESULT DO NOT FIT IN
20866            *          THE OPERAND-2 FIELD.
20867            *
20868            *
20869     1583   ADDD            JSZ  PULM      UNC        UBA  JSZ  TRP  XR5       NZRO  PULM FOR RD; TRAP IF CIR ( 9:1 ) NOT ZERO
20871     1584                    ADD       XR14                 JSL  CKLN XR7       UNC   LOST DIGIT BUFFER; CHECK VALID OPERAND LEN
20873     1585                    ADD                            JSL  CKBA XR8       UNC   ; CHECK OPERAND-1/2 ADDR
20875     1586            RH      ADD       XR11          RG     JSL  SPLT XR11      F2    OPERAND-1 WD ADDR;
20877                                                                                        CHK SPLIT STACK IF PRIV, OPERAND-2 WD ADDR
20878     1587            RA      ADD       RG                   JSL  MASK XR9       UNC   ; GET SOURCE/TARGET MASKS ( MSW & SIGN )
20880     1588            XR13    ADD       SP4A                 JSL  LEN  XR10      UNC   OP-2 SIGN MASK; DETERMINE ACC LENGTH
20882     1589            XR12    ADD       RE                   JSL  FCHA           UNC   OP-1 SIGN MASK; GET OPERAND-2 DATA
20884     158A            XR11    ADD       SP0A                 JSL  FCHB XR30      UNC   ; GET OPERAND-2 DATA, NEG/POS INDICATOR
20886     158B      RE    XR13    JSBS *+3        ZERO           ADD       SP3B CCA          JSB IF OP-1 & OP-2 SIGNS ALIGHN;
20888                                                                                       SET CCA
20889     158C            XR12    JSB  SLD2       ODD            ADD       CTR               LEFT 2-DIGIT SHIFT IF OP-1 := 000F
20891                                                                                       MSW REGN POINTER
20892     158D            XR12    JSB  SRD2       EVEN  FFFF XR13 ADDL      CTR               ELSE RIGHT 2-DIGIT SHIFT; MSW REGN POINTER
20894     158E            CIR     ADD  LSR        EVEN  SP3B XR31 ADD             HBF2        SKIP IF ADDD; SET F2 IF EITHER ONE IS NEG
20896     158F            UBB     CAD              HBF2  0098     ADDL      CTR               TOGGLE SIGN; REGN POINTER
20898     1590            CIR     ADD       SP4A                 BKX3 ADD       RH          NF2   SP4A := CIR; WD-COUNT - 1, SKIP IF NF2
20900     1591                    JSB  SUBM       F2     SP3B           ADD       RG    ICTR        DO SUBD IF 'A' & 'B' HAVE DIFF SIGNS
20902                                                                                       OPERAND-2 SIGN INDICATOR
```

```
RECORD  C S                  ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20904           *
20905           *
20906           *
20907           *          DECIMAL ADD ---
20908           *
20909           *          DO DECIMAL ADD IF BOTH OPERAND-1 AND OPERAND-2
20910           *          HAVE THE SAME SIGN.
20911           *
20912           *
20913   1592    ADDM       REGN ADD       SPOA       FFF0 CTRS ADDL      CTR             OP-2 DATA; REGN POINTER AT OP-1 DATA
20915   1593         0002       ADDL      RE              REGN ADD                       2; OP-2 DATA
20917   1594         RH    UBA  SUB       RH    HBF2      UBB  ADD                       SET F2 IF WD COUNT < 2
20919   1595         SPOA  REGN DADD      REGN  CTF1 UBB  REGN LINK          ICTR        DO DECIMAL ADD; INC REGN POINTER
20921   1596         UBB        ADD       REGN       000E CTRS ADDL      CTR             RESULT: OP-2 POINTER
20923   1597                    JSB  *+5          F2    000E      ADDL      RF              JSB IF DONE; RF := 000E
20925           *
20926           *
20927   1598               REGN ADD       SPOA       FFF0 CTRS ADDL      CTR             REPEAT DECIMAL ADD
20929   1599         RH    RE   SUB       RH    HBF2      REGN ADD                       SET F2 IF DONE;
20931   159A         SPOA  REGN DADD      REGN  F1TC UBB  REGN LINK          ICTR        DECIMAL ADD; NEXT REGN POINTER
20933   159B               UBB  ADD       REGN       RF   CTRS JSB  *-3  CTR       NF2  LOOPBACK IF NOT DONE
20935           *
20936           *
20937   159C                    ADD             NF1   FFFF XR13 ADDL      CTR             ; MSW POINTER
20939   159D                    ADD             SF3A       SP4A JSB  CMPD          EVEN  SKIP IF F3 ( DIGIT LOST ); JSB IF CMPD
20941   159E               REGN ADD       XR14            XR13 ADD       CTR             LOST DIGIT BUFFER; MSW POINTER
20943           *
20944           *
20945   159F    ADDE            ADD                            XR11 JSL  STOA SP2B       UNC   ; DO STORAGE, OPERAND-2 WD COUNT ADDR
20947   15A0                    JSB  AE13      F3A                  ADD             CRF         JSB IF DECIMAL OVERFLOW; CRF
20949   15A1                    ADD                                 JSL  S024         UNC   DO SDEC
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20952           *
20953           *
20954           *
20955           *         DECIMAL SUBTRACT ---
20956           *
20957           *         DO DECIMAL SUBTRACT IF OPERAND-1 AND OPERAND-2
20958           *         HAVE DIFFERENT SIGNS.
20959           *
20960           *
20961   15A2    SUBM      REGN ADD       SPOA CF2   FFF0 CTRS ADDL      CTR             OP-2 DATA; OP-1 POINTER
20963   15A3         RH        CAD       RH   HBF2  000F      ADDL      SP3B            DEC WD COUNT;
20965   15A4         SPOA REGN DSUB      REGN CTF1  UBB  CTRS JSB  *+5  CTR       F2    DO DECIMAL SUBTRACT
20967                                                                                    JSB IF DONE
20968           *
20969           *
20970   15A5                   ADD                                 ADD
20971   15A6                   ADD       SPOA       FFF0 CTRS ADDL      CTR             OP-1 DATA; OP-2 POINTER
20973   15A7         RH        CAD       RH   HBF2                ADD                 DEC WD COUNT
20975   15A8         SPOA REGN DSUB      REGN F1TC  SP3B CTRS JSB  *-2  CTR       NF2   DO DECIMAL SUBTRACT; LOOPBACK IF NF2
20977           *
20978           *
20979   15A9                   CIR  JSB  CMPD       EVEN       BKX3 ADD       RH   SF5B      JSB IF CMPD; WD COUNT - 1,SF5B
20981   15AA                   ADD                             XR13 JSB  ADDE CTR       F1    JSB IF NO NEGATE NEEDED
20983   15AB                   RG   CAD                 CCA   UBB  BKX3 ADD       CTR       TOGGLE CCA; LSW REGN POINTER
20985   15AC         RH        CAD       RH                       ADD                 DEC WD COUNT
20987   15AD         REGN DSUB           REGN CTF1                  ADD             DCTR  DO NEGATE
20989           *
20990           *
20991   15AE                   CAD                 RH         CAD       RH   HBF2      -1; DEC WD COUNT
20993   15AF         REGN DSUB           REGN F1TC  UBA  CTRS JSB  *-1  CTR       NF2   DO NEGATE; DO NEXT SET
20995   15B0                   ADD                             XR13 JSB  ADDE CTR       UNC   DO STORAGE
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

20998           *
20999           *
21000           *
21001           *         OPERAND-2 IS COMPARE WITH OPERAND-1 AND SET
21002           *         CCG, CCL, OR CCE IF OP-2 > OP-1, OP-2 < OP-1 OR
21003           *         OP-2 = OP-1.
21004           *
21005           *
21006           *
21007   15B1    CMPD            ADD       SPOA            BKX3 INC        SP3B      F5B   ; ACC WD-LEN - 1, SKIP IF JUST ' SUB '
21008   15B2                    CAD       RH              BKX3 JSL  SO24              F1    JSB JUST 'ADD' & OP-1 <> OP-2 <> 0
21011   15B3                    JSB  *+2        F1    UBB  XR13 JSB  *+2  CTR       NF5B  JSB IF OP-1 = OP-2
21013   15B4              RG    CAD             CCA             JSL  SO24              UNC   TOGGLE CC; DO SDEC
21015   15B5   CMPE SPOA REGN  IOR       SPOA        SP3B        CAD       SP3B CTF1       CALCULATE IF OP-2/OP-1 := 0
21017   15B6        UBA         ADD                 NZRO RH   CTRS JSB  *-1  CTR       F1    SKIP IF NZRO; LOOPBACK
21019   15B7                    ADD             CCA             JSL  SO24              UNC   SET CCE IF OPERAND-2 := OPERAND-1; DO SDEC
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21022            *
21023            *
21024            *
21025            *          BINARY TO DECIMAL CONVERSION ---
21026            *
21027            *          ON ENTRY :
21028            *
21029            *                     SR  := 3
21030            *                     RA  := SOURCE WORD COUNT
21031            *                     RB  := SOURCE WORD ADDRESS ( DB REL )
21032            *                     RC  := TARGET DIGIT COUNT
21033            *                     RD  := TARGET BYTE ADDRESS ( DB REL )
21034            *                            ( VALID AFTER PULM )
21035            *
21036            *               TARGET ADDR := TWO'S COMPLEMENT SOURCE CONVERTED
21037            *               TO PACKED DECIMAL DIGITS.  0 <= WORD COUNT <= 6,
21038            *               0 <= DIGIT COUNT <= 28.  THE TARGET FIELD IS FILLED
21039            *               WITH LEADING ZEROS IF NECESSARY.  IF THE SOURCE FIELD
21040            *               OVERFLOWS THE TARGET FILED, THE LEFT TRUNCATED RESULT
21041            *               IS STORED AND A DECIMAL OVERFLOW OCCURS.  IF EITHER
21042            *               COUNT IS ZERO, ONLY TOS ADJUSTMENT IS PERFORMED.
21043            *
21044            *
21045   15B8  CVBD          JSZ  PULM      UNC       UBA  JSZ  TRP  XR4       NZRO  PULM FOR RD; TRAP IF CIR ( 9:2 ) NOT ZERO
21047   15B9       FFE3 RC  ADDL      RH        FFF9 RA   ADDL                 NCRY  CHECK TARGET DIGIT COUNT & SOURCE WORD COUNT
21049            *
21050   15BA  BD33          ADD       XR3  CF1            JSL  BDER            UNC   CF1; BDER IF INVALID SOURCE WORD COUNT
21052   15BB                ADD       XR5  SF1       RH   JSL  BDER            POS   SF1; BDER IF INVALID TARGET DIGIT COUNT
21054   15BC                ADD       XR6  CF1       RA   JSL  SD24            ZERO  CF1; DO TOS ADJUSTMENT IF RA := 0
21056   15BD                ADD       XR9  SF3A      RC   JSL  SD24            ZERO  SF3A; DO TOS ADJUSTMENT IF RC := 0
21058   15BE                ADD                           JSL  CKBA XR5        UNC   CHECK TARGET ADDRESS
21060   15BF       05F5     ADDL      XR15             RG   JSL  SPLT XR11      F2    MASK; CHECK SPLIT STACK IF PRIV MODE
21062   15C0       RA       ADD       RH               BKX6 JSL  MASK BKX3      UNC   RH := RA; GET TARGET OPERAND MASKS
21064   15C1       RA       CAD       SP4A CF3A  0089       ADDL      CTR           WORD COUNT - 1, CF3A; LSW REGN POINTER
21066   15C2       RB   DB  ADD       SPOA ROD   0089 BKX3 SUBL      XR31          READ SOURCE MSW; MSW REGN POINTER
21068   15C3       UBA  SP4A ADD      SPOA ROD   UBA  DL   BNDE           CF4B      READ SOURCE LSW; CHECK BOUND, MSW ADDR >= DL
21070   15C4            OPA  ADD      SP4A HBF2  RE   UBA  BNDE                     SF2 IF NEG, < MSW >; CHECK IF SM >= LSW ADDR
21072   15C5       RH       CAD       RH         RH        JSB  *+3          ZERO  DECREMENT WORD COUNT; JSB IF WD COUNT := 0
21074   15C6            OPA  ADD       REGN                 ADD       DCTR        STORE INTO REGN; DECREMENT REGN POINTER
21076   15C7       SPOA     CAD       SPOA ROD         UBA  JSB  *-2  REGN  UNC   NEXT READ; LOOPBACK
21078            *
21079   15C8            XR13 ADD                           RA   JSL  LEN  RG    UNC   TARGET SIGN MASK;RG:=SOURCE WORD COUNT,
21081                                                                                  DETERMINE TARGET LENGTH                (2635)
21082   15C9       RA   SP4A IOR            CCA           RA   SUB       SP3B ICTR    SET CCA; -( WORD COUNT ), ICTR
21084   15CA       FFFC RA   ADDL      RH         E100      ADDL      XR14          WORD COUNT - 4; MASK
21086   15CB            REGN JSB  *+2        NZRO           ADD            ICTR    JSB IF NOT ZERO; NEXT REGN POINTER
21088   15CC                 JSB  *-1        UNC   SP3B     JSBI CVZO SP3B  ZERO  CHECK NEXT SOURCE DATA;
21090                                                                                CHECK IF DATA ALL ZERO, INC WORD COUNT
21091   15CD       RA        JSBC BD1        ZERO           CAD            CF5B    DO CONVERT 1 BINARY WORD; -1, CF5B
21093   15CE       RH        JSB  BD3        NEG   UBA  UBB  JSB  BD2  SP2B  ZERO  CONVERT 3 OR 2 BINARY WORDS
21095   15CF            XR5  ADD       SP4A           XR6  JSL  NEGB RG    F2    NEGATE THE BINARY WORDS, IF NECESSARY
21097   15D0       RH        CAD                      RH   JSL  BD4  XR31       ZERO  ;CONVERT 4 BINARY WORDS
21099   15D1       UBA       CSR             HBF2     UBA  JSL  BD5            ZERO  ;CONVERT 5 BINARY WORDS
21101   15D2            XR3  ADD                           JSL  BD6            F2    ;CONVERT 6 BINARY WORDS
21103   15D3            XR5  ADD             SF3A      UBA  JSL  BD7  SP2B   UNC   ;CONVERT 7 BINARY WORDS
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21106           *
21107           *
21108           *
21109           *          CVBD ( CONVERT 1 BINARY WORD ) ---
21110           *
21111           *          ON ENTRY :
21112           *                     XRA9 := BINARY WORD TO BE CONVERTED
21113           *
21114           *          ON EXIT  :
21115           *                     XRA8  := UPPER DECIMAL WORD
21116           *                     XRA9  := LOWER DECIMAL WROD
21117           *
21118           *          ONE BINARY WORD CAN BE CONVERTED TO A MAXIMUM
21119           *          OF 5 DECIMAL DIGITS.
21120           *
21121           *
21122     15D4  BD1        XR9  ADSB      SPOA CF3A  000E      ADDL      CTR             NEGATE DATA IF F2 SET; REPEAT COUNT FOR BTOD
21124     15D5             XR13 CSR            HBF2       BKX6 JSL  BTOD BKX3      UNC   SF2 IF TARG SIGN := 000F BYTE; DO CONVERT
21126     15D6             UBA  ADD            F2         UBB  ADD                       SKIP IF F2 SET; DECIMAL RESULT
21128     15D7             UBA  JSB  *+3       UNC        UBB  REPN                 03    JSB IF SIGN ON LEFT BYTE;
21130                                                                                     SHIFT TO PROPER FORMAT
21131     15D8             UBA  ADD  LSL  XR8             UBB  LINK          DCTR CTR0
21132     15D9             UBB  ADD       XR9             XR31 JSB  *+3  CTR      UNC   JSB DATA IF SIGN IS ON RIGHT BYTE
21134     15DA             UBA  CSR       XR9             UBB  LINK          DCTR CTR0
21135     15DB             UBB  ADD       XR8        0089 BKX3 SUBL      CTR             ; MSW REGN POINTER FOR STORING
21137           *
21138           *
21139           *          STORE DATA INTO MEMORY, AND IF DECIMAL OVERFLOWS
21140           *          TRAP OUT, OR ELSE DO STACK ADJUSTMENT AND NEXT.
21141           *
21142           *
21143     15DC  BDEN           ADD       XR14 CF3A       XR11 JSL  STOA SP2B      NCRY  XRA14 := 0, CLEAR OVERFLOW FLAG; DO STORE
21145     15DD                 JSB  BD33      F3A             ADD            CRF         JSB BD33 IF DECIMAL OVERFLOW; CRF
21147     15DE                 ADD                            JSL  SD24        UNC   DO STACK ADJUSTMENT, THEN NEXT
21149           *
21150           *
21151           *          SHIFT CONVERTED DATA INTO PROPER FORMAT
21152           *
21153           *
21154     15DF  BD3S 008A      ADDL      RH              RC   JSL  SLD1 SP1B      F2    REGN POINTER; SHIFT LEFT 1 DIGIT IF
21156                                                                                     SIGN IS IN RIGHT BYTE
21157     15E0             XR9  ADD                  0089 BKX3 SUBL      CTR             MSW REGN POINTER
21159     15E1             UBA  ADD  RLZ                  UBA  JSB  BDEN XR8       F2    DO STORE IF SIGN IN RIGHT BYTE
21161     15E2       UBA   UBA  ADD  LSL  RG                   JSL  SRD1        UNC   SHIFT RIGHT IF SIGN IN LEFT BYTE
21163     15E3       RG    RG   ADD  LSL  XR10       RH   BKX3 JSBS BDEN CTR    UNC   DO STORE;
21165           *
21166           *
21167           *          CVBD ( CONVERT ZEROS TO DECIMAL )
21168           *
21169           *
21170     15E4  CVZO           JSB  BDEN      CCA        008A BKX3 SUBL      CTR             DO STORE SET CCE; REGN POINTER
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21173           *
21174           *
21175           *
21176           *         CVBD ( CONVERT 2 BINARY WORDS ) ---
21177           *
21178           *         ON ENTRY :
21179           *
21180           *            BINARY WORDS ARE STORED IN TWO SETS, XRA8 - XRA9,
21181           *            AND XRB7 - XRB8.
21182           *
21183           *         ON EXIT  :
21184           *
21185           *            DECIMAL DATA ARE STORED INTO XRA7 - XRA9.
21186           *
21187           *         1. FIRST, BINARY WORDS ARE DIVIDED BY 100M TO GET
21188           *            THE UPPERMOST TWO DIGITS ( QUOTIENT )
21189           *         2. THEN, CONVERT THE REMAINDER TO DECIMAL DIGITS
21190           *         3. CONVERT THE QUOTIENT TO BE THE UPPERMOST 2 DIGITS
21191           *         4. TWO BINARY WORDS CAN BE CONVERTED TO A MAXIMUM OF
21192           *            10 DIGITS.
21193           *
21194           *
21195     15E5  BD2       XR8  ADSB      SP0A                XR8  LINK      SP3B          NEGATE IF F2 SET;
21197     15E6            XR13 CSR             HBF2           JSL  D100                UNC  SF2 IF SIGN IN RIGHT BYTE; DIVIDE BY 100M
21199     15E7            UBA  ADD       SP0A          001E       ADDL      CTR           BINARY WORDS := ( SP0A, SP3B ); REPEAT COUNT
21201     15E8            SP4A ADD       RG                  SP2B JSL  BTOD SP3B      UNC  CONVERT THE BINARY WORDS TO DECIMAL
21203     15E9            UBB  ADD       XR9                 UBA  ADD       XR7           DECIMAL DATA := ( XRA8, XRA9 )
21205     15EA            UBB  ADD       XR8           000E       ADDL      CTR           ; REPEAT COUNT
21207     15EB            RG   ADD       SP0A                     JSL  BTOD SP3B      UNC  DO CONVERSION FOR THE UPPERMOST 2 DIGITS
21209     15EC            UBB  ADD       XR7                      JSB  BD3S          UNC  STORE DATA
```

```
RECORD   C S.          ********** ALU A **********  ************** ALU B **************
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21212            *
21213            *
21214            *
21215            *          CVBD ( CONVERT 3 BINARY WORDS ) ---
21216            *
21217            *          ON ENTRY :
21218            *
21219            *             BINARY WORDS ARE STORED IN TWO SETS, XRA7 - XRA9,
21220            *             & XRB6 - XRB8.
21221            *
21222            *          ON EXIT  :
21223            *
21224            *             DECIMAL DATA ARE STORED INTO XRA6 - XRA9,&
21225            *             XRB5 - XRB8.
21226            *
21227            *          1. THREE BINARY WORDS CAN BE CONVERTED TO A MAXIMUM
21228            *             OF 15 DECIMAL DIGITS.
21229            *          2. FIRST, DIVIDE THE BINARY WORDS BY 100M TO GET THE
21230            *             UPPER 2 DECIMAL WORDS ( IN BINARY )
21231            *          3. CONVERT THE REMAINDER TO DECIMAL DIGITS ( LOWER WORDS )
21232            *          4. THEN, CONVERT THE QUOTIENT TO BE THE UPPER DECIMAL
21233            *             DIGITS.
21234            *
21235            *
21236     15ED   BD3        XR7  ADD       RG                    JSL  NEGB            F2    BINARY WORD; DO NEGATE IF F2 IS SET
21238     15EE              XR8  ADD       SP0A            XR8   ADD       SP3B             SETUP FOR DIVIDE BY 100M
21240     15EF                   ADD       XR5             RG    JSL  D100 SP2B       NCRY  DIVIDE BY 100M
21242     15F0              UBA  ADD       SP0A       001E       ADDL      CTR              SETUP FOR CONVERSION
21244     15F1              SP4A ADD       RG         SP3B       ADD       SP2B             QUOTIENT := ( RG, SP2B )
21246     15F2              XR13 CSR            HBF2             SP2B JSL  BTOD SP3B   UNC   SF2 IF SIGN := 0DOF;
21248                                                                                        CONVERT THE REMAINDER ( SP0A, SP3B )
21249     15F3              UBB  ADD       XR9                   UBA  ADD       XR7        LOWER DECIMAL WORDS
21251     15F4              UBB  ADD       XR8        001E       ADDL      CTR
21252     15F5              SP2B ADD       SP0A            RG    JSL  BTOD SP3B   UNC   CONVERT THE QUOTIENT TO UPPER DECIMAL WORDS
21254     15F6              UBB  ADD       XR7             UBA   ADD       XR5              DECIMAL DATA
21256     15F7              UBB  ADD       XR6             UBA   JSB  BD3S XR6      UNC   DECIMAL DATA; DO STORE DATA INTO MEMORY
21258     15F8                   ADD                             ADD                       NOP to keep system happy
21260     15F9                   ADD                             ADD                       NOP to keep system happy
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21265           %1600
21266           *
21267           *
21268           *
21269           *          DECIMAL TO BINARY CONVERSION ---
21270           *
21271           *          ON ENTRY :
21272           *
21273           *                     RA  := SOURCE DIGIT COUNT
21274           *                     RB  := SOURCE BYTE ADDRESS ( DB REL )
21275           *                     RC  := TARGET WORD ADDRESS ( DB REL )
21276           *             TARGET ADDR := SOURCE DATA CONVERTED TO TWO'S
21277           *             COMPLEMENT BINARY.  0 <= DIGIT COUNT <= 28;
21278           *             THE LENGTH OF THE RESULT DEPENDS ON THE NUMBER
21279           *             OF SOURCE DIGITS AS FOLLOWS :=
21280           *
21281           *                SOURCE DIGIT COUNT       TARGET WRODS
21282           *
21283           *                   1 THROUGH 4               1
21284           *                   5 THROUGH 9               2
21285           *                  10 THROUGH 18              4
21286           *                  19 THROUGH 28              6
21287           *                  29 THROUGH 32              8
21288           *
21289           *             IF THE DIGIT COUNT IS ZERO, ONLY SDEC PORTION OF THE
21290           *             INSTRUCTION IS PERFORMED.  THE 2 SOURCE PARAMETERS
21291           *             ARE DELETED IF SDEC := 0; ELSE DELETED ALL 3 ON TOS.
21292           *
21293           *
21294    1600   CVDB          ADD   XR8             UBA  JSZ  TRP  XR3       NZRO  ; TRAP IF CIR ( 9:2 ) NOT ZERO
21296    1601          SM     ADD   RE        OFFF       ADDL      BKX5            UBA := SM; MASK
21298    1602     RB         ADD   LSR                  JSL  CKB  XR5       UNC   SOUCE WORD ADDR; CK SOURCE ADDR
21300    1603          RH     ADD   XR11                 JSL  MSKB XR6       UNC   SOURCE WD-ADDR; GET SOURCE MASKS
21302    1604     RA   SP1B ADD             CTF1       RA   JSB  DBZR      ZERO  SET F1 IF SOURCE DIGIT COUNT > 28;
21304                                                                             JSB IF SOURCE DIGIT COUNT := 0
21305    1605     RC   DB   ADD   RG                       JSL  DBER XR7       F1    TARGET ADDR; JSB IF SOURCE COUNT > 28
21307    1606     FFF6       ANDL  XR7         0004 RA    SUBL                 NEG   -10; SKIP IF RA < 5
21309    1607     RREG RA    ADD             HBF2       RG   JSB  CKA1 SP1B  UNC   SET F2 IF 5 <= RA <= 9;
21311                                                                             JSB FOR RA < 5, TARGET ADDR
21312    1608          0012  ANDL  XR3                  RG   JSBI CKA1 SP1B  F2    18; JSB IF RA < 10, TARGET LSW ADDR
21314    1609     RREG RA    JSBS CKA1       POS   0002 UBB  ADDL      SP1B        JSB IF RA < 19; TARGET LSW ADDR
21316    160A          0002 UBB ADDL                 001C RA   SUBL                 TARGET LSW ADDR; SKIP IF RA > 28
21318    160B               ADD   XR6                  UBA  JSB  CKA1 SP1B  UNC   ;JSB IF RA < 29
21320    160C               ADD   XR2         0002 UBB  ADDL      SP1B        TARGET LSW ADDR
21322           *
21323           *          BOUNDS CHECKING
21324           *
21325    160D   CKA1       STA   ADD         RE         002F      ADDL      XR16        STATUS; LOOPCOUNT
21327    160E          RE   SP1B BNDE                  RG   DL   BNDE                 BOUNDS CHECKING; SM >= LSW ADDR,
21329                                                                             DL <= MSW ADDR
21330           *
21331           *
21332    160F               ADD                                RE   JSL  SPLT      NEG   CK SPLIT STACK IF PRIV MODE
21334    1610          XR11 ADD         SP0A                  BKX4 JSL  FCHB BKX3 UNC   SOURCE WD ADDR; FETCH SOURCE DATA, WD COUNT
21336    1611     RC   DB   ADD              WRD             UBA  ADD       XR14  NF2   MSW ADDR WRITE; STATUS, SKIP IF RA > 9
21338    1612          XR12 CSR              HBF2                 JSB  *+4            UNC   SF2 IF SOURCE SIGN := 000F; JSB IF RA <= 9
```

RECORD NO	C.S. ADDR	LABL	ALU A RREG	ALU A SREG	ALU A FUNC	ALU A SFNC	ALU A STOR	ALU A SPSK	ALU B RREG	ALU B SREG	ALU B FUNC	ALU B SFNC	ALU B STOR	SPEC	SKIP	COMMENT
21340	1613		FFE3	RA	ADDL		RE			UBA	ADD			LBF5		UBA := RA - 29; SF5B IF SOURCE SIGN := 000F
21342	1614			STA	ADD	LSR	XR14			RA	JSL	SRD1	SP1B		F5B	SHIFT RIGHT STATUS;
21344																SHIFT RIGHT 1 DIGIT IF F5
21345	1615		FFED	RA	ADDL		RH				JSL	SLD1	XR19		NF5B	UBA := RA - 19; SHIFT LEFT 1 DIGIT IF NF5B
21347		*														
21348		*														
21349	1616			XR9	ADD			CF3A	0004	RA	SUBL				NEG	SOURCE DATA; SKIP IF RA > 4
21351	1617				ADD		SPOA		UBA	XR7	JSB	DB1	RG		UNC	SPOA := 0; JSB FOR 1 TARGET WD CONVERSION
21353	1618		2710		ADDL		RG		0009	RA	SUBL				NEG	10K; SKIP IF RA > 9
21355	1619			UBB	SUB			CTF1			JSB	DB2	XR19		UNC	SF1 IF RA <= 9; XR19 := 0,
21357																JSB FOR 2 TARGET WD CONVERSION
21358	161A		RH		JSB	DB4		NEG	0003	DB	ADDL		XR31			JSB FOR 4 WD CONVERSION; UBB := DB + 3
21360		*														
21361		*														
21362	161B				JSB	DB8		UNC		RE	JSB	DB6			NEG	FOR 8 OR 6 WD CONVERSION

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21365            *
21366            *
21367            *
21368            *          CVDB ( CONVERT 1 - 4 DIGITS ) ---
21369            *
21370            *          ON ENTRY :
21371            *
21372            *                     RG := SOURCE DATA TO BE CONVERTED
21373            *                     F2 := 1, IF SOURCE SIGN MASK IS 000F
21374            *
21375            *          ON EXIT  :
21376            *
21377            *                     DATA WILL BE STORED INTO THE TARGET WD
21378            *                     ADDRESS, NEGATED IF NECESSARY.
21379            *
21380            *
21381            *
21382   161C     DB1       RG   CSR                       RG   CSL                        CIRCULAR
21384   161D               UBA  CSR                       UBB  CSL                             SHIFT
21386   161E               UBA  CSR                       UBB  CSL                                 TO RIGHT
21388   161F               UBA  CSR       SP4A F2         UBB  CSL                                         JUSTIFY
21390   1620               UBB  ADD       SP4A       RC   DB   ADD            WRD          SP4A := PROPER FOMRAT; TARGET WD ADDR
21392   1621          RC   DB   ADD            WRD          ADD                          TARGET WD ADDR
21394   1622               SP4A ADD            CF2          JSB  DTOB XR19       UNC     DO CONVERSION
21396   1623          UBA  UBB  ADD            DATA    XR14 ADD  SWAB             EVEN    CORRECT DECIMAL DATA;
21398                                                                                     SKIP IF NOT CCL
21399   1624                    JSB  SD23       UNC         UBA  SUB            DATA     DO SDEC; STORE DATA
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21402           *
21403           *
21404           *
21405           *          CVDB ( CONVERT 5 - 9 DIGITS ) ---
21406           *
21407           *          ON ENTRY :
21408           *
21409           *             TWO SETS OF DECIMAL DATA ARE STORED IN XRA7 - XRA9,
21410           *             & XRB6 - XRB8; AS 1ST, 2ND, & 3RD DECIMAL WORDS RESP.
21411           *             1. CONVERT THE UPPERMOST WORD, IF ANY, TO BINARY.
21412           *             2. MPY THE BINARY BY 10K, AND ADD WITH THE BINARY
21413           *                WORD CONVERTED FROM THE 2ND DECIMAL WORD.
21414           *             3. AGAIN, MPY THE RESULT BY 10K, AND ADD WITH THE
21415           *                BINARY WORD CONVERTED FROM THE 3RD DECIMAL WORD.
21416           *
21417           *
21418    1625   DB2        STA  ADD  LSR  XR14            XR6  JSB  *+6  SP1B      F2    SHIFT RIGHT STATUS; JSB IF SIGN := 000F
21420           *
21421           *          LEFT SHIFT DECIMAL DATA IF SOURCE SIGN MASK := 0F00
21422           *
21423    1626              UBB  ADD                       XR7  REPN           CF4B 03    RIGHT JUSTIFY THE DATA
21425    1627              UBA  ADD  LSL  XR8             UBB  LINK      SP2B DCTR CTR0
21426    1628              SP1B ADD                       XR8  REPN           03
21427    1629              UBA  CSL                       UBB  CSL            DCTR CTR0
21428    162A        000F  UBA  ANDL      SP4A       UBB  SP2B JSB  *+6      UNC
21429           *
21430           *          RIGHT SHIFT DECIMAL DATA IF SOURCE SIGN MASK := 000F
21431           *
21432    162B              XR8  ADD                       XR8  REPN           CF4B 03    RIGHT JUSTIFY THE DATA
21434    162C              UBA  ADD  LSR                  UBB  LINK      SP2B DCTR CTR0
21435    162D              XR7  ADD                       XR7  REPN           03
21436    162E              UBA  ADD  LSR  SP4A            UBB  LINK           DCTR CTR0
21437    162F              UBB  ADD       XR8             SP2B ADD
21438           *
21439           *          CONVERT THE UPPERMOST WORD ( IF ANY )
21440           *
21441    1630        UBB        ADD       XR9                  JSB  *+5  SP2B      NF1   LSW; JSB IF NF1 ( RA <= 8 )
21443    1631              SP4A JSB  *+4       ZERO  FFFF SP4A ADDL      CTR           JSB IF UPPERMOST WORD := 0; DEC COUNTER
21445    1632                   ADD                       REPC      SP3B DCTR CTR0   REPEAT ADD
21447    1633              UBA  ADD                  RG   UBB  LINK           DCTR CTR0
21448    1634              UBB  ADD       SP0A            UBA  ADD       SP2B          BINARY DATA ( LOWER, UPPER )
21450           *
21451           *
21452    1635   DB2B       XR8  ADD       SP4A                 JSB  DTOB           UNC   CONVERT 2ND WORD;
21454    1636        SP0A       ADD                       UBA  UBB  ADD
21455    1637              SP2B ADD       RE              UBA  UBB  LINK
21456    1638        2710       ADDL      SP4A                 UBB  JSB  X1W2 RG      UNC   MPAD DATA BY 10K
21458    1639              XR9  ADD       SP4A                      JSB  DTOB           UNC   CONVERT 3RD WORD;
21460    163A        SP0A       ADD       SP4A            UBA  UBB  ADD       RG
21461    163B              XR14 ADD  SWAB       HBF2                 JSB  DB4B           F4B   SF2 IF CCL; RTN IF FOR 4 WD CONVERSION
21463    163C                   ADD                  RC   DB   INC           WRD           ; MSW ADDR
21465    163D              SP2B ADD                  RG   SP4A LINK
21466    163E              UBA  ADSB                      UBB  LINK      SP1B DATA
21467    163F              UBA  ADD            DATA                 JSB  SD23           UNC   DO SDEC
```

```
 RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21470           *
21471           *
21472           *
21473           *          CVDB ( CONVERT 10 - 18 DIGITS ) ---
21474           *
21475           *          ON ENTRY :
21476           *
21477           *             DATA ARE STORED IN XRA4 - XRA8 IF SIGN := 0F00, OR
21478           *                                IN XRA5 - XRA9 IF SIGN := 000F.
21479           *
21480           *             REFER DB2 FOR CONVERSION ALGORITHM.
21481           *
21482           *
21483    1640   DB4         XR5  ADD       SP4A          BKX3 JSBI *+4  RG        F5B
21484    1641                XR4  ADD       SP4A     0088      ADDL      CTR
21485    1642                     CAD                RG        CAD       RG   ICTR ZERO
21486    1643                REGN ADD       REGN       UBA    CTRS JSB  *-1  CTR       UNC   MOVE DATA TO XRA5 - XRA9
21488           *
21489           *          CONVERT THE UPPERMOST WORD TO BINARY, & X 10K
21490           *
21491    1644                SP4A JSB  DTOB      UNC                 ADD       SP2B
21492    1645                     ADD       SP0A       UBA    UBB    JSB  X1WD RG        NZRO
21493           *
21494           *          CONVERT NEXT WORD TO BINARY, ADD, & X 10**12
21495           *
21496    1646                XR6  ADD       SP4A                XR16 JSB  DTOB CTR       UNC
21497    1647           SP0A      ADD                 UBA    UBB  ADD
21498    1648                SP2B ADD       RE         UBA    UBB  LINK      SP1B
21499    1649           UBA  UBB  JSB  *+3        ZERO          UBB  JSB  *+3  SP3B      ZERO
21500           *
21501    164A           00E8      ADDL      SP0A       D4A5       ADDL      SP3B
21502    164B           1000      ADDL      SP4A                  JSB  DDD  RG        UNC
21503           *
21504    164C                UBB  ADD       XR1                   SP4A ADD       XR4
21505    164D           SP0A      ADD                 SP3B       ADD       SP1B
21507           *
21508           *          CONVERT NEXT WORD TO BINARY, ADD, & X 10**8
21509           *
21510    164E                XR7  ADD       SP4A       UBA            JSB  DTOB BKX7      UNC
21511    164F           05F5      ADDL      RE         UBA    UBB     JSB  X2WD          UNC
21512           *
21513    1650           SP0A      ADD                 RC     XR31 ADD                WRD
21514    1651                     ADD                 UBA    XR4  LINK      XR4
21515    1652           RG        ADD                 UBA    SP2B LINK
21516    1653           UBA       ADD                 UBB    SP1B LINK      XR3
21517    1654                XR1  ADD       XR1        UBA    BKX7 LINK      XR2  SF4B
21518    1655                     ADD       SP0A                   JSB  DB2B SP2B      UNC   JSB TO CONVERT THE LAST 2 DECIMAL WORDS
21520           *
21521           *          ADD ALL BINARY RESULT, & STORE INTO MEMORY
21522           *
21523    1656   DB4B        SP2B ADD                 RG     SP4A LINK
21524    1657                UBA  ADD                 UBB    XR4  LINK      RG   DATA
21525    1658                     ADD                 UBA    XR3  LINK      SP2B
21526    1659                XR1  ADD       SP4A DATA  UBA    XR2  LINK      SP1B
21527    165A                UBB  ADD            DATA  RC     XR31 ADD                WRD
21528    165B                SP2B ADD            DATA                JSB  SD23          NF2   DO SDEC
```

```
 RECORD  C.S.          ********** ALU A **********  ************** ALU B **************
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21530            *
21531            *
21532            *
21533            *
21534            *        NEGATE RESULT IF NEEDED, & STORE INTO MEMORY
21535            *
21536            *
21537    165C            SP2B SUB        RG   CTF1       RG   LINK           DATA
21538    165D       RC   DB   ADD             WRD    RC   DB   INC           WRD
21539    165E            SP4A CAD        SP4A FITC       SP1B LINK          DATA
21540    165F            UBA  JSB  SD23     DATA       RG   ADD           DATA
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21542           *
21543           *
21544           *
21545           *          CVDB ( CONVERT 1 DECIMAL WORD TO BINARY ) ---
21546           *
21547           *               DECIMAL WORD := A*1000 + B*100 + C*10 + D
21548           *
21549           *               BINARY WORD  := A*1000/2**12 + B*100/2**8 +
21550           *                               C*10/2**4 + D.
21551           *
21552           *
21553     1660  DTOB UBA        ADD            NZRO  F000 UBA  ANDL                    SKIP IF DECIMAL IS NZRO; UBB := LEFTMOST DIG
21555     1661       UBB        ADD  LSR       RSB   UBA  XR19 IOR       XR19         RIGHT SHIFT, RTN IF DECIMAL IS ZERO;
21557                                                                                  SAVE ACC DIGIT
21558     1662       UBA        ADD  LSR             UBA       ADD  LSL               RIGHT SHIFT; UBB := A000
21560     1663       UBA  UBB   ADD  LSR             0F00 SP4A ANDL                   UBA := ( A*10 )/16; UBB := 0B00
21562     1664       UBA  UBB   ADD  LSR             UBA  UBB  ADD
21563     1665       UBA        ADD  LSR                  UBB  ADD
21564     1666       UBA  UBB   ADD  LSR             00F0 SP4A ANDL                   UBA := (( A*10 )/16 + B )*10/16;UBB := 00C0
21566     1667       UBA  UBB   ADD  LSR             UBA  UBB  ADD
21567     1668       UBA        ADD  LSR                  UBB  ADD
21568     1669       UBA  UBB   ADD  LSR       RSB   000F SP4A ANDL                   UBA := ((( A*10 )/16 + B )*10/16 + C )*10/16
21570           *
21571           *          MULTIPLY ( RE, RG ) BY ( SP0A, SP3B, SP4A )
21572           *
21573     166A  DDD          ADD                                 REPC                  REPEAT LOOP
21575     166B       RE   UBA  MPAD                  RG    UBB  LINK          DCTR CTRO  DO MPY
21577     166C            SREG ADD       RG                  SREG ADD       SP2B      RSB  PRODUCT ( RG, SP2B, SP0A, SP3B )
21579           *
21580           *
21581           *          MULTIPLY ( 05F5, E100 ) BY ( 0, 0, SP4A )
21582           *
21583     166D  X2WD UBB        ADD       SP4A        E100        ADDL      RG            DECIMAL WORD; RG := E100
21585     166E               ADD       SP0A                    REPN      SP3B     OF   REPEAT LOOP
21587     166F       RE   UBA  MPAD                  RG    UBB  LINK          DCTR CTRO  DO MPY
21589     1670            UBA  ADD       RG                  UBB  ADD       SP2B      RSB  PRODUCT ( RG, SP2B, SP0A, SP3B )
21591           *
21592           *
21593           *          MULTIPLY ( 0, RG ) BY ( 0, 0, 2710 )
21594           *
21595     1671  X1WD 2710    ADDL       SP4A                    ADD       RE            SP4A := 2710; RE := 0
21597     1672  X1W2         ADD        SP0A                    REPN      SP3B     OF   SP0A := 0; REPEAT LOOP
21599     1673       RE   UBA  MPAD                  RG    UBB  LINK          DCTR CTRO  DO MPY
21601     1674            UBA  ADD       RG                  UBB  ADD       SP2B      RSB  PRODUCT ( RG, SP2B, SP0A, SP3B )
```

```
 RECORD   C S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21604           *
21605           *
21606           *
21607           *          CVDB ( CONVERT 19 - 28 DIGITS ) ---
21608           *
21609           *          DATA ARE STORED IN XRA3 - XRA9 IF SIGN := 000F, OR
21610           *                            IN XRA2 - XRA8 IF SIGN := 000F.
21611           *
21612           *          REFER DB2 FOR CONVERISON ALGORITHM.
21613           *
21614           *
21615    1675   DB6        XR3  ADD        SP4A            BKX3 JSBI *+4  RG         F5B
21616    1676                XR2  ADD        SP4A       0088      ADDL      CTR
21617    1677                     CAD                   RG        CAD       RG   ICTR ZERO
21618    1678                REGN ADD        REGN       UBA  CTRS JSB  *-1  CTR       UNC
21619           *
21620           *          CONVERT THE UPPERMOST WORD TO BINARY, & 10K
21621           *
21622    1679                SP4A ADD                                JSL  DTOB SP2B       UNC
21623    167A          UBA   UBB  ADD        RG                 SP4A JSL  X1WD            NZRO
21624           *
21625           *          CONVERT NEXT WORD TO BINARY, ADD, & X 10**8
21626           *
21627    167B                XR4  ADD        SP4A                    JSL  DTOB            UNC
21628    167C          SP0A       ADD             SF1   UBA  UBB  ADD
21629    167D                SP2B ADD        RE         UBA  UBB  LINK      SP1B
21630           *
21631    167E          UBA   UBB  IOR             ZERO  05F5      ADDL      SP3B
21632    167F                     ADD        SP0A CF1   001F      ADDL      CTR
21633    1680          E100       ADDL       SP4A            SP1B JSL  DDD  RG         NF1
21634           *
21635    1681                SREG ADD        XR0                XR14 ADD  SWAB      LBF5
21636    1682                SP2B ADD        XR1                     ADD       XR3       F1
21637    1683          SP0A       ADD                        SP3B      ADD       XR3
21638           *
21639           *          CONVERT NEXT WORD TO BINARY, ADD, & X 10K
21640           *
21641    1684                XR5  ADD        SP4A                UBA  JSL  DTOB BKX7       UNC
21642    1685          UBA   UBB  ADD        RG                      JSL  X1WD            UNC
21643           *
21644           *          CONVERT NEXT WORD TO BINARY, ADD, & X 10**12
21645           *
21646    1686                XR6  ADD        SP4A                    JSL  DTOB            UNC
21647    1687          SP0A       ADD                             UBA  UBB  ADD
21648    1688                SP2B ADD                             UBA  UBB  LINK
21649    1689                UBA  ADD                             UBB  XR3  LINK      RG
21650    168A                XR1  ADD        XR1   CTF1          UBA  BKX7 LINK      RE
21651           *
21652    168B          1000       ADDL       SP4A            D4A5      ADDL      SP3B
21653    168C          00E8       ADDL       SP0A                    XR16 JSL  DDD  CTR       UNC
21654           *
21655    168D                UBB  ADD        XR3                     SP4A ADD       XR6
21656    168E          SP0A       ADD             F1        SP3B      ADD       XR5
21657    168F                XR0  ADD        RE    UNC               UBA  ADD       XR4
21658    1690                XR0  INC        RE              RG        ADD       BKX6
21659    1691                XR1  ADD        RG              D4A5      ADDL      SP3B
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21660     1692       00E8      ADDL      SP0A        RC   XR31 ADD             WRD
21661          *
21662     1693       1000      ADDL      SP4A             XR16 JSL  DDD  CTR       UNC
21663          *
21664     1694            SP4A ADD                      BKX6 ADD        RF
21665     1695            XR3  ADD            NCRY  UBA  XR4  LINK       XR4
21666     1696            RF   INC       RF         UBA       ADD
21667     1697       SP0A      ADD       RF              UBB  ADD
21668     1698            RF   ADD                 SP3B UBB  LINK       XR3
21669     1699            SP2B ADD       RF         UBA  RF   LINK       BKX3
21670          *
21671          *       CONVERT NEXT WORD TO BINARY, & X 10K
21672          *
21673     169A            XR7  ADD       SP4A                  JSL  DTOB            UNC
21674     169B       UBA  UBB  ADD       RG                    JSL  X1WD            UNC
21675          *
21676          *       CONVERT NEXT WORD TO BINARY, ADD, & X 10K
21677          *
21678     169C            XR8  ADD       SP4A                  JSL  DTOB            UNC
21679     169D       SP0A      ADD                       UBA  UBB  ADD
21680     169E            SP2B ADD       RE         UBA  UBB  LINK
21681     169F       2710      ADDL      SP4A             UBB  JSL  X1W2 RG        UNC
21682          *
21683          *       CONVERT LAST WORD TO BINARY, & ADD
21684          *
21685     16A0            XR9  ADD       SP4A             XR3  JSL  DTOB SP1B      UNC
21686          *
21687          *       ADD ALL THE BINARY RESULT, & STORE INTO MEMORY
21688          *
21689     16A1       SP0A      ADD                       UBA  UBB  ADD
21690     16A2            SP2B ADD       RE    NCRY  UBA  UBB  LINK
21691     16A3            RG   INC       RG              UBB  ADD
21692     16A4            RE   ADD            NCRY  UBB  XR6  LINK       RE
21693     16A5            RG   INC       RG              UBA  ADD
21694     16A6                 ADD                  UBB       ADD
21695     16A7            RG   ADD                  UBB  XR5  LINK       RH
21696     16A8            SP1B ADD       XR0   CTF1  UBA  XR4  LINK       SP2B DATA
21697     16A9                 INC                  UBA       ADD        SP1B      NF1
21698     16AA            RF   ADD       RE    DATA  UBA  BKX3 LINK       BKX3
21699     16AB            RE   ADD       SP4A                RH   ADD            DATA
21700     16AC                 ADD                       BKX3 ADD
21702     16AD            UBB  ADD            DATA         SP4A ADD            DATA
21703     16AE            SP1B ADD            DATA              JSB  SD23            NF5B  DO SDEC IF NF5B
21705          *
21706          *       DO NEGATE IF NECESSARY, THEN STORE INTO MEMORY
21707          *
21708     16AF                 JSB  DBNE      UNC   RC   DB   ADD            WRD
21709     16B0            SP2B ADD            DATA         RE   ADD            DATA
21710     16B1            RH   ADD            DATA         BKX3 ADD            DATA
21711     16B2            SP4A JSB  SD23      DATA         SP1B ADD            DATA
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21713             *
21714             *
21715             *
21716             *          CVDB ( CONVERT 29 - 32 DIGITS ) ---
21717             *
21718             *          ON ENTRY :
21719             *
21720             *             DECIMAL DATA ARE STORED IN XRA2 - XRA9, &
21721             *             REFER DB2 FOR THE ALGORITHM.
21722             *          **** THIS IS NOT SUPPORTED ON S64 ****
21723             *
21724     16B3    DB8             DIAG                          ADD
21725     16B4    SRDX            ADD                     XR13  ADD       CTR
21726     16B5                    ADD                     XR29  ADD       SP1B      RSB
21727     16B6                    ADD                           ADD
21728             *
21729             *          NEGATE THE DATA, & STORE OUT TO MEMORY
21730             *
21731     16B7    DBNE    RH   SUB       RH   CTF1      SP4A  LINK      SP1B
21732     16B8            SP1B CAD            F1TC      SP2B  LINK      SP2B
21733     16B9            SP1B ADD       SP4A           UBA   ADD       SP1B
21734     16BA            RE   CAD       RE   F1TC      BKX3  LINK      BKX3
21735     16BB            RG   CAD       RG   F1TC      XR11  LINK      XR11
21736     16BC      0007  DB   ADDL                          ADD             RSB
```

```
RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21738            *
21739            *
21740            *
21741            *          SD23 --- WILL POP THE STACK TWO OR THREE TIMES.
21742            *
21743            *                   IF CIR ( 11:1 ) := 0   , DECREMENT BY 2, OR
21744            *                                   := 1   , DECREMENT BY 3.
21745            *
21746            *
21747     16BD   SD23           ADD                       XR19 ADD                 NZRO  ; SET CCE IF XRB19 := NZRO
21749                                                                                     XRB19 := ACC DIGITS
21750     16BE                  ADD                            ADD           CCA
21751     16BF   DBZR 0010 CIR  ANDL                           ADD           EPP2        CHECK CIR ( 11:1 ); POP2
21753     16C0          UBA     ADD           ZERO             ADD           CRF         SKIP IF CIR ( 11:1 ) := 0; CRF
21755     16C1                  ADD           EPOP             JSZ  NEXT           UNC   POP ONE; DO NEXT
```

```
 RECORD  C S                ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21758            *
21759            *
21760            *
21761            *          DOUBLE LOGICAL MULTIPLY ---
21762            *
21763            *          ON ENTRY :
21764            *
21765            *                       RA := LOW-ORDER END OF OPERAND-1
21766            *                       RB := HIGH-ORDER END OF OPERAND-1
21767            *                       RC := LOW-ORDER END OF OPERAND-2
21768            *                       RD := HIGH-ORDER END OF OPERAND-2
21769            *                             ( VALID AFTER PULM )
21770            *
21771            *             THE TWO DOUBLE-WORD OPERANDS ON THE TOP OF STACK
21772            *             ARE LOGICALLY MULTIPLIED.  THE TWO OPERANDS ARE
21773            *             REPLACED BY THE FOUR-WORD LOGICAL PRODUCT.  CARRY
21774            *             IS CLEARED IF THE HIGH-ORDER 32-BITS ARE ALL ZEROS;
21775            *             OTHERWISE, CARRY IS SET.
21776            *
21777            *
21778  16C2  DMPY            JSZ  PULM      UNC        UBA  JSZ  TRP          NZRO  PULM FOR RD; TRAP IF CIR ( 9:3 ) NOT ZEROS
21780  16C3        RC         ADD       SP4A CF1        UBA  ADD       SP3B        SET UP FOR MPAD
21782  16C4                   ADD       SP0A                 REPN          CCRY 1F     ; CLEAR CARRY, REPEAT LOOP FOR MPAD
21784  16C5        RB   UBA   MPAD                   RA   UBB  LINK          DCTR CTR0  DO MPAD
21786  16C6             UBA   ADD       SP4A CCA        UBB  ADD       RC          SET CC;
21788  16C7        UBA  UBB   IOR             ZERO   SP3B      ADD       RA   CRF   SKIP IF HIGH-ORDER 32 BITS ARE ALL ZEROS;
21790  16C8        0400 STA   IORL        STA                  SP4A ADD       RD        ZERO  SET CARRY; SKIP IF HIGH-ORDER 16 BITS ZEROS
21792  16C9        SP0A       ADD        RB                        JSZ  NEXT           UNC   DO NEXT
21794  16CA                   ADD                   CCA  UBA  RC   IOR                  NZRO  SET CCE; SKIP IF NZRO
21796  16CB                   ADD                        RA   JSZ  NEXT           ZERO  DO NEXT IF ALL ZEROS
21798  16CC                   INC                   CCA       JSZ  NEXT           UNC   SET CCG; THEN DO NEXT
```

```
RECORD   C.S.        ********** ALU A **********  ********** ALU B **********
  NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21801          *
21802          *
21803          *
21804          *         ROUTINE TO NEGATE BINARY WORDS FOR CVBD
21805          *
21806          *
21807    16CD  NEGB      XR8  SUB       XR8  CTF1       XR8  LINK      SP1B
21808    16CE            UBB  ADD       XR9             XR5  ADD
21809    16CF            UBB  CAD       XR6  F1TC       RG   LINK      RG
21810    16D0            UBB  ADD       XR7             XR3  ADD
21811    16D1            UBB  CAD       XR4  F1TC       XR4  LINK
21812    16D2            UBB  ADD       XR5        FFF9 RA   ADDL              POS
21813    16D3                 ADD                      SP1B ADD       XR8      RSB   ;RETURN IF RA <= 6
21815    16D4            XR3  CAD       XR3  F1TC            ADD       XR8      RSB
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21817           *
21818           *
21819           *
21820           *          TRAP ROUITNES FOR CVDB
21821           *
21822           *
21823           *
21824   16D5    DBER 2000 STA  ANDL                          ADD            CRF       USER TRAP ENABLED? CRF
21826   16D6                   ADD                     UBA   JSB  *+2             ZERO  ; JSB IF USER TRAP NOT ENABLED
21828   16D7                   JSZ  TRPR       UNC           JSZ  TRPD            NF1   TRAP TARGETS
21830   16D8         0800 STA  IORL       STA               JSB  SD23            UNC   SET OVERFLOW; DO STACK ADJUSTMENT
21832           *
21833           *
21834           *
21835           *          TRAP ROUTINES FOR CVBD & CVAD
21836           *
21837           *
21838           *
21839   16D9    BDER 2000 STA  ANDL                          ADD            CRF       USER TRAP ENABLED? CRF
21841   16DA                   JSZ  TRPO       F3A     UBA   JSB  *+3             ZERO  DECIMAL OVERFLOW TRAP;
21843                                                                                 JSB IF USER TRAP NOT ENABLED
21844   16DB         0002 DSPL XORL                          JSZ  TRPR            F1    IS IT FROM CVAD; INVALID DECIMAL LEN TRAP
21846   16DC                   JSZ  TRPS       UNC     UBA   JSZ  TRPA            ZERO  TRAP TARGETS
21848   16DD         0800 STA  IORL       STA               JSL  SD24            UNC   SET OVERFLOW, DO STACK ADJUSTMENT
21850           *
21851           *
21852           *
21853           *          ROUTINE FOR MPYD PRODUCT >= 29 DIGITS, TESTING
21854           *          IF OVERFLOW OCCURS
21855           *
21856           *
21857           *
21858   16DE    MY29           ADD                FFF0 XR1  ANDL                 NZRO  ; SKIP IF PRODUCT >= 29 DIGITS
21860   16DF                   ADD                         ADD                  RSB   ; RETURN
21862   16E0           XR10 ADD        STA               XR26 JSL  ER13 RC      UNC   RESTORE STATUS; TRAP FOR DECIMAL OVERFLOW
21864   16E1                   ADD                         ADD                        NOP to keep system happy
21866   16E2                   ADD                         ADD                        NOP to keep system happy
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21871           %1700
21872           *
21873           *
21874           *
21875           *          DECIMAL MULTIPLY ---
21876           *
21877           *          ON ENTRY :
21878           *
21879           *                     RA  := OPERAND-1 DIGIT COUNT
21880           *                     RB  := OPERAND-1 BYTE ADDR ( DB REL )
21881           *                     RC  := OPERAND-2 DIGIT COUNT
21882           *                     RD  := OPERAND-2 BYTE ADDR ( DB REL )
21883           *                        ( VALID AFTER PULM )
21884           *
21885           *          OPERAND-2 ADDR := PRODUCT OF THE OPERAND-1 DATA
21886           *          TIMES THE OPERAND-2 DATA.  THE MAXIMUM LENGTH ALLOWED
21887           *          FOR OPERANDS IS 28 DIGITS, THE MINIMUM IS ZERO.  IF NOT
21888           *          WITHIN THESE RANGES, TRAP WILL OCCUR WITH INVALID DIGIT
21889           *          COUNT.  IF EITHER DIGIT COUNT IS ZERO, ONLY SDEC IS
21890           *          PERFORMED.  IF THE RESULT DOES NOT FIT INTO THE OPERAND-2
21891           *          ADDR, THEN AN OVERFLOW TRAP OCCURS.  THE RESULT IN THIS
21892           *          CASE WILL BE LEFT TRUNCATED UNLESS THE ACTUAL RESULT IS
21893           *          GREATER THAN 28 DIGITS, IN WHICH CASE NOTHING WILL BE
21894           *          STORED, CCA IS SET ON THE RESULT.
21895           *
21896           *
21897    1700   MPYD           JSZ  PULM      UNC        UBA  JSZ  TRP  XR5       NZRO  PULM FOR RD; TRAP IF CIR ( 9:1 ) NOT ZERO
21899    1701                  ADD                                JSL  CKLN XR7      UNC   CK OPERAND-1/2 LENGTHS
21901    1702                  ADD                                JSL  CKBA XR8      UNC   CK OPERAND-1/2 ADDRESSES
21903    1703       RH         ADD       XR11             RG   JSL  SPLT XR11      F2    OPERAND-1 ADDR; CK SPLIT STACK IF PRIV MODE
21905                                                                                      OPERAND-2 ADDR
21906    1704           STA    ADD       XR10             BKX6 JSL  MASK XR22      UNC   SAVE STATUS; GET OPERAND-1/2 MASKS
21908                                                                                      SAVE OPERAND-2 WD COUNT -1
21909    1705                  ADD                        XR11 ADD       XR20            ; SAVE OPERAND-2 WD ADDR
21911    1706           XR13   ADD                        BKX7 ADD       XR23            ; SAVE OPERAND-2 MSW MASK
21913           *
21914           *
21915    1707       0020       ADDL      SPOA             UBA  JSL  SCNB XR21      UNC   32; SCAN FOR OPERAND-1 LEADING ZEROS,
21917                                                                                      SAVE OPERAND-2 SIGN MASK
21918    1708       SPOA   RH   SUB       SPOA             RH   JSL  SCNA RE        UNC   SPOA := 32 - ACTUAL OPERAND-1 DIGIT COUNT
21920                                                                                      SCAN FOR OPERAND-2 LEADING ZEROS, RE := RH
21921           *
21922           *
21923    1709       RH     RE   SUB             HBF2       RE   ADD            CF4B ZERO  SF2 IF RE < RH; SKIP IF RE := 0
21925    170A       0003   RC   ADDL                  RC         ADD                 NZRO  UBA := ACTUAL OP-2 DIGIT COUNT + 3;
21927                                                                                      SKIP IF ACUTAL OP-2 DIGIT COUNT := 0
21928    170B       SPOA   UBA  JSBS ER13      NEG   RE   UBA  ADD  LSR       SF4B        TRAP IF RESULT > 28; SF4 IF EITHER OP := 0
21930    170C              UBB  ADD  LSR  SP4A              UBA  ADD       RF            ACC LENGTH - 1;
21932                                                                                      RF := SAVE FOR TESTING RESULT > 28
21933    170D              XR12 ADD       SPOA  NF2        BKX6 ADD       BKX3          OPERAND-1 SIGN MASK; OPERAND-2 WD COUNT - 1
21935    170E              XR11 JSB  MSWP       UNC        XR11 ADD                     JSB IS DIGIT COUNT OF OP-2 > OP-1
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

21938           *
21939           *
21940   170F        RE       CAD  LSR              FFFF SP4A ADDL      XR15           MULTIPLIER WD LEN; ACC WD LEN -1
21942   1710        UBA      ADD  LSR  XR14        0008 SREG SUBL                POS  MULTIPLIER WD LEN - 1;
21944   1711                 ADD                    FFFE SREG ADDL      XR15           ; UBB := PRODUCT LEN -1 ( 0 TO 7 WDS )
21946           *
21947           *        FETCH MULTIPLIER
21948           *
21949   1712                 XR11 ADD       SPOA           BKX4 JSL  FCHB BKX3      UNC  M'IER ADDR; FETCH DATA, M'IER WD COUNT - 1
21951   1713                 XR12 CSR       RG   HBF2      RE   JSB  *+3  SP1B      F4B  SF2 IF SIGN MASK := 000F; JSB IF EITHER := 0
21953   1714                      ADD       XR15 CF2       XR31 JSL  SRD1 XR30      F2   XRA15 := 0; SHIFT DATA RIGHT IF F2
21955   1715                      ADD                      RG   JSL  SR3D           EVEN ;SHIFT RIGHT 3 DIGITS IF SIGN MASK := 0F00
21957           *
21958           *        SAVE THE MULTIPLIER
21959           *
21960   1716                 XR12 ADD       SPOA      00C3       ADDL      XR14           SIGN MASK; REGN POINTER FOR LSW MULTIPLIER
21962   1717                 XR6  ADD                 00C0       ADDL      CTR            MSW MULTIPLIER; MSW MULTIPLIER POINTER
21964   1718                 XR7  ADD                 UBA  XR11  XFRR      REGN ICTR      NEXT MULTIPLIER; SAVE IN XRB64, ICTR
21966   1719                 XR8  ADD                 UBA  SREG  XFRR      REGN ICTR      NEXT MULTIPLIER; SAVE IN XRB65, ICTR
21968   171A                 XR9  ADD           CF3A  UBA  SREG  XFRR      REGN ICTR      LSW MULTIPLIER; SAVE IN XRB66, ICTR
21970   171B                 XR12 ADD       SPOA CF1   UBA  SREG  XFRR      REGN           OP-1 SIGN MASK; XRB67 := LSW MULTIPLIER
21972   171C                 XR11 JSB  MSWP       UNC        SREG  ADD       SP1B           SWAP FOR MULTIPLICAND; SP1B := OP-2 WD ADDR
21974           *
21975           *        FETCH MULTIPLICAND
21976           *
21977   171D                 SP1B ADD       SPOA           BKX4 JSL  FCHB BKX3      UNC  M'CAND ADDR; FETCH DATA,M'CAND WD COUNT - 1
21979   171E                      ADD                      XR30 JSB  MZRO           F4B  JSB IF EITHER := 0
21981   171F                 XR12 CSR       RG   HBF2 UBB  XR31 INC                CCA  SF2 IF SIGN MASK := 000F; SET CCA
21983   1720                      ADD       XR15 CF2       RE   JSL  SRD1 SP1B      F2   XRA15 := 0; SHIFT DATA RIGHT IF F2
21985   1721                      ADD                      RG   JSL  SR3D           EVEN ;SHIFT RIGHT 3 DIGITS IF SIGN MASK := 0F00
21987           *
21988           *        MOVE DATA FROM XRA1 - XRA9 TO XRA19 - XRA27, &
21989           *        ZERO OUT XRA1 - XRA9 AND XRB1 - XRB9.
21990           *
21991   1722        0013     ADDL      SP4A           XR15 INC       RG             19; PRODUCT LENGTH
21993   1723        FFFE     ADDL      SPOA      0089       ADDL      CTR            -2; REGN POINTER
21995   1724        RG       CAD       RG   HBF2           ADD       REGN           SF2 IF DONE; REGN := 0
21997   1725                 REGN AND  REGN           0012 CTRS ADDL      CTR       F2   REGN := 0; REGN POINTER, SKIP IF DONE
21999   1726                 SREG ADD  REGN           UBB  SP4A JSBS *-2  CTR       UNC  REGN := 'MOVED' DATA; LOOPBACK IF NOT DONE
```

```
RECORD  C.S.        ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL  RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22002           *
22003           *
22004           *
22005           *         SETUP 2X, 4X, 8X OF THE MULTIPLICAND
22006           *
22007           *
22008   1727    MPYR 009B       ADDL                      BKX4 JSB  MPYX RH        UNC   SETUP 2X
22010   1728         00A7       ADDL                      BKX4 JSB  MPYX RH        UNC   SETUP 4X
22012   1729         00B3       ADDL                      BKX4 JSB  MPYX RH        UNC   SETUP 8X
22014           *
22015           *         DO ADD & SHIFT
22016           *
22017   172A                 INC        SP4A              XR14 JSB  MPYM CTR       UNC   1; DO LSD'S OF THE MULTIPLIER
22019   172B                 ADD                          UBA  JSL  SRD1 SP1B      UNC   SHIFT RIGHT
22021   172C                 INC        SP4A              XR14 JSB  MPYM CTR       UNC   1; DO NEXT DIGITS OF THE M'IER
22023   172D                 ADD                          UBA  JSL  SRD1 SP1B      UNC   SHIFT RIGHT
22025   172E                 INC        SP4A              XR14 JSB  MPYM CTR       UNC   1; DO NEXT DIGITS OF THE M'IER
22027   172F                 ADD                          UBA  JSL  SRD1 SP1B      UNC   SHIFT RIGHT
22029   1730                 INC        SP4A              XR14 JSB  MPYM CTR       UNC   1; DO MSD'S OF THE M'IER
22031           *
22032           *         ADJUST THE DATA FOR PROPER OUTPUT FORMAT
22033           *
22034   1731         XR3  ADD        RF                RF   JSL  MY29           ZERO  RF := XRA3; JSB TO TEST IF RESULT > 28
22036   1732         XR7  ADD                          XR21 CSR       SP2B HBF2       SF2 IF SIGN := 000F
22038   1733         XR9  ADD        RG          UBA         ADD       XR6
22039   1734         XR1  ADD        RH                RF   ADD       XR2
22040   1735              JSB  MYST       F2           RG   ADD       XR8             STORE DATA IF ALREADY ALIGN
22042   1736              ADD                     0089 BKX3 SUBL      CTR             REGN POINTER AT MSW
22044   1737              ADD                          RH   JSL  SLD2 XR0       UNC   SHIFT LEFT 2 DIGITS
22046           *
22047           *         STORE THE DATA
22048           *
22049   1738    MYST      SP2B CSL        XR13      0089 BKX3 SUBL      CTR             TARGET SIGN MASK; REGN POINTER AT MSW
22051   1739               ADD            XR14           XR26 ADD       RC              XRA14 := 0, LOST DIGIT REGISTER;
22053                                                                                  RESTORE RC
22054   173A               ADD                 CF3A      XR20 JSL  STOA SP2B      UNC   CF3A FOR LOST DIGIT FLAG; STORE OUT DATA
22056   173B               JSB  ER13           F3A                ADD                   JSB IF LOST DIGIT
22058   173C               ADD                                   JSL  S024        UNC   DO SDEC
22060           *
22061           *         EITHER OPERAND IS ZERO ( F4 := 1 )
22062           *
22063   173D    MZRO       ADD                 CCA       XR22 ADD       BKX6            SET CCE; TARGET WD COUNT - 1
22065   173E               ADD                           XR23 ADD       BKX7            ; TARGET MSW MASK
22067   173F               ADD                                ADD       BKX3            ; BKX3 := 0
22069   1740               ADD                     0089       ADDL      CTR             ; LSW REGN POINTER
22071           *
22072           *         ZERO OUT DATA BUFFERS
22073           *
22074   1741    ZREG 0008       ADDL       RG                XR21 CSR       SP2B            # OF DATA BUFFERS; TARGET SIGN MASK
22076   1742                    ADD                          CTRS ADD       XR0             SAVE INITIAL COUNTER
22078   1743                    ADD        REGN                    ADD       REGN DCTR       REGN := 0; DCTR
22080   1744         RG         JSB  *-1        NZRO  RG          CAD       RG              LOOPBACK IF NOT DONE
22082   1745                    JSB  MYST       UNC          XR0  JSB  FCH2 CTR       NF4B  STORE OUT ZEROS; RTN IF FROM FCHA
```

```
RECORD  C.S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22085           *
22086           *
22087           *
22088           *          DO 1 DIGIT OF THE MULTIPLIER
22089           *
22090   1746    MPYM       XR14 ADD       SP0A             XR15 INC       BKX3 PSHR       M'IER WD LEN; PRODUCT LEN
22092   1747         FFFF  SP4A ADDL      SP4A             REGN JSB  *+9  SP1B      ZERO  BIT POINTER; JSB IF THE M'IER := 0
22094   1748         009B       ADDL      RH               REGN ADD       SP1B LBF5       1X M'CAND; SF5B IF BIT SET
22096   1749              SP4A  ADD                        XR15 JSB  MPAA  SP2B      F5B   DO ADD IF F5B
22098   174A         00A7       ADDL      RH               SP1B ADD  LSR  SP1B LBF5       2X M'CAND;
22100   174B                    ADD                        XR15 JSB  MPAA  SP2B      F5B   DO ADD IF F5B
22102   174C         00B3       ADDL      RH               SP1B ADD  LSR  SP1B LBF5       4X M'CAND;
22104   174D                    ADD                        XR15 JSB  MPAA  SP2B      F5B   DO ADD IF F5B
22106   174E         00BF       ADDL      RH               SP1B ADD  LSR  SP1B LBF5       8X M'CAND;
22108   174F              SP4A  ADD                        XR15 JSB  MPAA  SP2B      F5B   DO ADD IF F5B
22110   1750         SP0A       CAD       SP0A HBF2  00C3 UBA  ADDL      CTR             SF2 IF DONE; NEXT M'IER POINTER
22112   1751              XR2   JSB  *-A         NF2      SP1B ADD  LSR  REGN DCTR       JSB FOR NEXT M'IER; SAVE REMAINDING M'IER
22114   1752              XR4   ADD                       UBA  ADD       XR1  POPR       SAVE DATA IN XRB'S
22116   1753              XR5   ADD                       UBA  ADD       XR3
22117   1754              XR6   ADD                       UBA  ADD       XR4
22118   1755              XR8   ADD                       UBA  ADD       XR5
22119   1756         RE   RC    ADD                       UBA  ADD       XR7       RSB   TOTAL # OF DIGITS; RTN
22121           *
22122           *          DO ADD
22123           *
22124   1757    MPAA       SP4A ADD       XR11 CF1   RH       ADD       CTR  PSHR       SAVE WORD POINTER; PROPER M'CAND POINTER
22126           *
22127           *
22128   1758         FFFE  SP4A ADDL      SP4A             CAD  LSL  SP3B DCTR       SP4A := SP4A - 2; -2, DCTR
22130   1759               REGN ADD       RG         0088  SP4A ADDL      CTR             M'CAND;
22132   175A               REGN ADD                  SP3B  RH   ADD       RH   DCTR       M'CAND; NEXT M'CAND POINTER
22134   175B         UBA   REGN XFRS                  SP3B  SP2B ADD       SP2B HBF2       RESULT; SF2 IF DONE
22136   175C         RREG  REGN DADD      REGN F1TC  RG    UBA  LINK      ICTR            DO ADD & STORE RESULT;
22138   175D         UBB        ADD       REGN             RH   JSB  *-5  CTR       NF2   LOOPBACK IF NOT DONE
22140           *
22141           *
22142   175E                   ADD             CF1              ADD             POPR
22143   175F                   ADD                              ADD
22144   1760               XR11 ADD       SP4A CF3A            ADD             CF2  RSB
```

```
RECORD  C S.          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22147           *
22148           *
22149           *
22150           *        SWAP OPERANDS
22151           *
22152           *
22153   1761    MSWP     UBB  ADD        XR11             UBA  ADD        XR11            WD ADDR;
22155   1762             RC   ADD        RE               BKX4 ADD        BKX6            DIGIT COUNT; WD COUNT
22157   1763         RE  XR13 XFRS       XR12             BKX6 ADD        BKX4            SIGN MASK;
22159   1764         RREG     ADD        RG               BKX5 ADD        BKX7            ; MSW MASK
22161   1765         SPOA     ADD        XR13 RSB         BKX7 ADD        BKX5
22162           *
22163           *
22164           *
22165           *        DO 2X, 4X, OR 8X OF THE MULTIPLICAND
22166           *
22167           *
22168   1766    MPYX 000B     ADDL       SP4A             UBA  ADD        CTR  PSHR       11; REGN POINTER
22170   1767         SPOA RH  INC        RH   HBF2        XR23 ADD        BKX7 DCTR       SF2 IF DONE; TARGET MSW MASK
22172   1768              REGN ADD       RG          000B CTRS ADDL       CTR             DATA; NEW CTR
22174   1769              REGN ADD                   FFF2      ADDL       SP3B            DATA; MASK
22176   176A         UBA  UBA  DADD       REGN CTF1  RG   RG   LINK           ICTR
22177   176B         UBB       ADD        REGN       SP3B CTRS JSBI *+6  CTR       F2    ; JSB IF DONE
22179           *
22180           *
22181   176C         000B      ADDL       SP4A                  CAD  LSL  XR29 DCTR       SP4A := 000B; XRB29 := FFFE
22183   176D              REGN ADD        RG               UBA  CTRS ADD        CTR
22184   176E              REGN ADD                         RH   XR29 ADD        RH   HBF2  ; DEC WD COUNT BY 2
22186   176F         UBA  UBA  DADD       REGN F1TC  RG   RG   LINK           ICTR
22187   1770         UBB       ADD        REGN       SP3B CTRS JSBI *-4  CTR       NF2   ; LOOPBACK IF NOT DONE
22189           *
22190           *
22191   1771                   ADD                   UBB  SP4A ADD        CTR  POPR
22192   1772                   ADD        REGN CF2              ADD        ICTR
22193   1773                   ADD        REGN             XR22 ADD        BKX6 CF1  RSB   ; TARGET WD COUNT - 1
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22196            *
22197            *
22198            *
22199            *          CHECK OPERAND LENGTHS FOR SLD, NSLD, SRD, ADDD,
22200            *          CMPD, SUBD, & MPYD.
22201            *          IF 0 > OPERAND LENGTH >= 29 DIGITS, TRAP TO ER17,
22202            *          OR IF OPERAND LENGTH = 0, DO STACK ADJUSTMENT & NEXT.
22203            *
22204            *
22205     1774   CKLN RC   SP1B JSB  ER17       CRRY  RA    SP1B JSB  ER17          CRRY  ER17 IF RA OR RC >= 29 OR < 0
22207     1775             ADD       XR2                   RA   JSL  S024          ZERO  ; JSB IF RA := 0
22209     1776             ADD       XR3  RSB              RC   JSL  S024          ZERO  RETURN; JSB IF RC := 0
```

```
 RECORD   C.S.          ********** ALU A **********  ********** ALU B **********
   NO     ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22212             *
22213             *
22214             *
22215             *          CHECK TARGET OPERAND ADDRESS
22216             *
22217             *          ON ENTRY :
22218             *
22219             *                       RC := TARGET DIGIT COUNT
22220             *                       RD := TARGET BYTE ADDRESS ( DB REL )
22221             *
22222             *          ON EXIT  :
22223             *
22224             *                       RG := TARGET WORD ADDRESS
22225             *                    BKX6 := TARGET WORD COUNT - 1
22226             *
22227             *
22228     1777  CKBA RD        ADD  LSR                    SM   ADD       RE              TARGET WD-ADDR; RE := SM
22230     1778       UBA  DB  ADD       RG          RD         ADD            LBF5        TARGET WD ADDR; SET F5 IF BYTE ADDR IS ODD
22232     1779       UBA  DL  SUB                NCRY  RC         ADD  LSR  BKX6 LBF5 NF5B  SKIP IF DL > WD ADDR;
22234     177A       RE   RG  SUB                CRRY  UBB        INC       BKX6 LBF5       SKIP IF WD ADDR <= SM; TARGET WD COUNT
22236     177B       RF   RG  ADD       RG               ADD       XR3  CF5B       WRAPAROUND ADDR; XRB3 := 0
22238     177C                ADD       XR4         BKX6 ADD  LSR  BKX6            XRA4 := 0; TARGET WD COUNT - 1
22240     177D                ADD       XR8         UBB  RG   ADD                  XRA8 := 0; LSW ADDR
22242     177E       RG   DL  BNDE            NF3A  RE   UBB  BNDE                BOUNDS CHECK, DL <= MSW ADDR, SM <= LSW ADDR
22244             *
22245             *
22246             *
22247             *          CHECK SOURCE OPERAND ADDRESS
22248             *
22249             *          ON ENTRY :
22250             *
22251             *                       RA := SOURCE DIGIT COUNT
22252             *                       RB := SOURCE BYTE ADDRESS ( DB REL )
22253             *
22254             *          ON EXIT  :
22255             *
22256             *                       RH := SOURCE WORD ADDRESS
22257             *                    BKX4 := SOURCE WORD COUNT - 1
22258             *
22259     177F       RB        ADD  LSR        RSB         SM   ADD       RE              SOURCE WD-ADDR; RE := SM, RTN IF F3 SET
22261     1780  CKB  UBA  DB  ADD       RH          RB         ADD            LBF5        SOURCE WD ADDR; SET F5 IF BYTE ADDR IS ODD
22263     1781       UBA  DL  SUB                NCRY  RA         ADD  LSR  BKX4 LBF5 NF5B  SKIP IF DL > WD ADDR;
22265     1782       RE   RH  SUB                CRRY  UBB        INC       BKX4 LBF5       SKIP IF WD ADDR <= SM; SOURCE WD COUNT
22267     1783       RF   RH  ADD       RH               ADD       XR4  CF5B       WRAPAROUND ADDR; XRB4 := 0
22269     1784                ADD       XR5         BKX4 ADD  LSR  BKX4            XRA5 := 0; SOURCE WD COUNT - 1
22271     1785                ADD       XR9         UBB  RH   ADD                  XRA9 := 0; LSW ADDR
22273                                                                               TICB TO SLOW DOWN RSB AVOID S KILL  (1000 )
22274     1786       RH   DL  BNDE                    RE   UBB  BNDE                (1010)
22276     1787                ADD                         ADD                 RSB   BOUNDS CHECK, DL <= MSW ADDR, SM <= LSW ADDR
```

```
 RECORD  C.S.                ********** ALU A **********  ************** ALU B **************
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22279            *
22280            *
22281            *
22282            *         CHECK SPLIT STACK
22283            *
22284            *         ON ENTRY :
22285            *
22286            *                     RB  := SOURCE BYTE ADDRESS ( DB REL )
22287            *                     RD  := TARGET BYTE ADDRESS ( DB REL )
22288            *
22289            *         ON EXIT  :
22290            *
22291            *                   XRA11 := SOURCE WORD ADDR
22292            *                   XRB11 := TARGET WORD ADDR
22293            *
22294            *
22295     1788   SPLT RB       ADD  LSR  RG         RD       ADD  LSR  SP3B PSHR       SOURCE WORD ADDR; TARGET WORD ADDR
22297     1789                JSB  *+4       FSS        DL   ADD                       JSB IF SPLIT BANKS;
22299     178A       UBB  DB  JSBC *+3       CRRY  Z    DB   JSBS *+3           NCRY  JSB IF DL >= DB; OR DB > Z
22301     178B                ADD                                ADD            POPR NZRO  DO POPR
22303     178C                ADD                                ADD                 RSB   ;RETURN
22305     178D       RG   DB  ADD       XR11       SP3B DB   JSB  *-2  XR11        UNC   SOURCE WORD ADDR; TARGET WORD ADDR
```

```
RECORD  C.S.                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22308           *
22309           *
22310           *
22311           *         GET TARGET OPERAND MASKS
22312           *
22313           *         ON ENTRY :
22314           *
22315           *                     RC := TARGET DIGIT COUNT
22316           *                     RD := TARGET BYTE ADDRESS ( DB REL )
22317           *
22318           *         ON EXIT  :
22319           *
22320           *                   BKX7 := TARGET MSW MASK
22321           *                  XRA13 := TARGET SIGN MASK
22322           *
22323           *
22324   178E    MASK RC       CSR       SP4A HBF2  OFFF       ADDL      BKX7          SET F2 IF DIGIT COUNT IS ODD; MASK
22326   178F         0F00     ADDL      XR13       OFFF       ADDL      BKX5    NF2   MASKS; SKIP IF NF2
22328   1790         RD   SP4A ADD             EVEN  -         CAD       BKX7          SKIP IF SIGN IN LEFT BYTE; MSW MASK
22330   1791         XR13 ADD  SWAB XR13             RD        ADD             EVEN  SIGN MASK; SKIP IF BYTE ADDR IS EVEN
22332   1792              ADD       XR7  NF3A        BKX7 ADD  LRZ  BKX7          ; UBB := TARGET MSW MASK
22334           *
22335           *
22336           *
22337           *         GET SOURCE OPERAND MASKS
22338           *
22339           *         ON ENTRY :
22340           *
22341           *                     RA :=  SOURCE DIGIT COUNT
22342           *                     RB :=  SOURCE BYTE ADDRESS ( DB REL )
22343           *
22344           *         ON EXIT  :
22345           *
22346           *                   BKX5 :=  SOURCE MSW MASK
22347           *                  XRA12 :=  SOURCE SIGN MASK
22348           *
22349           *
22350   1793              ADD             RSB       ADD       XR6
22351   1794    MSKB 0F00 ADDL      XR12  RA        CSR             POS   MASK; SKIP IF DIGIT COUNT IS EVEN
22353   1795         RB   UBB ADD             EVEN  CAD       BKX5          SKIP IF SIGN IN LEFT BYTE; MSW MASK
22355   1796         000F ADDL      XR12  RB        ADD             EVEN  SIGN MASK; SKIP IF BYTE ADDR IS EVEN
22357   1797              ADD             RSB       BKX5 ADD  LRZ  BKX5          RETURN; UBB := SOURCE MSW MASK
```

```
RECORD  C S                ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22360           *
22361           *
22362           *
22363           *          CALCULATE THE ACC LENGTH
22364           *
22365           *          ON ENTRY :
22366           *
22367           *                     XRA12 := SOURCE SIGN MASK
22368           *                     XRA13 := TARGET SIGN MASK
22369           *
22370           *          ON EXIT  :
22371           *
22372           *                      BKX3 := TARGET WORD COUNT - 1
22373           *
22374           *
22375   1798    LEN        XR12 ADD             EVEN  UBA        ADD        BKX3 PSHR EVEN  SKIP IF SOURCE SIGN MASK := 0F00;
22377                                                                                       SKIP IF TARGET SIGN MASK := 0F00
22378   1799         RG          ADD  LSR  RH   UNC   RC         ADD  LSR   SP2B       UNC   UBA := SLEN; UBB := TLEN
22380   179A         UBA         INC       RH         UBB        INC        SP2B             UBA := SLEN + 1; UBB := TLEN + 1
22382   179B               XR12 ADD             EVEN        BKX3 JSB  *+2                ODD   JSB IF SOURCE SIGN MASK := 0F00;
22384                                                                                       JSB IF TARGET SIGN MASK := 000F
22385   179C         RH          INC       RH                    ADD                      UBA := SLEN + 2;
22387   179D                     ADD       XR10             SP2B ADD  LSR   BKX3 POPR      ;SHIFT RIGHT
22389   179E         RH          ADD  LSR                   RH   SP2B SUB                NEG   UBA := SLEN (SHIFTED); SKIP IF RH <= SP2B
22391   179F                     ADD                  RSB   UBA        ADD        BKX3         RETURN, UBB := TARGET WORD COUNT - 1
```

```
 RECORD   C.S.                ********** ALU A **********  ********** ALU B **********
   NO     ADDR   LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22394             *
22395             *
22396             *
22397             *          FETCH SOURCE OPERAND DATA
22398             *
22399             *          ON ENTRY :  XRA12 := SOURCE SIGN MASK
22400             *                       SP0A := SOURCE WORD ADDRESS
22401             *                       BKX4 := SOURCE WORD COUNT - 1
22402             *                       BKX5 := SOURCE MSW MASK
22403             *
22404             *          ON EXIT  :  DATA WILL BE STORED INTO XRA1 - XRA9,&
22405             *                      THE SAME DATA ARE ALSO STORED INTO
22406             *                      XRB0 - XRB8.
22407             *
22408             *
22409    17A0   FCHB      XR12 ADD       SP4A          BKX4 ADD          PSHR       SIGN MASK; WD COUNT - 1
22411    17A1        SP0A UBB  ADD       RH    ROBD  0089      ADDL      CTR        READ LSW; LSW POINTER
22413    17A2        SP0A UBA  SUB             CTF1        BKX5 JSB  FCH2 SP1B       UNC   SET F1 IF ONE WD FETCH; DO FETCHING
22415             *
22416             *
22417             *          FETCH OPERAND-2 DATA
22418             *
22419             *          ON ENTRY :  XRA13 := OPERAND-2 SIGN MASK
22420             *                       BKX6 := OPERAND-2 COUNT - 1
22421             *                       BKX7 := OPERAND-2 MSW MASK
22422             *
22423             *          ON SUBROUTINE RETURN, DATA WILL BE STORED IN
22424             *          XRA16 TO XRA24, AND ALSO XRB17 TO XRB25.
22425             *
22426             *
22427    17A3   FCHA UBB       ADD       SP0A          UBB  BKX6 ADD       RH   ROD        UBA := WORD ADDR; READ LSW
22429    17A4        UBA  UBB  SUB             CTF1          BKX7 ADD       SP1B PSHR       SET F1 IF ONE WD FETCH;
22431    17A5             JSB  ZREG            MEDJ  0099      ADDL      CTR              ZERO OUT DATA BUFFERS; LSW POINTER
22433             *
22434             *
22435    17A6   FCH2 FFF0       ADDL      RG            0089 BKX3 SUBL      XR13             MASK; MSW REGN POINTER
22437    17A7             SP4A ADD             ODD                CAD       SP3B             SKIP IF SIGN MASK := 000F; -1
22439    17A8        RG        ADD   RLZ  RG            UBA  OPB  AND       SP2B             MASK; MASK SIGN DIGIT
22441    17A9             RH   ADD             ROD   000F UBB  ANDL                  NZRO  READ LSW; SKIP IF SIGN IS IN RIGHT BYTE
22443    17AA                  INC            RG    CCA        SP2B ADD  SWAB SP2B       SET CCG; MOVE SIGN DIGIT TO LEFT BYTE
22445    17AB        RG   OPA  AND            RG            UBA       ADD       XR31     MASK THE LSD'S; XRB31 := 0001
22447    17AC        RG        CSR                         000D SP2B SUBL                 NZRO  ; SKIP IF SIGN IS NEG
22449    17AD             ADD                                  UBA  ADD       XR31 CCA        ; XRB31 := 8001
22451             *
22452             *
22453    17AF        SP0A      JSB  *+4           F1    RH        CAD       RH   ROAD       JSB IF DONE; READ NEXT DATA
22455    17AF             RG   DADD            REGN        UBA       ADD            DCTR     STORE DATA INTO REGN; DEC REGN POINTER
22457    17B0        UBA  RG   JSBS ER15        NZRO  UBB  RH   SUB                 CTF1     JSB IF INVALID DECIMAL DIGIT; SET F1 IF DONE
22459    17B1             OPA  ADD             RG                  RG   JSB  *-3  REGN       UNC   DATA; STORE INTO REGN, DO NEXT
22461             *
22462             *
22463    17B2        RG   SP1B AND             REGN          SP3B CTRS ADD       CTR  POPR       MASK MSD'S; SET MSW REGN POINTER
22465    17B3             UBA  DADD                  CF1           UBA  ADD       REGN DCTR       STORE
22467    17B4                  ADD                               UBA  UBB  JSBS ER15 REGN       NZRO  ; JSB IF INVALID DECIMAL DIGIT
22469    17B5             STA  ADD                   RSB                     ADD                    STATUS, RETURN;
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22472           $NOWARN
22473           *
22474           *
22475           *          STORE DATA INTO MEMORY
22476           *
22477           *          ON ENTRY :
22478           *
22479           *                       SP2B   := TARGET WORD ADDRESS
22480           *                       BKX3   := ACC LENGTH - 1
22481           *                       BKX6   := TARGET WORD COUNT - 1
22482           *                       BKX7   := TARGET MSW MASK
22483           *                      XRB11   := TARGET WORD ADDRESS
22484           *
22485           *          DATA WILL BE STORED FROM XRA1 - XRA9 INTO MEMORY
22486           *          LOCATIONS, WITH STARTING ADDRESS AT SP2B.  REGN
22487           *          POINTER WILL BE ADJUSTED SO THAT ONLY THE APPROPRIATE
22488           *          DATA WILL BE STORED.
22489           *
22490           *          ON EXIT  :  XRA14  := ACC LOST DIGITS
22491           *                      XRB15  := ACC DIGIT SUM
22492           *                      F3     := 1 IF XRA14 NOT ZERO ( DIGITS LOST )
22493           *
22494           *
22495    17B6   STOA      XR14 ADD        SPOA CF1        BKX3 ADD             PSHR        SAVE ACC LOST DIGITS; ACC LEN - 1
22497    17B7            SP2B ADD             ROD  UBB   BKX6 JSBS STOE RG         ZERO  READ TARGET ADDR; JSB IF ACC LEN := TARG LEN
22499    17B8                 ADD                  UBB        JSB  STOG             POS   STOG IF ACC LEN > TARGET LEN
22501    17B9                 JSB  STOE       MEDJ UBB   CTRS ADD       CTR             STOE IF LESS; ADJUST REGN POINTER
22503           *
22504           *
22505    17BA   STOG SPOA REGN IOR        XR14       RG         CAD       RG   ICTR ZERO  SAVE DIGITS LOST REG; DEC REGN POINTER
22507    17BB        UBA      ADD        SPOA                  JSB  *-1             UNC   DIGITS LOST REG; LOOPBACK TILL MSW POINTER
22509           *
22510           *
22511    17BC   STOE      REGN ADD        SP4A            BKX7 CAD             CF1         MSW := MSW MASK
22513    17BD        UBB  OPA  AND        SP4A       UBA  BKX7 AND       RH              TARGET WORD; MSD'S
22515    17BE                 ADD                     UBB  SP4A RSUB                       CHECK IF DIGIT LOST
22517    17BF        UBB  XR14 IOR        XR14 ZERO                ADD       XR15            SKIP IF DIGITS LOST; XRB15 := 0, ACC DIGIT
22519    17C0                 CAD             SF3A            BKX8 ADD       SP3B CF2        SET F3 FOR DIGITS LOST; WD COUNT - 1
22521           *
22522           *
22523    17C1            XR13 CSR                 HBF2 UBA   UBB  JSB  *+4  RG         NEG   SET F2 IF SIGN := 000F; JSB IF ONE WD
22525    17C2        RH   SP4A IOR                 DATA RH    XR15 IOR             XR15 ICTR  STORE INTO DATA; ACC DIGIT SUM
22527    17C3                 ADD             SP4A      RG         CAD       RG   CTF1        SP4A := 0; LOOPCOUNT - 1
22529    17C4            REGN ADD             RH                  JSB  *-2             F1    NEXT WORD; LOOPBACK IF NOT DONE
22531           *
22532           *
22533    17C5                 ADD                 NF2  SP3B SP2B ADD             ROAD F2     READ TARGET LSW
22535    17C6        FFFO RH  ANDL            RH      F000 RH   ANDL       RH              RH := LSD'S
22537    17C7             STA ADD  LSR             BIT8 000D      ADDL      RG              SKIP IF CCL; SIGN
22539    17C8        000C     ADDL            RG          RH    XR15 IOR       XR15  NZRO  SIGN; SKIP IF ACC DIGIT SUM NOT ZERO
22541    17C9             UBB ADD             SPOA F2     000C       ADDL      RG          SAVE ACC DIGIT SUM; SIGN
22543    17CA             OPA ADD  RRZ        SP4A CF1                ADD                   POPR F2     TARGET DATA, SKIP IF F2
22545    17CB        SPOA     ADD             XR15 NZRO   RG          ADD  SWAB RG          CF2         SKIP IF ACC DIGIT SUM IS NOT ZERO; SWAB SIGN
22547    17CC                 ADD                  CCA    RH   SP4A IOR                  CF2         SET CCE, TARGET DATA
22549    17CD                 ADD                         UBB  RG   IOR                  DATA RSB    TARGET DATA & PROPER SIGN
22551           $WARN
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22553           *
22554           *
22555           *
22556           *         TRAP ROUTINES ---
22557           *
22558           *         1. ER13 IF PACKED-DECIMAL OVERFLOW.
22559           *         2. ER15 IF INVALID PACKED-DECIMAL DIGIT.
22560           *         3. ER17 IF DIGIT COUNT > 28, OR NEGATIVE.
22561           *
22562           *
22563    17CE   ER13            ADD           SF3A            JSB  *+6            UNC   SET F3; JSB
22565    17CF   ER15 0005 DSPL  XORL      RG                  ADD                       IS IT FROM CVDB?
22567    17D0                   ADD                    0009 UBA  SUBL                 NZRO  ; SKIP IF FROM MPYD
22569    17D1                   ADD           CF1        XR26 ADD       RA             ; RESTORE RA
22571    17D2                   JSB  *+2      CF3A       RG   JSL  DBER              ZERO  JSB DBER IF FROM CVDB; ELSE JSB *+2
22573    17D3   ER17            ADD           CF3A            ADD            SF1         CF3A; SF1
22575    17D4        2000 STA   ANDL                          ADD            CRF         USER TRAP ENABLED ? CRF
22577    17D5                   JSZ  TRPO     F3A        UBA  JSB  *+2               ZERO  JSB *+2 IF USER TRAP NOT ENABLED;
22579                                                                                      ELSE JSB TO TRAP ROUTINE
22580    17D6                   JSZ  TRPR     UNC             JSZ  TRPD              NF1   TRAP TARGETS
22582           *
22583           *
22584    17D7        0800 STA   XORL      STA                 ADD                       SET OVERFLOW
22586           *
22587           *
22588           *
22589           *         S024 --- WILL POP THE STACK BY 0, 2, OR 4 TIMES.
22590           *
22591           *                  IF CIR ( 10:2 ) := 00, DECREMENT BY 0,
22592           *                                  := 01, DECREMENT BY 2, OR
22593           *                                  := 10, DECREMENT BY 4.
22594           *
22595           *
22596    17D8   S024 0030 CIR   ANDL                          ADD            CRF         CHECK CIR ( 10:2 ); CRF
22598    17D9        0020 CIR   ANDL                     UBA  JSZ  NEXT         ZERO  CHECK CIR ( 10:1 ); DO NEXT IF 00
22600    17DA                   ADD                    UBA  ADD               EPP2 ZERO  ; POP2, SKIP IF CIR ( 10:1 ) := 0
22602    17DB                   JSZ  NEXT     UNC             ADD            EPP2        JSZ NEXT; POP2 MORE
```

```
RECORD  C S          ********** ALU A **********  ********** ALU B **********
  NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22605           *
22606           *
22607           *         SCAN SOURCE OPERAND FOR LEADING ZEROS
22608           *
22609           *         ON ENTRY :  XRA11 := SOURCE WORD ADDRESS
22610           *                     BKX4 := SOURCE WORD COUNT - 1
22611           *                     BKX5 := SOURCE MSW MASK
22612           *
22613   17DC    SCNB      XR11 ADD             ROD   RA        ADD        RH   PSHR ODD   READ SOURCE (MSW); SKIP IF # DIGIT IS ODD
22615   17DD         RB        CSR             HBF2  UBB       INC        RH             ; F2 IF BYTE ADDR IS ODD; ADD 1 IF EVEN
22617   17DE                   ADD                         BKX5 ADD                 NF2    ; SKIP IF DIGIT ADDR IS EVEN
22619   17DF         UBB  OPA  AND        SP4A  CF2   0002 RH   ADDL       RH             MASK MSD'S; ADD 2 IF ODD
22621   17E0         UBA       ADD                         BKX4 ADD
22622           *
22623           *
22624   17E1         UBB       JSB  SCND      ZERO        UBA  JSB  SCND             NZRO  DO SCAN; JSB  IF WD COUNT := 0, OR
22626                                                                                      NON-ZERO IS DETECTED
22627   17E2              XR11 INC       XR11  ROD               ADD                       READ NEXT ADDRESS
22629   17E3         FFFC RH   ADDL      RH          FFFF        ADDL       BKX5          DIGIT COUNT - 4; UBB := -1
22631   17E4         UBB  OPA  AND       SP4A        RREG BKX4   JSB  *-3   BKX4   UNC   MASK; DEC THE WD COUNT
22633           *
22634           *
22635   17E5    SCND RH        ADD              NZRO  F000 SP4A   ANDL             ZERO  ; CHECK IF 1ST DIGIT := 0
22637   17E6                   JSB  SCNE        UNC               JSB  SCNE        UNC   ; JSB IF NO ZERO
22639   17E7         RH        CAD       RH     NZRO  0F00 SP4A   ANDL             ZERO  CHECK FOR 2ND DIGIT
22641   17E8                   JSB  SCNE        UNC               JSB  SCNE        UNC
22642   17E9         RH        CAD       RH     NZRO  00F0 SP4A   ANDL             ZERO  CHECK FOR 3RD DIGIT
22644   17EA                   JSB  SCNE        UNC               JSB  SCNE        UNC
22645   17EB         RH        CAD       RH                       ADD
22646           *
22647           *
22648   17EC    SCNE           ADD         XR6   NF2             ADD             POPR NZRO  XRA6 := 0; POPR
22650   17ED         RH        ADD         RC    CF2             ADD                  RSB   NEW TARGET DIGIT COUNT; RETURN
22652           *
22653           *
22654           *         SCAN OPERAND-2 FOR LEADING ZEROS
22655           *
22656           *         ON ENTRY :  XRB11 := OPERAND-2 WORD ADDRESS
22657           *                     BKX6 := OPERAND-2 WORD COUNT - 1
22658           *                     BKX7 := OPERAND-2 MSW MASK
22659           *
22660   17EE    SCNA RC        ADD         RH    ODD             XR11 ADD             ROAD  SKIP IF OPERAND-2 DIGIT COUNT IS ODD;
22662                                                                                      READ OPERAND-2 MSW
22663   17EF         UBA       INC         RH                    BKX7 ADD             PSHR  INC DIGIT COUNT BY 1; OP-2 MSW MASK
22665   17F0         UBB       ADD               SF2   RD              ADD                  EVEN  SF2; SKIP IF EVEN BYTE ADDR
22667   17F1         UBA  OPA  AND         SP4A        0002 RH   ADDL       RH             MASK MSW DATA; ADD 2 TO DIGIT COUNT
22669   17F2         UBA       ADD                         BKX6 ADD
22671           *
22672           *
22673   17F3              UBB  JSB  SCND      ZERO        UBA  JSB  SCND             NZRO  JSB IF EITHER DATA OR WD COUNT IS ZERO
22675   17F4                   ADD                         XR11 INC       XR11 ROAD        ; READ NEXT ADDRESS
22677   17F5         FFFC RH   ADDL      RH          FFFF        ADDL       BKX7          DIGIT COUNT - 4; NEW MSW MASK
22679   17F6         UBB  OPA  AND       SP4A        RREG BKX6   JSB  *-3   BKX6   UNC   MASK THE DATA; DEC WD COUNT, DO NEXT WORD
22681   17F7                   ADD                               ADD                       NOP to keep system happy
22683   17F8                   ADD                               ADD                       NOP to keep system happy
```

```
 RECORD   C S                ********** ALU A **********  ********** ALU B **********
   NO     ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22686           ************************************************************************
22687           *                                                                      *
22688           *    ABSD / NEGD                                                       *
22689           *                                                                      *
22690           *    1.  Get two valid TOS registers containing the relative byte      *
22691           *        byte address to the packed decimal field and the digit        *
22692           *        count.                                                        *
22693           *    2.  If the digit count is less than zero or greater than 28       *
22694           *        then perform the ILLEGAL DECIMAL OPERAND (%17) TRAP.          *
22695           *    3.  Fetch the word containing the sign nibble at the              *
22696           *        effective address EA = (DB + (DC / 2)) / 2.                   *
22697           *    4.  Bounds check the effective address if not in split stack      *
22698           *        mode.  If the bounds test fails, add 32k to check for a       *
22699           *        DL-DB address, refetch the word, and bounds check again.      *
22700           *    5.  If the sign nibble is in the left byte, swap the left         *
22701           *        and right bytes so the sign nibble will be in the right       *
22702           *        nibble.                                                       *
22703           *    6.  Extract the last 2 bits of the sign (right) nibble            *
22704           *        (C = 00, D = 01, F = 11) of the right byte.                   *
22705           *    ABSD                                                              *
22706           *                                                                      *
22707           *    7.  Make the value unsigned by masking an F into the sign         *
22708           *        nibble.                                                       *
22709           *    8.  Set CCG.                                                      *
22710           *    9.  If the sign bits = 01 (the sign was negative => D), then      *
22711           *        set CCL.                                                      *
22712           *                                                                      *
22713           *    NEGD                                                              *
22714           *                                                                      *
22715           *    7.  Remove the sign by masking a zero into the sign nibble.       *
22716           *    8.  Set CCG and mask a D into the sign nibble.                    *
22717           *    9.  If the sign bits = 01 (the sign was D => negative), then      *
22718           *        then set CCL and mask a C into the sign nibble.               *
22719           *                                                                      *
22720           *    ABSD / NEGD                                                       *
22721           *                                                                      *
22722           *    10.  Pop the digit count.                                         *
22723           *    11.  Pop the relative base address parameter if SDEC = 1.         *
22724           *    12.  Clear the overflow bit in the STATUS register.               *
22725           *    13.  If the left and right bytes were swapped in Step 5,          *
22726           *         then swap them back.                                         *
22727           *    14.  Rewrite the word containing the sign nibble.                 *
22728           *    15. Next instruction.                                             *
22729           *                                                                      *
22730           *                                                                      *
22731           ************************************************************************
22732           *
22733           $NOWARN
```

```
 RECORD  C S          ********** ALU A **********  ********** ALU B **********
   NO    ADDR  LABL RREG SREG FUNC SFNC STOR SPSK  RREG SREG FUNC SFNC STOR SPEC SKIP  COMMENT

22737          %1800
22738    1800  ABSD                 ADD             CF1                  ADD                          F1 DETERMINES ABSD OR NEGD
22740    1801  NEGD                 JSZ   PULM      SRL2  SR   SM    JSZ   PUL2 SP3B       SRZ     NEED TWO VALID TOS REGISTERS CONTAINING
22742                                                                                                 RBA OF PACKED DECIMAL FIELD AND DIGIT COUNT
22743    1802                       ADD             FFE3                 ADDL                         MAXIMUM NUMBER OF DECIMAL DIGITS IS 28
22745    1803       RA   UBB  JSB   ABN2    CRRY  RA                   ADD   LSR                    TRAP IF LENGTH < 0 OR LENGTH > 31
22747                                                                                                 DC/2
22748    1804       RB   UBB  ADD   LSR     SF3A  RB   UBB   CSR                  HBF2             (RBA + (DC/2))/2
22750                                                                                                 F2:=SIGN NIBBLE IN LEFT OR RIGHT BYTE
22751    1805       UBA  DB   ADD         RH  ROD                  JSB   *+3                NF5B    FETCH WORD CONTAINING SIGN NIBBLE
22753                                                                                                 SKIP BOUNDS TEST IF SPLIT STACK MODE  (CSTX)
22754    1806       UBA  DL   SUB               CRRY  SP3B UBA    SUB                  CRRY    BOUNDS TEST DL <= EA                  (2324)
22756                                                                                                 BOUNDS TEST EA <= S                   (2324)
22757    1807                 JSB   *+0         UNC                  JSL   LDWC           UNC     JUMP TO SELF; JUMP TO BOUNDS CHECK    (CSTX)
22759    1808            OPA  ADD         SP4A F2   000F            ADDL                         SIGN NIBBLE IN RIGHT BYTE
22761                                                                                             MASK FOR SIGN NIBBLE
22762    1809            OPA  ADD   SWAB  SP4A        UBA   UBB   AND        RH        F2       SIGN NIBBLE IN LEFT BYTE
22764                                                                                             MASK SIGN BITS IN RIGHT BYTE
22765    180A                 JSB   ABN1        F1    UBA   SREG  AND        RH                 ABSD/NEGD ?
22767                                                                                             MASK SIGN BITS IN LEFT BYTE
22768    180B       000F SP4A IORL        RH          000D            ADDL                       ABSD - CHANGE SIGN NIBBLE TO F
22770                                                                                             SIGN NIBBLE FOR A NEGATIVE NUMBER
22771    180C       RH   UBB  SUB                 NZRO                 INC                CCA     TEST FOR SIGN NIBBLE = D => NEGATIVE
22773                                                                                             SET CCG DEFAULT
22774    180D                 CAD                 CCA                  JSB   ABN2           UNC     SET CCL
22776                                                                                             SKIP NEGD OPTION
22777    180E  ABN1 FFF0 SP4A ANDL        SP4A        000D            ADDL                       NEGD - CHANGE SIGN NIBBLE TO 0
22779                                                                                             SIGN NIBBLE FOR A NEGATIVE NUMBER
22780    180F       RH   UBB  SUB                 NZRO                 CAD                CCA     TEST FOR SIGN NIBBLE = D => NEGATIVE
22782                                                                                             SET CCL DEFAULT
22783    1810                 INC                 CCA   UBA            ADD                NZRO    SET CCG
22785                                                                                             DETERMINE WHICH SIGN TO REPLACE PREVIOUS
22786    1811       000D SP4A IORL        RH          000C SP4A IORL       RH                     MASK NEGATIVE SIGN IF PREVIOUSLY POSITIVE
22788                                                                                             MASK POSITIVE SIGN IF PREVIOUSLY NEGATIVE
22789    1812  ABN2                 ADD                 F2         SP1B ADD                EPOP EVEN SIGN NIBBLE BELONG IN LEFT OR RIGHT BYTE?
22791                                                                                             POP DIGIT COUNT; SDEC 0 OR 1?
22792    1813       RH             ADD   SWAB RH                        ADD                EPOP    SIGN NIBBLE SHOULD BE IN LEFT BYTE
22794                                                                                             POP RBA IF SDEC = 0
22795    1814                      JSZ   TRPR        NF3A                 ADD                CLO     ILLEGAL DECIMAL OPERAND LENGTH TRAP  (%17)
22797                                                                                             CLEAR OVERFLOW IN STATUS REGISTER
22798    1815       RH             ADD               DATA                 ADD                NEXT    REWRITE WORD CONTAINING SIGN NIBBLE
22800                                                                                             NEXT INSTRUCTION
22801          $WARN
22802    1816                      ADD                                    ADD                        NOP's to prevent WCS Parity errors from
22804    1817                      ADD                                    ADD                          crashing system during instruction pre-fet
22806          *
22807          *
22808          ********************* END OF ICF/55 MICROPROGRAM *********************
```

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

ABCS	005D	<-	0150							
ABDS	005C	<-	057A	057B	05AE					
ABN1	180E	<-	180A							
ABN2	1812	<-	1803	180D						
ABRT	0DC6	<-	0DAF							
ABS0	1153	<-	1143							
ABS1	1156	<-	114D							
ABS2	115E	<-	113B	113C	114F	114F	1158	1158		
ABSD	1800	<-	101E							
ABSN	113A	<-	1008							
ABT1	0DC9	<-	0DB9							
ABTH	00C4	<-	014F							
AD	0939	<-	0945	0960						
AD14	1494	<-	142A	142D	1430	1438	1445			
AD17	1495	<-	1417	1417						
ADAX	03EB									
ADBX	03F1									
ADDD	1583	<-	1408	1409	140A					
ADDE	159F	<-	15AA	15B0						
ADDM	1592									
ADJ1	0F1A	<-	0F16							
ADJS	0F14	<-	08D3	0B3C	0F19	0F1C	10A2	1100	1163	1204	1257
ADS1	137B	<-	1378							
ADSB	066F									
ADSC	057B	<-	0577	0590						
ADSD	05AE	<-	05A6							
ADSE	057A	<-	0571							
ADSI	0658									

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

ADSS	1375									
ADXA	03EC									
ADXB	03F2									
AE	093A	<-	0936	0937						
AE13	1577	<-	15A0							
AEBC	11A5	<-	113E	113E	116D	116D	117A	117A	117D	117D
AIX1	055A	<-	0557							
AIXF	0559	<-	0551							
AIXR	0557	<-	0555							
ALG1	119B	<-	1184	118B	118D					
ALG2	11A1	<-	1165	1165	1167	1167	1168	1168	119E	
ALGN	1162	<-	1007							
AMD	0744									
AMD1	0745	<-	074C							
AMID	0749									
AMIP	0750									
AMIS	0748									
AMP	074D									
AMP1	074E	<-	0752							
AMS	0743									
AND	03FB									
ANDI	066E									
ASL	0876									
ASMN	0746	<-	0755							
ASR	087A									
ASXI	066B									
BAF0	043C	<-	0439	043E						
BAF1	043E	<-	043B							

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

BAF2	043F	<-	043C						
BAF3	0441	<-	0419	0422					
BAF4	0443	<-	0414	0418	041C	041E	043C	043F	
BAR0	0449	<-	0446	044B					
BAR1	044B	<-	0448						
BAR2	044C	<-	0449						
BAR3	044E	<-	042D	0436					
BAR4	0450	<-	0428	042C	0430	0432	0449	044C	
BBF0	041C	<-	0421						
BBF1	0422	<-	0420						
BBFC	0413	<-	0411						
BBR0	0430	<-	0435						
BBR1	0436	<-	0434						
BBRC	0427	<-	0425						
BCC	05F0								
BCC1	05F1	<-	05E8	05EF	05FC				
BCCI	05F3								
BCI1	05BF	<-	05CE						
BCI2	05C2	<-	05CC						
BCY	05B9								
BCY1	05BA	<-	05CD						
BCYI	05BE								
BD1	15D4	<-	15CD						
BD2	15E5	<-	15CE						
BD3	15ED	<-	15CE						
BD33	15BA	<-	15DD						
BD3S	15DF	<-	14AF	14C5	14E0	14F6	15EC	15F7	
BD4	149B	<-	14BC	15D0					

BD5	14B0	<-	14D5	15D1																	
BD6	14C6	<-	14F1	15D2																	
BD6R	14D6	<-	14BF																		
BD7	14E1	<-	15D3																		
BDCC	00D7	<-	07C6																		
BDEN	15DC	<-	15E1	15E3	15E4																
BDER	16D9	<-	1494	1495	15BA	15BB															
BDTR	14F2	<-	14DF																		
BIDE	0C39	<-	0C20																		
BKPT	0603	<-	0326																		
BNC1	05C5	<-	05CF																		
BNCI	05C9																				
BNCY	05C4																				
BNDL	0B48	<-	0B4C																		
BNDV	0057	<-	0025	00E7	00FB	010F	016F	01CA	037B	03C2	049E	049F	04CA	04CB	04CC	04CD	04DE	04EF	04F0	05D9	
			05E2	060C	0612																
BNI1	05CA	<-	05D0																		
BNOI	05D0																				
BNOV	05CF																				
BOV	05CD																				
BOVI	05CE																				
BRE	05DA																				
BREI	05DE																				
BRID	05F9																				
BRIS	05F8																				
BRO	05D1																				
BROI	05D5																				
BRS	12CD	<-	1274																		
BSFO	0414	<-	0417																		

BSF2	0418	<-	0414										
BSF3	0419												
BSR0	0428	<-	042B										
BSR2	042C	<-	0428										
BSR3	042D												
BTOD	14F7	<-	14A2	14A7	14AA	14BA	14D4	14F0	15D5	15E8	15EB	15F2	15F5
BTST	13F1												
BYR1	0AAD	<-	0AAB										
BYRS	0AA8	<-	0A6D	0A8B									
CAB	13D6												
CABT	108F	<-	1088	1088									
CAD1	1421	<-	141F										
CAD2	1431												
CAD3	1432	<-	142B										
CAD4	1433	<-	142C										
CAD5	1435	<-	143E	1440									
CAD6	1439	<-	1434	1441									
CAD7	1441	<-	1427	1429	142F								
CAD8	144D	<-	143C	1448									
CAD9	1450	<-	144D										
CADA	1442	<-	1435	144A	144C								
CADB	1445	<-	1438										
CADC	1446	<-	1431	1442									
CAF1	046B	<-	0470										
CAF2	046E	<-	046A										
CAR1	047A	<-	047F										
CAR2	047D	<-	0479										
CBAF	046A	<-	0462										

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

CBAR	0479	<-	0471													
CBE	07F3	<-	07D9	07DC	07DF	07E9										
CBE1	07F4	<-	07CA													
CBFD	02EB															
CBSF	0463															
CBSR	0472															
CCL	0270	<-	0297													
CCPA	08F9	<-	08D6	08DD												
CCPI	08F7	<-	08F0													
CCPP	08D3	<-	0D0C													
CCPR	08E6															
CCPW	08ED	<-	08E5													
CEBC	1253	<-	1208	1208	120B	120B	125F	125F	1262	1262	127F	127F	12E0	12E0	12E7	12E7
CGE2	0A95	<-	0A2F													
CGET	0A98	<-	0A95													
CHL0	00D0	<-	00B3													
CHR	094A	<-	092A	092D	0937	0941	0948	09E2	0A5D							
CHR1	094D	<-	094B													
CHRA	099A	<-	0941	0942	094D											
CHRC	09AF	<-	099E													
CHRS	09C5	<-	095F	09C7	09FA											
CHRX	09C8	<-	0960	09BA	0ABC											
CHRZ	0AB6	<-	09C7													
CINR	02A2	<-	02A2													
CINT	0298	<-	024B	0261												
CIRD	06E4	<-	06D9													
CKA1	160D	<-	1607	1608	1609	160B										
CKAB	145E	<-	141E	146C												

CKB	1780	<-	1602						
CKBA	1777	<-	1502	1518	1585	15BE	1702		
CKCD	1456	<-	141B	146A					
CKCH	0F0E	<-	0AC9						
CKLN	1774	<-	1501	1517	1584	1701			
CKPB	01FA	<-	0235						
CKSM	02D4	<-	0002	0791					
CLCT	02FD	<-	0504						
CLD0	0243	<-	0209						
CLD1	0247								
CLDD	02AF								
CLDE	0002	<-	0F0D						
CLDR	0264	<-	02AE						
CLDT	01DE								
CLER	0984	<-	0909						
CLFF	09B0	<-	0A4A	0A51					
CLFN	027B								
CLIF	09B6	<-	09B9	09F3	09F4	09F7	09F8	09F8	
CLKI	0D00	<-	053D						
CLKX	0D03								
CLLP	0272	<-	0274	09AD					
CLP	0275	<-	0273						
CLSO	122C	<-	1214						
CLSS	1220	<-	1213						
CM10	10C4	<-	10C2	10C7					
CM11	10CB	<-	10B6						
CM12	10CD	<-	10AE						
CM13	10CE	<-	10D5						

CM14	10CF	<-	10CC	10D3										
CM15	10D6	<-	10B4											
CM16	10E4	<-	10C5											
CM17	10ED	<-	10C1	10D1	10FD									
CM18	10F3	<-	10A5											
CM19	10F7	<-	10A8	10A8	10AC	10AC	10D8	10D8	10DF	10DF	10E8	10E8	10EF	10EF
CM20	10FA	<-	10B5	10C6	10D2									
CM21	10FB	<-	10A4	10C0	10C6	10D2								
CM22	10D4	<-	10CC											
CM26	10E6	<-	10CB	10CD										
CM30	10FD	<-	10AA											
CMB0	0488	<-	0453	045C										
CMB1	0460	<-	045B											
CMBD	045C													
CMBE	0480	<-	0466	0469	046D	0470	0475	0478	047C	047F				
CMBI	0485	<-	0466	046D	0475	047C								
CMBN	0482	<-	0465	0468	046C	046F	0474	0477	047B	047E				
CMBP	0453													
CMBR	0471	<-	0461											
CMD	0961	<-	090D											
CMD0	0963	<-	0968											
CMD1	0965	<-	0967											
CMD2	0776	<-	077F											
CMDM	0775													
CMID	077C													
CMIP	0785													
CMIS	077B													
CMNI	0668													

CMP	067C												
CMP1	0781	<-	0787										
CMPD	15B1	<-	159D	15A9									
CMPE	15B5												
CMPL	0BEB	<-	0BEE										
CMPM	0780												
CMPS	10A1	<-	100C										
CMPT	10A0	<-	1016										
CMS	0774												
CMS0	10A9	<-	10F6										
CMS1	10AA	<-	10A5										
CMS2	10AF	<-	10BA	10BC	10BD								
CMS3	10B0	<-	10B8										
CMS4	10B9	<-	10B6										
CMS5	10BB	<-	10B7										
CMS6	10BE	<-	10BB	10C7	10D3								
CMS7	10C0	<-	10B5										
CMS8	10C8	<-	10C2										
CMS9	10C3	<-	10C9	10CA	10FE								
CNT0	0DBA	<-	0D9D	0DB4									
CNT1	0DBB	<-	0DB6	0DB8									
CNXT	1064	<-	1056										
COB2	1015												
COBL	1000												
COBT	1007												
CONI	1078	<-	106A	106D	106E								
CONX	0DB0	<-	0DA7	0DAB	0DAC								
COUT	124F	<-	1205	1206	1222	1225	122B	1233	1236	1244	1249	124A	124D

COVP	123F	<-	122B						
CPAB	00C9	<-	027E						
CPCS	00C8	<-	028E						
CPEX	09A6	<-	0918	0951	09AE	09C9	0A0A	0AC6	0F0E
CPRB	05E3								
CPRI	05E9								
CPT0	00C7	<-	0274	02CC					
CPX1	09AE	<-	0971	09AE					
CPX2	0971	<-	096E	096E					
CRC1	09EE	<-	09EE						
CRC2	09EF	<-	09EA						
CRC4	09F9	<-	09E8	09F6					
CRCC	09E7	<-	0911						
CRCD	0911	<-	096F						
CRCI	09E3	<-	0910						
CRCJ	0910	<-	093E						
CRCS	0AF0	<-	0A91						
CRT1	024C	<-	02AA						
CRT2	0262	<-	0257	02AA					
CRTW	0254	<-	0254	0257					
CRUW	0258	<-	0256						
CSEG	1067	<-	1055	1055	1062	1062			
CSF1	0464	<-	0469						
CSF2	0467	<-	0463						
CSL	0886								
CSN2	009D	<-	009B						
CSNC	0095	<-	0086	0D42					
CSR	088A								

SYMBOL TABLE CROSS REFERENCE

SYMBOL	CSADDR																			
CSR1	0473	<-	0478																	
CSR2	0476	<-	0472																	
CSRQ	0AC7	<-	0279	0AFE																
CSTH	00CE	<-	0046																	
CSTV	0044	<-	010E	011E	0121	0125	0126	012C	0148	0161	0181	0183	0183	1069	1071	1075	1077	107C	107E	1084
CTAB	0D05																			
CTLV	00C6	<-	0148	107D																
CTRP	124E	<-	1230	1230	123B	123B														
CTSO	1234	<-	1215																	
CTSS	1223	<-	1213																	
CUNS	124A	<-	1217																	
CVAD	1416	<-	1401																	
CVBD	15B8	<-	1403																	
CVDA	1466	<-	1402																	
CVDB	1600	<-	1404																	
CVND	1203	<-	1019	101A	101B	101C														
CVZO	15E4	<-	15CC																	
CYCL	0CE9	<-	0D14																	
D03G	04F3	<-	0400	0408	0442	0444	044F	0451	04DD	04F1	0542	0546	057F	0583						
D03L	04F6	<-	04F2																	
D03S	04F1	<-	0481	0484	0488	04C2	04C4	04ED	04EE											
D071	05B0	<-	0599	05AD																
D07G	05B1																			
D07L	05B6	<-	05B0																	
D07S	05AF	<-	054D	0558	0561	0568	0594	05AA												
D100	14FA	<-	149D	14A0	14A5	14B2	14B5	14B8	14C8	14CC	14CF	14D2	14E1	14E5	14E8	14EB	14EE	15E6	15EF	
DA1	147A	<-	1476	1480	1482															
DA15	1496	<-	147C																	

DA17	1497	<-	1466									
DA2	147C	<-	1478									
DA3	1480											
DA4	1483	<-	147D									
DA5	148B	<-	1488									
DA6	148D	<-	1485	1486	1489	148B						
DA7	148F	<-	148D									
DAB1	03C6	<-	03BC									
DABC	03C8	<-	03C2	03C4	03CA							
DABI	03C9											
DABZ	03C4											
DADS	067F											
DASL	088E											
DASR	0893											
DB1	161C	<-	1617									
DB2	1625	<-	1619									
DB2B	1635	<-	1655									
DB4	1640	<-	161A									
DB4B	1656	<-	163B									
DB6	1675	<-	161B									
DB8	16B3	<-	161B									
DBBC	0489	<-	040A	045F								
DBCC	0493	<-	040B	0455	045E							
DBER	16D5	<-	1605	17D2								
DBNE	16B7	<-	16AF									
DBNZ	134D	<-	1319									
DBZR	16BF	<-	1604									
DCER	0A5D	<-	0933	096A	0A10	0A96	0A97	0A9F	0AA3	0AA6	0AAF	0AB5

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

```
DCH     0A92  <-  0A5E  0A7D
DCJP    0AB0  <-  0A7A  0A8E  0A96
DCMD    07C4  <-  0325  0600
DCMP    0695
DCNB    0A9E  <-  0A54  0A5A
DCPU    0AA1  <-  0A53  0A59  0A74  0A79  0A8D
DCSL    08A2
DCSR    08A7
DCU0    07CA
DCU1    07D4  <-  07CB
DCU2    07D6  <-  07CC
DCU3    07DA  <-  07CD
DCU4    07DD  <-  07CE
DCU6    07E0  <-  07CF  07D0
DCU7    07E9  <-  07D1
DCU8    07EC  <-  07D2
DCU9    07F1  <-  07D3
DDD     166A  <-  164B  1680  168C  1693
DDE     08C0  <-  0D10
DDEL    08BA
DDIV    06AA  <-  069B
DDUP    1303
DEA     071A  <-  070B
DEC     140C
DECA    06BB
DECB    06BD
DECT    1400
DECX    06B9
```

DEL	08BB	<-	03EB	03EE	03F9	03FA	03FB	066F
DELB	08B9							
DFLC	1346	<-	1318					
DFLT	0800							
DFLZ	080B	<-	0801	0855				
DIE	07C1							
DIM2	0092	<-	0089	0310				
DIMS	0084	<-	0091	0174	136C	1395		
DISP	135F	<-	1357					
DIV	0677							
DIVI	0661							
DIVL	0687	<-	06C8					
DLSL	0898							
DLSR	089D							
DMA	0A11	<-	0A62	0A82				
DMAB	0933	<-	0A44	0A48	0A4C	0A4E		
DMAM	0A29	<-	0938					
DMCK	092B	<-	0929					
DMD	073B							
DMD1	073C	<-	0742					
DMDV	0699	<-	0F21					
DMID	073F							
DMIS	073E							
DMNX	06A7	<-	06A9					
DMON	0A01	<-	09FF	0A17	0A19	0A23	0A55	
DMP1	0267	<-	0264	02D3	02FD	0301		
DMPE	0004	<-	0C0A					
DMPX	00F8	<-	0005	0C09				

DMPY	16C2	<-	1400												
DMR1	0A31	<-	0A28												
DMR2	0A39	<-	0A34												
DMRR	0A28	<-	0A30	0A30											
DMS	073A														
DMSO	06A5	<-	06A8	06A9											
DMUL	069C														
DNEG	0693														
DNVL	09CF	<-	09D0												
DONE	0BEF	<-	0BEC												
DPF	0649														
DPF1	0650	<-	0654												
DPFS	0653	<-	064E												
DPPX	1354														
DSCP	02AB	<-	0263												
DSFT	08AC	<-	0892	0897	089C	08A1	08A6	08AB							
DSIX	059A	<-	0595	0597											
DSJB	098D	<-	097D												
DSR	0973	<-	098C												
DSRJ	098B	<-	0905												
DSTV	0047	<-	056F	0570	0575	0576	058B	058E	05A4	05A5					
DTOB	1660	<-	1622	1635	1639	1644	1646	164E	1679	167B	1684	1686	169A	169C	16A0
DTS1	00F5	<-	0686	13F0											
DTST	13F0														
DUMP	0214	<-	0F0D												
DUP	1300														
DVL1	0691	<-	068B												
DVL2	068C	<-	067A												

DXB1	03D2	<-	03CC																	
DXBC	03D4	<-	03D0	03D5																
DXBI	03D5																			
DXBZ	03D0																			
DXCH	1JCE																			
DXI1	03D7	<-	03CF																	
DY03	0E6F	<-	0E35																	
DY25	0E71	<-	0E36	0E37																
DY26	0E73	<-	0E4D																	
DZRO	03F6																			
EADD	0BAF	<-	0B4D																	
EBND	0B3C	<-	0B34	0B35	0B36	0B37														
ECM	020A	<-	0204	0227	0248	07C1														
ECMP	0BE6	<-	0B39																	
EDI2	1266																			
EDI3	131B	<-	128D	1294	12A1	12A7	12B2	12C0	12C6	12D4	12D7	12E4	12E9	12EF	12F1	12FA	1312	132F	1330	1331
			133B	133C	133E	1345	134A	134C	1353											
EDI4	1267	<-	131C																	
EDI5	130F	<-	1277																	
EDI6	131A	<-	12D3	12F7	1352															
EDIT	1256	<-	100B																	
EDIV	0B76	<-	0B53																	
EDLP	0B9D	<-	0B9A																	
EDT1	1270																			
EDT2	1315																			
EDVL	0B92	<-	0B8E																	
EDVZ	0BA5	<-	0B76																	
EE13	155B	<-	1513																	
EFC3	0B4E	<-	0BB8																	

EFCH	0BF1	<-	0BE8	0BED						
EFOV	0BD6	<-	0BD4	0BD5						
EMPY	0B54									
ENDF	132D	<-	1315							
ENDP	1057	<-	100E							
ENEG	0BDB	<-	0B38							
ENP1	0C93	<-	0C90							
ENPF	0C92	<-	0D13							
ENRM	0BC7	<-	0B75	0BA4						
ER13	17CE	<-	155B	1577	16E0	170B	173B			
ER15	17CF	<-	17B0	17B4						
ER17	17D3	<-	1774	1774						
ERON	0D80	<-	0D1D							
ERXT	0D92	<-	0D1E							
ESUB	0BAE	<-	0B4D							
ETBL	0B34									
ETO2	00DB	<-	04A7	04D1	04E2					
ETO3	00DF	<-	0317	0400	0408	0453	045C	053E	057C	140E
ETRP	1325	<-	129D	12B8	1336	1336				
EVAN	0BFA	<-	0B8C	0B8D	0B8D					
EXAM	0934	<-	092A	092F						
EXF	063F									
EXFP	0B2F									
EXFS	0646	<-	0642							
EXI6	0178	<-	016E	016F	01AD	01AE				
EXI7	0185	<-	017B	017D						
EXI8	0186	<-	0184							
EXI9	0170	<-	0191	01AF						

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

EXIT	0163											
EXM1	0938	<-	0942									
EXSU	0176	<-	0166									
EXT	0DC4	<-	0DBE									
EXUV	0833	<-	0820									
EZRO	0BF7	<-	0B54	0B54	0B76	0BA5	0BB0	0BC6				
FADD	081C	<-	083A									
FADJ	0824	<-	0835									
FCH0	0950	<-	093D									
FCH1	094F	<-	0900	093F	094E	0959	095F	0992	09D7	09FB	09FE	
FCH2	17A6	<-	1745	17A2								
FCH4	094E	<-	0967	0A3E	0A7B							
FCHA	17A3	<-	1589									
FCHB	17A0	<-	1507	151C	158A	1610	1712	171D				
FCHL	0A7B	<-	0A3E	0A3F	0A44	0A45	0A46	0A47	0A52	0A7C	0AF1	0AFA
FCHR	0A7A	<-	0A73	0A74	0A75	0A78						
FCM0	0819	<-	081B									
FCMP	0817											
FCRT	0A7C	<-	0A79									
FDIV	0845											
FDS1	0592	<-	0579									
FDSX	0599	<-	0596	0598								
FDZR	0851	<-	0846									
FILL	11D4	<-	1183	118C								
FINI	0BE5	<-	0BDE									
FIUL	098A	<-	09E4									
FIXR	0857											
FIXT	0861											

FLSH	07A5	<-	0D0A					
FLT	080C							
FLT1	0810	<-	0815					
FLTN	0815	<-	080E					
FLTZ	0816	<-	080D					
FMPY	083B							
FNEG	0854							
FOV	0832							
FOVZ	0871	<-	0869					
FSSR	04A0	<-	049D					
FSUB	083A							
FTST	086E	<-	0860	0868				
FUNZ	0874	<-	0858	0861				
FXSL	0869	<-	085A	0863				
GDRT	0916	<-	00B5					
GOXD	0DDE	<-	0DD3					
GRP7	0F1E							
GSMC	049D	<-	048F	0490	0499	049A		
GTP1	0CE6	<-	0CE8					
GTPR	0CE4	<-	0C99	0C9B	0CA4	0CA6	0CB5	0CB7
GTPX	0CE9	<-	0CE5					
HALT	031C							
HIOP	0C28	<-	0C22					
HLTP	09A1	<-	0949	0949	094D	09BC		
HMIT	05FD	<-	0022					
IABC	03BD	<-	03BB	03BF				
IABI	03BE							
IABZ	03BB							

SYMBOL CSADDR

IAI1 03C0 <- 03CA
IC 12B9 <- 1272 1273
ICI 12C3 <- 1273 1274
ICP 12EA <- 1276
ICS 0036 <- 00A6 027F 0796
ICSH 00C3 <- 014F
IDY 0972 <- 0906
IDYB 097F <- 0975
IFC 09BD <- 09B5 0AB6 0ABA
IFC1 09C1 <- 09C1
IFCC 09B5 <- 09B2
IMD 0730
IMD1 0731 <- 0739
IMD2 0733 <- 073D
IMID 0736
IMIS 0735
IMS 0073 <- 0025
IMSM 072F
IN9D 00AC <- 00A8
INCA 06BA
INCB 06BC
INCX 06B8
INIA 0C56 <- 0C57
INIR 0C63 <- 0C63
INIT 0C50 <- 0C1D
INL2 0995 <- 0994
INT1 0901 <- 09BC 09FB
INT8 002B <- 0040 005C 005E 0060 0063 07EB 07F0 07F2

INT9	009E	<-	0054	007B	009D	06E3	07A4	0D4D	135D	1361
INTH	0993	<-	0901							
INVM	00CC	<-	0B28							
IOA1	07B4	<-	07B9							
IOA2	07B7	<-	07B7							
IOA3	07B8	<-	07B5							
IOAF	07B1	<-	07AE	07C3						
IOBS	0C00									
IOEE	00D4	<-	00F2	0B2E						
IOER	00F1	<-	09D6							
IOLT	0C0D	<-	0F00							
IOMS	0B20	<-	0080	0090	00AE	00B4	00BD	00BE	0247	024A
			026D	026F	027C	027D	0290	0291	0296	0299
			079E	07B6	08DB	08F1	0C31	0C40	0C4C	0C54
			0F0A							
IOR	09CB	<-	0923	092B	0930	0931	0958	095A	097F	0980
			09D9	09E3	09E7	09F1	09F2	09F4	09F6	0A1E
			0AE0	0AED	0AF3	0F11				
IOW	09CA	<-	090C	0939	093A	0944	0946	094C	095E	0986
			0987	0988	0989	098A	098F	0991	0999	099C
			09C8	09EB	09ED	09EF	09F0	0A00	0A05	0A06
			0A71	0A72	0A89	0A8F	0A90	0AB7	0AB8	0AB9
IR	0010									
IR1	0032	<-	0056	005E	0060	00AA	00AB	0797	0C91	1097
IRD	0012									
IRDN	001F	<-	030B	0445	0452	0487	04C7	04D8	04E9	051F
			10BF	1133	1324	13FE				
IRH	0021	<-	0010	0012	0020	0323				
IRO	0064	<-	0024	0660	0675	068F	0871	0872	0872	
IRX	00AE	<-	00A9							
IS	12F0	<-	1277							
ISR1	0C8C	<-	0C8E							
ISTR	0C89	<-	0C92	0D12						

INT9										
INTH										
INVM										
IOA1										
IOA2										
IOA3										
IOAF										
IOBS										
IOEE										
IOER										
IOLT										
IOMS	0247	024A	024D	024F	0250	0251	0253	0255	0258	025B	025E	0260
	0296	0299	029B	029E	02A0	02A1	02A3	02A4	02A5	02A9	02E8	0314
	0C4C	0C54	0C59	0C5C	0C5F	0C61	0C62	0C64	0C65	0C66	0C68	0F09
IOR	0980	098D	099A	099F	09AB	09B0	09B6	09B8	09BE	09C1	09C5	
	09F6	0A1E	0A31	0A32	0A3A	0A66	0AC8	0AD2	0AD4	0AD8	0AD9	0ADC
IOW	0946	094C	095E	0986	096D	0970	0976	0978	0979	097A	0982	0983	0985	0986
	098F	0991	0999	099C	099D	09A0	09A2	09AF	09B2	09B3	09B4	09BB	09C0	09C3
	09F0	0A00	0A05	0A06	0A07	0A09	0A18	0A21	0A22	0A29	0A3C	0A65	0A6E	0A6F
	0A90	0AB7	0AB8	0AB9	0ABB	0AC1	0AC2	0AF2	0AF5	0AF6	0AF7	0AF8	0AF9	
IR1										
IRD										
IRDN	0521	052B	055A	055C	0563	059C	08CA	08DA	08F8	0C18		
IRH										
IRO										
IRX										
IS										
ISR1										
ISTR										

ITAB	0900					
IXBC	03CD	<-	03CB	03CE		
IXBI	03CE					
IXBZ	03CB					
IXIT	0192					
IXT1	0197	<-	0195	01B9		
IXT2	0196					
IXT3	0160	<-	0196	01B7	01B8	
IXT4	01B2	<-	00A8	0196		
IXT7	019F	<-	01B1			
IXT8	01A2	<-	01BE			
IXT9	01BA	<-	01B8			
JDCU	07FF	<-	07FB			
L1	0E4D	<-	0E47			
L2PF	0303	<-	079C	07B4	08D5	0F07
LAD	037D					
LADS	06C0					
LAI5	01F1	<-	01EE			
LAID	0380					
LAIP	01ED					
LAIS	037F					
LAP	01EA					
LAP1	01EC	<-	01F0			
LAP5	01F3	<-	01EB			
LAQ	037A					
LAS	0379					
LB32	0368	<-	035D	035D	0364	
LBD	0361					

LBD1	0365	<-	0363	
LBDL	0366			
LBDR	0367			
LBI1	0373	<-	0371	
LBID	036C			
LBIS	036B			
LBL1	0DA5	<-	0DA1	
LBPH	03E2	<-	035B	0361
LBQ	035B			
LBS	035A			
LC2	0287	<-	0285	
LCK1	02DA	<-	02DF	
LCMP	06BE			
LD	0288	<-	0286	
LD2	0285	<-	0279	
LDD	0349			
LDD1	034A	<-	0355	
LDDW	1030	<-	101D	
LDE1	0721	<-	071E	
LDEA	071B			
LDID	0352			
LDIS	0351			
LDIV	06C8			
LDNE	028D	<-	028B	
LDP	0356			
LDP1	0357			
LDPH	03DE	<-	0349	0352
LDS	0348			

LDV1	06CB	<-	06CE				
LDV2	06CE	<-	06C9				
LDW	1020	<-	101D				
LDW1	1046	<-	1023				
LDW2	1049	<-	1033				
LDWC	1043	<-	1025	1025	1035	1035	1807
LDXA	03EF						
LDXB	03ED						
LEN	1798	<-	1506	151B	1588	15C8	
LI32	0376	<-	0372	0372			
LLB3	0159	<-	0156				
LLBL	0151						
LLNI	0655						
LLS1	13F8	<-	13FC				
LLS2	13FD	<-	13FA				
LLS3	13FE	<-	13FB				
LLSH	13F2						
LMPY	06C3						
LOAD	0200	<-	0003				
LOOP	09CE	<-	09CF				
LPSH	03DB	<-	0329	032E	036C	0380	
LSEA	070D						
LSL	087E						
LSR	0882						
LSSD	070A						
LST	0CCD						
LST0	0CD8	<-	0CD3				
LST5	0CD4	<-	0CD1				

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

LSTA	0986	<-	09E6				
LSTL	0C72	<-	0C6C				
LSTX	0C6A						
LTCE	00D6	<-	02E0				
LTCK	02D8						
LTOP	0302	<-	0246	0C1A	0C3D	0C43	
LUP1	0E3F	<-	0E4C				
LUP2	0E60	<-	0E58				
LUP3	0E6B	<-	0E61				
LWD	0329						
LWD1	032A	<-	0331				
LWID	032E						
LWIP	0335						
LWIS	032D						
LWP	0332						
LWP1	0333	<-	0337				
LWP2	0334	<-	0CD7				
LWS	0328						
LXD	0339						
LXD1	033A	<-	0341				
LXID	033E						
LXIP	0345						
LXIS	033D						
LXNI	0657						
LXP	0342						
LXP1	0343	<-	0347				
LXS	0338						
MA	1291	<-	1270				

MABF	055B	<-	0550	0594	05AA			
MABR	0562	<-	0554					
MABS	0549	<-	053C					
MAF	054F							
MAOE	04C5	<-	04B6					
MAP1	02E3	<-	02E8					
MAP2	02E7	<-	02E4					
MAR	0553	<-	054E					
MASK	178E	<-	1504	151A	1587	15C0	1704	
MAX	0558	<-	0552	0556				
MBAF	0439							
MBAR	0446	<-	0438					
MBBF	041A	<-	0413					
MBBR	042E	<-	0427					
MBLA	053B							
MBLR	0545	<-	0540					
MBSF	040F							
MBSR	0423	<-	040E					
MBWA	04B6	<-	04AB					
MBWI	04C6	<-	04AE	04BB				
MBWS	04AC							
MBWX	04C0	<-	00E6	04B0	04B4	04BA	04BE	
MC	128A	<-	1270					
MCM1	08CD	<-	08CB					
MCM2	08CF	<-	08CC	08CD				
MCMD	08C4	<-	0D09					
MDOJ	0538	<-	0505	050F	051C	0526	0530	
MDS	056B							

MDSR	0597	<-	0593				
MDWO	1332	<-	1317				
MFDS	0588						
MFSU	0586						
MLBR	0582	<-	057D				
MLBS	0569						
MLOG	0788	<-	0D0B				
MMAP	02E1						
MMD1	0763	<-	0769				
MMDN	0EA3	<-	0E89				
MMDQ	0762						
MMID	0766						
MMIP	0771						
MMIS	0765						
MMP	076A						
MMP1	076B	<-	0773				
MMP2	076C	<-	0764				
MMTM	0E99	<-	0E95				
MMXT	0EA5	<-	0E77	0E7B	0E7C		
MN	129E	<-	1271	1272			
MN1	12B3	<-	12A1				
MPAA	1757	<-	1749	174B	174D	174F	
MPMS	0761						
MPY	0670						
MPY1	0674	<-	076F				
MPYD	1700	<-	140B				
MPYI	0659						
MPYL	0681						

MPYM	1746	<-	172A	172C	172E	1730	
MPYR	1727						
MPYS	1581	<-	1533	1534	1545	1561	157D
MPYX	1766	<-	1727	1728	1729		
MRTO	00D3	<-	09D0	0B25			
MSK	0E66	<-	0E67				
MSKB	1794	<-	1603				
MSTA	0E77	<-	0D22				
MSTO	00CB	<-	09CE	0B23			
MSWP	1761	<-	170E	171C			
MTA1	0619	<-	060C				
MTBA	060E						
MTBX	0621						
MTCP	02CB	<-	04FA				
MTDS	059F						
MTSW	059D						
MTX1	0626	<-	0623				
MVB1	040B	<-	0407				
MVBA	0437	<-	040D				
MVBD	0408						
MVBL	053E						
MVBP	0400						
MVBS	040E						
MVBW	04A1						
MVLB	057C	<-	056A				
MW0	0515						
MW1	0517						
MWAE	04B8	<-	04BF				

MWAO	04BC	<-	04C5						
MWBF	0520	<-	0518	0542	057F				
MWBR	052A	<-	0527	0546	0583				
MWD	050F								
MWE3	0531	<-	050B	050C	0511	0513	0515	0516	
MWF	0518								
MWF1	0519	<-	050E						
MWIX	051D	<-	0519	0528	0543	0547	0580	0584	05AB
MWOL	00E6	<-	04AF	04B3	04B9	04BD			
MWP	0505								
MWR	0527	<-	0517						
MWR1	0528	<-	050E						
MWSE	04B1	<-	04AC						
MWSO	04AE	<-	04B5						
MWX	051B	<-	051A	0529	0544	0548	0581	0585	
MY29	16DE	<-	1731						
MYST	1738	<-	1735	1745					
MZRO	173D	<-	171E						
NBAF	1218	<-	1220	1226	122C	1237			
NBAS	121B	<-	1228	1231	123C				
NCHL	00CF	<-	00AF	0C1B	0C45				
NCON	00D2	<-	02A7						
NCYC	0CEF	<-	0D15						
NEG	067B								
NEGB	16CD	<-	15CF	15ED					
NEGD	1801	<-	101F						
NEGL	0BDE	<-	0BE2						
NEWC	07C3	<-	07AC						

SYMBOL CSADDR SYMBOL TABLE CROSS REFERENCE

NEXO	0C33	<-	0C38	0C3F	0C41	0C4B	0C4F	0C5B	0C5E	0C69									
NEX1	0D49	<-	0790	08BD	08BF	08C3	08D2	08E4	08EC	08F6	08FC	0C8F	0CF0	0D07	0D45	0D48			
NEXS	0C37	<-	0C24	0C27	0C28	0C29													
NEXT	0007	<-	008F	0092	0110	0141	015B	01C0	01C2	01CB	01EC	0318	0344	037E	03BD	03C3	03C8	03CD	03D4	03DA
			03E6	03E9	03EA	03EC	03ED	03F0	03F1	03F2	03F3	03F5	03F8	03FC	04F5	04F8	053A	05B5	05B8	05BD
			05C3	05C8	05D2	05D4	05D6	05D9	05DB	05DD	05DF	05E2	05F0	05F2	05F3	05F7	060D	0618	061F	0656
			0657	0658	0660	0669	067B	067D	06B8	06B9	06BA	06BB	06BC	06BD	06BF	0747	07FF	0816	08B9	08BA
			08BB	0BED	0BEF	0C36	1058	1305	135B	136A	136B	137A	137D	13CB	13C8	13F1	13F9	1455	1493	16C1
			16C9	16CB	16CC	17D9	17DB													
NMSK	0E6A	<-	0E64																	
NOP	0006	<-	0083	0094	01D5	0619	0620	0628	066B	066C	066D	066E	0D49	1374						
NORM	082B	<-	0844	0850																
NOSX	00D1	<-	0079																	
NOT	03FC																			
NRM	0061	<-	0307																	
NRNB	0948	<-	0940																	
NRUN	0940	<-	0935																	
NSDV	04F9	<-	0263																	
NSL1	1506	<-	1512	1513	1515															
NSLD	1511	<-	1505																	
NSPG	1283	<-	1267	128B	1292	129F	12BA	12BC	12C4	12C7	12CE	12D6	12F3	12FB	133D	133F	1346	1347	134B	134D
NSRC	1306	<-	128E	1295	12A2	12A8	12B3	1332												
NSYC	00CD	<-	029D																	
NTRG	130A	<-	128F	129B	12A5	12AB	12B0	12B6	12C1	12CB	12EE	12F5	12FD	132E	133A					
OBSO	0AEC	<-	0ADB																	
OBSR	0ADD	<-	0AEF																	
OLD1	07AE	<-	07B0																	
OLDC	07AD																			
OR	03F9																			
ORI	066C																			

OVFL	13AA	<-	138C						
OVPN	11DE	<-	1190	1194	1242	1247	1339		
PARC	104D	<-	100D						
PAUL	0912	<-	0AD6	0AD7	0ADF	0AEB	0F13		
PAUS	031D								
PCAL	0100								
PCL0	0113	<-	0101						
PCL1	0101	<-	01F9						
PCL3	0117	<-	0035	00C0	010F	0284			
PCL6	0126	<-	011F						
PCL7	012C	<-	012A						
PCL8	0147	<-	0132	0133	0188	0189			
PCLP	012B	<-	011A	0125	0129				
PCLX	012A	<-	011C						
PCN	01BF	<-	0193						
PCSO	0111	<-	010A						
PFW	005A	<-	0028	05FF					
PFW0	0798	<-	005B						
PFWM	0026	<-	0073						
PHDB	13C5	<-	13BB						
PLDA	0701								
PLSA	0700								
PON	0791	<-	0001						
PONE	0001								
PSDB	1359								
PSEB	135A								
PSEH	00CA	<-	135A						
PSH1	13BE	<-	13B3	13B4	13B6	13B7	13B8	13B9	13BC

SYMBOL TABLE CROSS REFERENCE

SYMBOL	CSADDR																		
PSHA	00E3	<-	0004	031E	0788	07A5													
PSHF	13CA	<-	13B9																
PSHL	003F	<-	0043	0046	0047	0049	004A												
PSHM	0018	<-	0078	008B	0152	01C1	01C6	01EB	01EE	01F5	030E	0332	0335	037A	037D	03E7	03EF	03F4	03F6	054B
			054C	056B	056D	0588	0589	058B	059F	05A0	05A2	0655	0701	070E	080C	08C6	0C0E	0C6D	0CD6	0D44
			0F00	1099	1300	1303													
PSHR	13B2																		
PSM2	001B	<-	0356	03F6	071B	0C93	1303												
PSTA	0706	<-	0700																
PSTK	0107	<-	0116																
PSTO	13C2	<-	13C1	13C9															
PTOS	0105	<-	0114																
PUL2	000B	<-	030C	053F	054A	0699	069A	0725	07F7	08C4	0BE6	0C3C	0D93	104D	1057	105B	113A	125B	1362	13F4
			13F5	140F	1801															
PULM	0008	<-	030D	053E	053F	0549	054A	054B	054C	056B	056C	0589	058A	05A0	05A1	0899	089A	0706	0714	0725
			07F7	08BC	08BE	08C0	08C4	08C5	0BDB	0BE6	0C11	0C13	0C3C	0C42	0C72	0C84	0CCE	0CE2	0CEB	0D46
			0D95	0E21	1020	1030	104F	1057	105B	113A	1362	13F4	13F5	140E	140F	1416	1500	1516	1583	15B8
			16C2	1700	1801															
QAS1	08B0	<-	08B6																
QAS2	08B7	<-	08AE	08B4															
QASL	08AD																		
QASR	08B4																		
RBKD	13B0	<-	139A	139B															
RBYA	0A6C	<-	0A6A																
RBYB	0A66	<-	0A63																
RCCR	0D44	<-	0D05																
RCLK	03E7																		
RCMD	0C42	<-	0C08																
RDBE	0A76	<-	0A70																
RDCU	07F7	<-	0D11																
RDE1	0A73	<-	0A70																

RDEN	0A70	<-	0A4B	0A4C	0A4D						
RDT	0A49	<-	0A39								
RDT1	0A4D	<-	0A49								
RDT3	0A53	<-	0A50								
RDTA	0A52	<-	0A57								
RDTB	0A4B	<-	0A57								
READ	0A5E	<-	0903	0907							
REMO	11EF	<-	122F	123A							
RESQ	13AC	<-	13A7	13A8							
RESX	13AD	<-	13A4								
RESZ	13AE	<-	13A0								
RIO4	0C4C	<-	0C4B	0C4E							
RMIT	0324	<-	0024								
RMSK	0316	<-	03E8								
RMW	095A	<-	090A	0947	09A5						
RRE1	0A00	<-	0A62								
RREC	09FF	<-	0A16								
RRG	0958	<-	090B								
RSDL	13AF	<-	139D								
RSRT	0082	<-	00F4	0602	06E0	07D5	07E8	07F2	07F6		
RSW	0F00	<-	13F3								
RTCL	0A56	<-	09BA								
RWIO	0C44	<-	0C3E	0C49							
RWIX	0C4A	<-	0C46								
S024	17D8	<-	1536	1549	15A1	15B2	15B4	15B7	173C	1775	1776
SB32	03AA	<-	039C	039C	03A5						
SBCK	04EF	<-	04D7	04E7							
SBD	03A1										

SBDL	03A8	<-	03AB					
SBDR	03A6	<-	03A4					
SBF	08BC	<-	0D0E					
SBFD	02F9							
SBI1	03B4	<-	03B2					
SBID	03AD							
SBIS	03AC							
SBL	08BE	<-	0D0F					
SBQ	0399							
SBS	0398							
SCAL	01C3							
SCAN	0629							
SCLK	03E5							
SCLR	0D46	<-	0D06					
SCN0	062C	<-	0629					
SCN1	0631	<-	062B					
SCNA	17EE	<-	1708					
SCNB	17DC	<-	1511	1707				
SCND	17E5	<-	17E1	17E1	17F3	17F3		
SCNE	17EC	<-	17E6	17E6	17E8	17E8	17EA	17EA
SCNX	062D							
SCNZ	01C6	<-	01C4					
SCU	04DF	<-	0587					
SCU0	04E7	<-	04EC					
SCU1	04EA	<-	04E6					
SCU2	04ED	<-	04E8	04EB				
SCU3	04EE	<-	04E8	04EB				
SCW	04CE	<-	059E					

SCW0 04D6 <- 04DA
SCW1 04D9 <- 04D5
SCW2 04DB <- 04D7 04DA
SCX0 0632 <- 062D
SD13 1491 <- 1466 149A
SD23 16BD <- 1624 163F 165B 165F 16AE 16B2 16D8
SD24 1453 <- 1418 1418 15BC 15BD 15DE 16DD
SDD 038E
SDD1 038F <- 0397
SDE1 072B <- 0728
SDEA 0725 <- 071A
SDID 0394
SDIS 0393
SDS 038D
SED 1368
SED1 136B <- 1368
SED2 136C <- 1369
SETC 134B <- 1319
SETQ 13A6 <- 138A
SETR 137E
SETX 13A4 <- 1389
SETZ 139D <- 1387
SFC 133D <- 1317
SFLC 133F <- 1318
SHIO 0C1A <- 0C00 0C01 0C06
SHLT 00D8 <- 00C1 00C2 00C3 00C4 00C5 00C6 00C7 00C8 00C9 00CA 00CB 00CC 00CD 00CE 00CF 00D0 00D1 00D2
00D3 00D4 00D5 00D6 00D7
SI32 03B8 <- 03B3 03B3
SINC 0D4A <- 0D0D

SIO1	0C1E	<-	0C3B					
SIOE	0C2E	<-	0C26	0C26	0C2B	0C2C		
SIOP	0C23							
SL3D	1562	<-	153A					
SL3J	1541	<-	1565					
SL5D	1510	<-	153B					
SLD	1500	<-	1405	1406				
SLD1	154A	<-	1523	153A	157F	15DF	1615	
SLD2	155C	<-	1524	1538	158C	1737		
SLD3	1537	<-	150D					
SLD4	153E							
SLD5	153A	<-	1537					
SLD7	153C	<-	1538	1539				
SLDN	1544	<-	150F	153F				
SLST	1546	<-	1510					
SMD	0754							
SMD1	0755	<-	075A					
SMEM	01CD	<-	01C5					
SMID	0757							
SMIP	075E							
SMIS	0756							
SMK1	030F	<-	0315					
SMP	075B							
SMP1	075C	<-	0760					
SMS	0753							
SMSK	0308	<-	03E5					
SPGS	1278	<-	1266	12F8	131A			
SPLT	1788	<-	1503	1519	1586	15BF	160F	1703

SR3D	157E	<-	152A	1715	1721							
SR5D	1526	<-	152B									
SRD	1516	<-	1407									
SRD1	1566	<-	150E	152A	1563	15E2	1614	1714	1720	172B	172D	172F
SRD2	1578	<-	150E	1528	158D							
SRD3	1527	<-	1522									
SRD4	152D	<-	1580									
SRD5	152A	<-	1527									
SRD7	152C	<-	1528	1529								
SRDN	1532	<-	1525	152E								
SRDX	16B4	<-	150C									
SRPA	0014											
SRPU	0013	<-	0014	0016								
SRST	1535	<-	1526									
SSE1	0716	<-	0714									
SSEA	0713	<-	070C									
SST	0CD9											
SST0	1331	<-	1316									
SST1	1330	<-	1316	133C								
SST8	0CE2	<-	0CDE									
SST9	0CE3	<-	0CE1									
SSTA	13A0	<-	1388									
SSTL	0C82	<-	0C7C									
SSTX	0C7A											
STAX	03EE											
STBX	03EA											
STDB	1398	<-	1385									
STDL	139B	<-	1386									

STNU	00F9	<-	0013														
STNX	01D6	<-	00FA														
STO1	004B	<-	0019	001D	0F18												
STO2	0051	<-	00DE	00E2	0111	0112	0167	01D1	01F5	03DC	03E0	03E3	0536	1047	104B		
STOA	17B6	<-	1535	1547	159F	15DC	173A										
STOE	17BC	<-	17B7	17B9													
STOG	17BA	<-	17B8														
STOH	00C5	<-	0053														
STOP	031B	<-	0327	0601	0796	07F5											
STOS	01CC	<-	01C5														
STOV	0052	<-	004D	004F	01A3	01A4	1371	1379	137C	1381	13B1	13C4					
STPO	11E3	<-	1156	1170	123E												
STPS	11FB	<-	122A														
STRT	0943	<-	0936														
STSB	1396	<-	1383														
STSM	13A8	<-	138B														
STSR	0029	<-	0059	0071													
STTH	00C2	<-	0043														
STTV	0041	<-	010E	013A	0154	0157	015D	01C9									
STU1	0048	<-	0009	0F1B													
STUN	0049	<-	000F	0115	0168	0177	01CD	01D3	01DB	01DD	04F7	05B7	1379	137B	137C	1396	13B1
SUBM	15A2	<-	1591														
SUFS	12E5	<-	12D9														
SUFT	12D5	<-	1275														
SUSB	0ABE	<-	00F2	097E													
SUSC	0ABD	<-	09DB														
SUSP	09D8	<-	0962	096F	0973	0984	0994	09E5	09E8	09EE	0A93						
SWD	0385																

SWD1	0386	<-	038C			
SWID	0389					
SWIS	0388					
SWS	0384					
SWUB	04DE	<-	04D4	04E5		
SXIT	01CF					
TABE	014C	<-	018E	018F		
TABP	0149	<-	0137			
TABS	014D	<-	014B			
TABT	0150	<-	014D			
TAS1	13E0	<-	13DB	13E2		
TAS2	13DE	<-	13E4			
TASL	13DA					
TASR	13E2					
TBA	0605					
TBC	0633					
TBC1	0634	<-	0638	063B	063D	
TBFL	0E75	<-	0E6D			
TBX	061A					
TC1	06D2	<-	06D7			
TC2	06D8	<-	06D4			
TCBC	063D					
TCCS	1099	<-	1017	1018		
TCLK	06CF	<-	0027			
TCSQ	0AFB	<-	0079			
TDSX	05AD	<-	05AC			
TE	1326	<-	1315			
TEST	13EF					

TIC0	06DE							
TIC1	06E1	<-	06DD					
TICK	06DC							
TICR	06E0	<-	06DB					
TIM1	0E17	<-	0E12					
TIME	00E8	<-	00F3	031F	0322	06CF	06DA	0AFB
TIMR	0E00	<-	0D20					
TLK1	0977	<-	09EA	09EC	0A60	0A7F		
TMRQ	0E20	<-	0D21					
TNS0	13EE	<-	13E7					
TNS1	13EC	<-	13EE					
TNSL	13E5							
TON	0D40	<-	0D08					
TR	10FF	<-	101E					
TR1	1113	<-	1123					
TR2	1114	<-	111E	1121				
TR3	111F	<-	111C					
TR4	1122	<-	111D					
TR5	1132	<-	1120	1130				
TR55	1124	<-	1112					
TR6	1126	<-	112E	1131				
TR6E	0175	<-	016C					
TR7	1134	<-	1101	1114	1122	1126		
TR8	1137	<-	1103	1103	1106	1106		
TRAP	0000	<-	0D04					
TRBC	0638							
TRCE	005F	<-	0150					
TRN1	110D	<-	1107					

SYMBOL TABLE CROSS REFERENCE

SYMBOL	CSADDR																		
TRN2	1110	<-	110C																
TRP	003E	<-	0000	01C0	0B32	0B3A	0B3B	0C02	0C03	0C04	0C05	0C07	0C08	0C17	0D16	0D17	0D18	0D19	0D1A	0D1B
			0D1C	0F20	0F20	1009	100A	1010	1015	1016	140C	140D	1500	1516	1583	15B8	1600	16C2	1700	
TRP1	0065																			
TRP2	0066	<-	0832																	
TRP3	0067	<-	0831	0832																
TRP4	0068	<-	0661	0678	0688	06AB														
TRP5	0069	<-	0853																	
TRP6	004A	<-	0175	018E	0193	031D	03E5	053B	0569	057C	0588	059F	0700	070B	0C0D	0C6A	0C7A	0CCD	0CD9	1084
			1354	1368	1382	13BB	13F3													
TRPA	006B	<-	1160	11A3	1251	132B	16DC													
TRPD	006C	<-	1499	16D7	17D6															
TRPO	006A	<-	16DA	17D5																
TRPR	006E	<-	1160	11A3	1251	1499	16D7	16DB	17D6	1814										
TRPS	006D	<-	16DC																	
TRPZ	006F																			
TSBC	063B																			
TSRT	00F3	<-	09AD																	
UAN1	0839	<-	0836																	
UANS	0837	<-	0824																	
UEXI	00C1	<-	0028																	
UNCL	0055	<-	013D																	
UNFL	13AB	<-	138C	1390	1390															
UNP1	02BD	<-	02C7																	
UNP2	02BE	<-	02C4																	
UNPK	02BC	<-	02AD	02CE	02D6	04FD	0503													
USHT	04C8	<-	04B0	04B3	04BA	04BD														
UTRP	0070	<-	0065	0066	0067	0068	0069	006A	006B	006C	006D	006E	006F	0BAD	0BDA					
V	0AB4	<-	0957	0A2E																

VLB	11A8	<-	1183	118C				
VLD	11B1	<-	11A9					
VMLB	11B8	<-	118A	1216	1221	1224	1232	1235
VMLD	11C6	<-	11B9					
VRET	11DC	<-	11AB	11B2	11BB	11C7	11D8	
VTRP	11DB	<-	11B4	11B4	11CB	11CB		
VZRO	0BAB	<-	0BF9					
WAIT	0969	<-	0902					
WBC1	04CA	<-	04B1	04B8				
WBC4	04CC	<-	04B4	04BE				
WBEN	0A88	<-	0A85					
WBNE	0A89	<-	0A87					
WBX1	0A1E	<-	0A1A	0A24	0A25	0A27		
WBXD	0A1A	<-	0A14					
WCCE	00D5	<-	02D7					
WCMD	0C3C	<-	0C0C					
WDIE	07C2	<-	07C2					
WDT	0A58	<-	0A39					
WDTB	0A8C	<-	0A58					
WFMS	0B24	<-	0B25					
WIO4	0C40	<-	0C3F					
WIOC	0C3F	<-	0C4A					
WMS1	0F07	<-	0F0C					
WMST	0F05	<-	00F8					
WRIM	09FC	<-	090F					
WRIT	0A7D	<-	0904	0908				
WRX1	0A24	<-	0A1B					
WRXD	0A25	<-	0A5C	0A80				

WRXE	0A26	<-	0A24				
WUN	0A8E	<-	0A8C				
WUNL	0A8F	<-	0A59	0A8D			
X1W2	1672	<-	1638	169F			
X1WD	1671	<-	1645	167A	1685	169B	
X2WD	166D	<-	164F				
XAX	13D4						
XBR	105B	<-	100D				
XBX	13D8						
XCH	13D2						
XCH1	13D3	<-	13D7	13EF			
XCHD	1362	<-	1355				
XDCU	07FC	<-	07F9				
XDIE	0DE6	<-	0DCD	0DD1	0DE3	0DE4	
XDMA	0A0B	<-	090E				
XEQ	136D						
XFIN	0DD5	<-	0DE5				
XGDB	0DCC	<-	0D1F				
XOR	03FA						
XORI	066D						
XT11	0A46	<-	0A48				
XTIN	0A40	<-	0A3D				
XTIR	0A47	<-	0A43				
XXX	0E21	<-	0E22				
ZER1	03F5	<-	03F7				
ZERO	03F4						
ZREG	1741	<-	17A5				
ZROB	03F8						

ZROW	0836	<-	082A	083E	0848
ZROX	03F3				

ADDRESS				
0000:	38E5 7510 C3E3 E40C	38E8 C904 0C03 DE46	38E5 7194 DDF8 4B53	38E5 7104 CCC8 C802
0004:	38E5 7135 D1F6 E78C	38E5 7510 CF83 E5CD	38E5 7110 CCF3 E40C	38E5 7010 CCF3 E51D
0008:	5D72 B179 CCF5 FFFE	58EB 6710 8483 B08D	3CE5 7000 CC83 FC0C	5D72 B179 CCF5 FFFE
000C:	38E2 A110 BC73 B08C	3D65 7043 CC83 E84C	3CE5 7003 CC85 DC8C	2975 7511 C495 E49D
0010:	38E5 7510 C213 E40D	4D4A 4D20 2638 5820	3945 7579 C215 3FFF	3975 7511 CF95 E49D
0014:	58E9 2710 8133 B08C	38F8 7110 CCF2 251D	58E9 2710 8133 B08C	38F8 7110 CCF2 251D
0018:	5D74 B1E1 DCF5 E49D	24C3 7047 CCF3 F92D	1CE5 7000 CCF3 FC0C	5CE4 B110 DCF3 E49D
001C:	24A3 7111 CCF5 E49D	24C3 7047 CCF3 F92D	1CE5 7000 CCF3 FC0C	3940 A979 00F5 3FFE
0020:	38AC 3510 D213 E50D	68E0 4B10 00F3 A51D	64A5 7224 F493 D7F7	38A1 4908 0073 E241
0024:	34B9 34C4 D643 CC92	64AB B445 D733 E95E	64D0 0A28 1093 C801	28E5 7104 CCF6 DB3E
0028:	38D5 74C5 DC13 E56A	38E5 702E CCF3 F0A9	38E5 7023 CCF3 EC8D	5D74 B12E DCF5 F098
002C:	2543 7140 CCF3 E49C	3933 6112 4CF3 ECAC	3AC5 711D CCF3 EC03	38E3 6110 CCF3 E40D
0030:	6173 7109 CC35 C008	58E3 6110 ECF5 AC6C	5CE4 B149 DCF4 4002	38A3 7190 8CF5 E5BD
0034:	46E5 7110 CCB3 E50C	74E5 7104 CCC3 C45F	38E0 0910 0AF6 E40C	28E0 F139 CCFA E000
0038:	2970 F110 DCF0 A40D	38FF F910 F8F5 A40D	2CF2 E110 8CF2 A40D	296F F910 FCF4 A40D
003C:	2964 F109 CCF5 C005	3EB3 4000 ACF3 E50D	38E5 7149 CCF4 2003	38E6 7110 CCF3 E57C
0040:	40E5 7105 CCD3 E4AF	38E5 7149 CCF4 2202	62C0 1A20 FCF3 E40C	38A5 7425 D3F3 E70A
0044:	38E5 7149 CCF4 2402	62C0 1A20 FCF3 E40C	38A5 7425 D3F3 E73A	38E5 7549 D3F4 2603
0048:	3975 7111 CCF5 E49C	38E5 7549 D3F4 2802	38E5 7549 D3F4 2A03	6142 0B79 0094 3FFE
004C:	60B5 6310 80B3 E68D	2CD2 30C5 CCF3 E54A	38F0 0828 7EF3 E000	3D0B 3305 48F5 254B
0050:	60E2 0D10 00B3 E40C	3945 7179 CCF5 3FFE	68E2 0B49 00F4 3002	2CA6 70C5 CCD3 E717
0054:	38E6 B510 D9E3 E5DD	40E6 7159 CCD4 0203	38E5 7105 CCF3 E4CB	38E6 3159 CCFB 8000
0058:	38E5 7149 CCF4 0203	2CE5 7105 CCD3 E4A6	38E5 7110 CCF3 E5CD	38E1 8904 04F6 DE62
005C:	38E6 7559 D2B4 0402	0CE6 7149 CCB4 3E03	38E5 75A5 D323 E4AF	0CE6 7159 CCB4 0003
0060:	38E5 75A5 D323 E4AF	3946 7111 CCF5 270C	38E5 7149 CCF4 0602	2CE5 7105 CCD3 E4AF
0064:	60EE FB10 FEB3 E40D	38E5 7105 DCD3 E5C3	38E0 0905 04D3 E5C3	38E0 0905 06D3 E5C2
0068:	38E0 0905 08D3 E5C3	38E0 0905 0AD3 E5C2	38E0 0905 16D3 E5C3	38E0 0905 18D3 E5C2
006C:	38E0 0905 1AD3 E5C3	38E0 0905 1CD3 E5C3	38E0 0905 1ED3 E5C2	38E0 0905 20D3 E5C3
0070:	60E4 0B49 00F4 3202	28E8 B510 D293 E59C	38E5 7010 CCF3 E7FD	38A1 88C5 04F3 E499
0074:	38A0 8910 04F1 E67C	38A0 1910 2093 E67D	38FD F510 C182 E5ED	2CF0 0B08 7194 0181
0078:	2CD7 E110 CEF3 E50D	40EB 35A4 CD17 6BEF	60EA B149 C0F2 C000	38E5 7579 D9E4 7FF1
007C:	A4EB 211E CCF3 FC09	28BB 3110 4CF3 E403	BD15 6313 C5F3 E50C	38E5 7109 CCF2 E001
0080:	38EA 0904 0063 EC83	6072 0B10 00F3 E678	50A4 3020 CCF3 E40D	3945 7510 C063 E60D
0084:	38E1 8909 0474 401F	BCEA B110 C8F3 E40D	BCA9 F510 D953 E93D	3875 7110 CCF3 E67C
0088:	BCE5 7110 C8F3 E40D	28FB 2510 9923 E40D	38E5 C910 B893 E40C	38A0 F510 C183 A5E7
008C:	BCE6 F11D CCD7 6458	3880 08CF 1194 6850	28B0 08CD 1194 6054	38EB B508 C072 E000
0090:	34E5 7104 CAB3 EC83	44E5 7225 81F3 E612	38EB B51D C073 E40D	3875 7180 CCF3 E678
0094:	39B 3510 C063 E50D	BCE5 7139 C0F1 FEFE	38A0 08C8 1EF3 E001	BCE5 5310 C9F3 E90D
0098:	38E0 0910 2CF6 E40D	38E0 0910 26F3 EA3C	38E6 7110 CCD3 EA3C	3CE8 B510 D9D3 E40D
009C:	3CE5 7110 CCD3 E6DC	38E6 B549 C9E4 1802	5D74 B12E DCF5 F098	2543 7140 CCF3 E49C
00A0:	3933 6112 4CF3 ECAC	3AC5 711D CCF3 EC03	3973 6109 CCF5 C00D	6173 7109 CC35 8009
00A4:	3962 091D 0093 EF29	68C8 B312 48F4 62DD	3995 7510 D361 E6DD	695A F1D0 CEF3 E6DC
00A8:	38E5 7504 DAC6 86CB	38E7 F9B5 FEF6 A6BB	1CA8 B511 D323 E50D	38D5 751D D323 ED03
00AC:	3911 E94A 00F3 C001	2CA3 7000 CCF3 E50C	38E0 0904 A093 AC82	18EB 1510 9CF8 E43D
00B0:	18E6 F11E CCF3 E408	2897 F1A0 C0F3 E44C	2CE5 6119 C0F2 C000	18E9 3510 DD07 69E1
00B4:	38E5 7104 CCF6 6C82	3867 3104 CCF7 645B	5CB4 7110 DCF3 E40D	1CB3 7111 CCF3 E40D
00B8:	38D0 A110 DCF3 E50E	38E0 A109 BCF2 E001	74F6 3110 CCC4 240C	38E1 8910 0078 A5FD
00BC:	1860 0B0A 0E67 5801	38F5 C904 0095 6C83	34E7 7184 C067 6C83	5C75 7110 DCF3 E67C
00C0:	38A1 F104 DCB5 C45F	38E5 7510 DD84 240D	38E5 7511 DD84 240C	38E5 7511 DD84 2400
00C4:	38E5 7509 DD84 0006	38E5 7509 DD84 0009	38E5 7509 DD84 000A	38E5 7509 DD84 000C
00C8:	38E5 7509 DD84 000F	38E5 7509 DD84 0011	38E5 7509 DD84 0012	38E5 7509 DD84 0014
00CC:	38E5 7509 DD84 0017	38E5 7509 DD84 0018	38E5 7509 DD84 001B	38E5 7509 DD84 001D
00D0:	38E5 7509 DD84 001E	38E5 7509 DD84 0021	38E5 7509 DD84 0022	38E5 7509 DD84 0024
00D4:	38E5 7509 DD84 0027	38E5 7509 DD84 0028	38E5 7509 DD84 002B	38E5 7509 DD84 002D
00D8:	38E5 7109 CCF3 C0C1	38E5 7110 CCF3 ED0C	A4E5 7310 CCF3 E40D	397F F912 FAF4 79AC
00DC:	5CA4 B0DE DCF1 F098	24E3 7043 CCF3 EC9D	457A 9601 E515 DC8C	397F F912 F8F4 79AD
00E0:	5CA4 B0DE DCF1 F098	24E3 7043 CCF3 EC9D	457A 9601 E515 DC8C	5D74 B12E DCF5 F098

ADDRESS				
00E4:	24E3 7140 CCF3 E49D	38E8 3110 CCF3 E4AC	00E5 7054 DC03 D303	38E5 7105 CCF3 E55C
00E8:	28E5 7109 CC93 8061	38E5 7109 CCF3 8143	3885 7110 CCF2 E40D	3885 711D CCF2 EC0E
00EC:	38E5 7110 CCF2 3C0D	3915 7110 CCF3 A40C	38E5 7110 CCF3 E40C	2085 711C CC82 3C41
00F0:	38E8 3179 CCF3 C000	39C6 B0C8 CCF3 FE00	3865 7434 CD42 EAFB	28E5 7509 CE84 4144
00F4:	34E5 7510 C823 E40D	2818 B110 CCF3 E79C	3808 7110 CCF3 E78D	38E8 7110 CCF3 E7DC
00F8:	38E5 7104 CCF1 7C17	64E4 0B10 00F3 E40C	38A5 7024 CCF3 C75A	3965 7510 C575 E4AC
00FC:	38E5 7110 CCF3 E40C			

ADDRESS																
0100:	70E6	7110	CC93	F68D	293B	3512	D134	752C	5CF4	B110	DCF1	E407	34F7	A110	ECF2	E404
0104:	38D5	7112	CCF3	F68C	5175	712E	CCD5	F099	24B3	7140	CCF3	E49D	38B3	6109	4CF1	4008
0108:	4EC5	7110	ACF3	E40C	2C83	6310	84F4	240D	6053	7047	CC33	F846	5903	5118	6CF1	BFFF
010C:	4C54	2110	CC95	E4AC	40DE	3062	C4F3	FC0C	70EB	2505	8443	E504	4459	9795	8575	A45E
0110:	38D5	7510	C075	240C	34E8	B510	C513	E4AD	3885	7510	C510	248D	517C	B120	CCD1	E4BD
0114:	5CE4	B509	F053	800E	5979	1710	8493	E4AD	5CE0	B513	D075	EC0D	38E0	5910	2446	E5AC
0118:	7105	5110	ECD1	E405	10E2	F113	DCF3	D2BC	6105	7045	C463	E4AE	384B	1110	CC43	E6BC
011C:	3CFB	5710	D2A7	240D	18F5	50C0	9C90	E40D	2DC3	3025	CCF3	E510	3476	00C5	CCF3	E49B
0120:	3CA5	7110	C4F3	E6CC	0CA5	7067	8CF3	DD11	38F5	7110	CCF3	EACC	38E5	7110	CCF3	E59D
0124:	38F5	7110	CCF1	E405	2CFB	3445	C444	24AF	3C79	9710	C443	DC0D	388E	3110	CCF3	CECD
0128:	1CE5	7110	CC43	E40D	38F5	7500	D2B1	E40D	38E2	F105	DCF3	E4B2	3872	F110	CCF1	240D
012C:	1D0B	3548	C443	C000	2CB5	7110	C228	6403	6105	7138	C893	FE01	3CB2	F020	CCD4	65D7
0130:	3475	5311	0433	DDC0	2C70	F0C0	8C93	DC01	2CE2	F105	8CF3	E51F	3CE9	9749	D472	C009
0134:	3CF5	7308	F4F6	01FE	28D7	A111	80F3	E6BD	38A5	60CD	B263	FC04	2CF8	B510	D493	2A8D
0138:	3D14	0CF2	00F3	EE8C	28F3	7110	CCF3	E586	44F9	2718	E414	7FFF	44FB	3110	C2F3	E40C
013C:	2C6A	B1D2	C0F2	F40C	611A	B50E	C553	D807	BCE0	0B00	0695	3F0C	0D11	F110	CCB5	3F0D
0140:	3618	B110	CCFB	240D	38E5	7510	C074	FC3D	38E5	7159	CCF3	CD21	38B5	7110	CCF3	E50D
0144:	BCE0	0B09	09F3	8165	3B05	7110	CCF2	240C	38E5	7010	CCF6	640C	38E0	F1A5	8CF3	E522
0148:	38F5	7425	C443	E719	28A3	70F0	CCF1	240D	40E5	7110	CCD3	E58D	0E11	F105	CC3B	2537
014C:	70E5	7110	CCD3	E5BD	08EB	3510	D503	E40C	6832	0AC8	00F3	C1FD	28E8	B505	CC33	E711
0150:	3845	7445	C5F3	E575	60E5	7110	CA73	F68D	70E2	2175	ECF3	2460	70F5	7110	FC93	E4F4
0154:	28EB	6510	E416	E40D	60F5	7110	C4D2	E407	72CB	6710	F593	E540	70E9	F510	C413	E40C
0158:	38D8	7113	CCF1	DF6E	3CE5	7109	C473	C522	2CB2	F110	DCF3	E6BD	3C75	70C5	CC73	E41D
015C:	3CF5	7110	CCD7	240C	3875	7045	80F3	E504	28D3	3111	9CF3	FC0D	2CE5	7110	8CF3	E40D
0160:	3DC3	2020	8CF3	E40D	38EB	3505	8443	E58B	3CE5	7010	CC73	E40D	5962	B179	CCD4	3FF8
0164:	70E5	6113	EC72	EAAD	3D6F	B110	AC65	E4AD	54EB	6713	B763	FE2C	6075	70D7	CC93	F945
0168:	1CF9	971D	8492	E800	3CE5	7159	CC24	7FFE	40F2	B040	DCF0	E40C	44E5	42A0	8813	FC01
016C:	3CE5	70F5	CCC3	E5D7	4EC5	40A0	4C93	C800	3535	4107	E4F3	F5E2	04EB	66C5	E573	E9E3
0170:	3872	0910	00F5	E40D	0CD8	A310	88F5	260D	3464	3110	CCF5	A40C	6835	70A0	CCF3	E401
0174:	0CAC	3035	CCB3	E610	38E5	7410	C4A3	E40D	4955	70D2	CCF3	F80C	3958	30D7	CCF3	E925
0178:	B0E5	4310	8826	E5AD	3830	5910	24F1	6404	38E2	F111	CCF3	EABD	08E8	B513	D851	22BC
017C:	38F2	F063	DCF3	D40C	3CE5	7105	CCD3	E617	10F2	F110	DCF7	240D	14B5	5110	4CF6	E40D
0180:	3DC3	22C0	94F3	E40D	28A0	F025	4CF3	E511	3CE5	7110	C4F6	E40C	28F9	9625	4441	6511
0184:	38E5	7105	CCF3	E61B	3CE5	71A5	CCD3	E61B	3852	F111	8CF3	D401	2C50	E0C0	8C93	D400
0188:	2CE2	F105	8CF3	E51F	3CF9	9708	5476	01FF	3CD1	FB11	FEF3	E6BD	3CA7	F110	C0F3	E800
018C:	3CE4	0D13	00F1	6C0C	28F3	71B0	CC43	2C0C	30F9	34F5	D4C3	E529	2C15	4045	4C93	E532
0190:	38F5	7110	CCF4	E40C	0425	5305	5EFB	25C3	716B	3110	CCF3	E62C	68E5	74F5	DBF6	E528
0194:	58AF	F828	F673	C201	3CE5	7105	CC33	E65E	28A8	B445	DB03	E6CA	38EF	E913	00F2	DEBD
0198:	1CA2	F110	DCF3	E50D	1CE2	F113	CCF1	FEBD	3CF6	7179	CC63	FFFC	38B2	D110	FCF5	A80C
019C:	3CF5	7110	CCF4	980C	3CA5	5110	8CD6	A40C	58A2	B110	CC72	A40D	356E	2111	EC63	E6AC
01A0:	3CB5	5111	EC63	E6AD	58FF	F910	F876	64AC	2962	B110	DC95	E40D	1CF9	3710	A525	640C
01A4:	18D9	3711	A523	E62C	3CB5	7111	CCC3	E6AD	3CE5	7159	CC23	FFFE	3D35	6312	C812	F40C
01A8:	4CF5	6110	8C90	E40D	61A5	6110	E443	E40D	3997	F82D	FEF3	EC01	38A2	0901	00F3	E50D
01AC:	38B5	711D	CCF3	E903	1028	B528	D780	8001	3AC5	70C7	CCF3	EDE3	3815	751E	D703	FC06
01B0:	0CE6	0110	C293	F80C	2D65	7105	CCD3	E67F	38E0	0909	0AF3	CE00	2882	F110	CCF3	E50C
01B4:	38E6	7109	CCF4	0024	3CE2	F110	CC93	F80C	2D05	7112	CCD3	EABC	3D6A	B510	DB03	E40C
01B8:	34F8	A625	FB03	E6EB	38E5	7105	CCF3	E65F	34D2	B110	DC95	A73D	2CE0	0910	0473	E6DC
01BC:	3CD5	7111	CCF3	E6AD	58A5	7110	CC66	640C	58EF	F905	F8F3	E68A	70EF	FB10	F8F3	E40C
01C0:	70A8	F4C5	C073	E4F8	38ED	F510	C183	E40D	38E3	F509	C071	C008	713B	3112	CCF4	752C
01C4:	38A5	7522	DC63	F68D	5CE0	B505	DCD3	A733	388D	F51D	C183	F688	38C5	7118	CCF2	FFFE
01C8:	50F5	7110	CCD3	E4D7	70EB	6710	E413	FC0D	4539	A710	C573	FF0D	4CB3	F105	AC75	241C
01CC:	38EC	3000	CCF3	E4BC	3965	7047	CCF3	E925	38E8	3113	CCF5	E00D	5D74	B12E	DCF5	F098
01D0:	24E3	7140	CCF3	F89C	5D79	2710	E513	E4AC	7135	6112	FC93	F40D	5809	271E	A493	EC06
01D4:	3935	7110	CCF5	030C	38D5	7510	C065	E40D	38E0	A910	0093	E40C	5D74	B12E	DCF5	F098
01D8:	24E3	7140	CCF3	E49D	38D5	7110	CCF3	E50D	38E5	7100	CCF3	E4AD	3945	7106	CCF5	2525
01DC:	60E2	0B10	00F3	E40D	38A5	7425	C493	E76F	0401	0001	0000	4400	0314	0200	0000	0500
01E0:	0000	0000	0401	0001	0000	4400	0341	0200	0000	0500	0000	0000	0300	0100	0440	8000

ADDRESS																
01E4:	0E40	0000	0002	0000	FFF2	0407	0001	0000	4400	0342	0302	0002	0000	0400	0343	0200
01E8:	0000	0500	0000	0000	0180	0000	0008	0013	70E5	6319	A073	E401	28FD	F427	C183	EFCF
01EC:	4CE3	E105	AC73	E41D	70E2	2319	A073	E401	38FD	F427	C183	EFC7	1D35	611E	0CF3	DC04
01F0:	3C75	651E	9EC3	F405	1D35	611E	0CF3	DC04	3C75	611E	8C73	F405	5CE5	7109	0CF2	CA01
01F4:	38A0	0912	08F3	F80C	38BD	F467	C183	E945	4CE3	D110	EC73	E40C	38E5	7110	CC93	FD0D
01F8:	38E5	7110	CCF3	E40C	3946	3511	D015	270C	3985	7062	CCF3	C40D	38E5	7110	CCF8	240C
01FC:	38E5	7110	CCF3	E40C	3855	7110	CCF3	E70D	38E5	7110	CCF3	E40C	38F5	7000	CCF1	640D
0200:	38E0	1909	2099	A001	1C70	0B10	F0F4	5D80	28E5	7159	C2F9	E000	38A0	0910	0499	24A3
0204:	38D5	7510	D0A3	E50D	3A44	F110	CC84	2403	38EA	3510	D077	640C	8C63	70C6	CCF1	A41A
0208:	39B5	7022	CCF7	040D	39B6	3506	D436	E417	38EF	0909	0816	8240	38A0	8910	0063	A65C
020C:	38D0	5910	70F3	E72C	38A6	1910	F133	E67D	38EB	B110	CCF3	DA7D	38E5	7510	D0E3	DA71
0210:	38F0	1910	2013	E66D	2CF5	711F	CCF3	C400	28B5	708C	C0F3	E450	38E5	6000	BC13	C50D
0214:	38E0	6910	98F7	25DC	2905	3110	CCF3	E40C	2CE3	7109	CCF3	C602	2C75	3110	CCFC	240D
0218:	1CE3	7109	CCF9	A001	38A3	6110	4CF4	6801	4993	7110	CCF3	E40D	2CE3	7111	CCF6	640C
021C:	5473	7108	CCF3	C0F0	58B3	7110	CCF4	2402	5DA3	7110	CCF3	E40D	2CE3	7110	CCF3	F80D
0220:	2CE3	7110	CCF7	640C	6183	7110	CCF3	E40C	2D03	7110	CCF9	2403	4CB3	7110	CCF4	2402
0224:	50E3	7110	CC53	F40C	2CE3	7110	CCF3	E40C	74E3	7159	CCF9	E001	38E5	7105	CC93	E42A
0228:	28E3	7139	DC1A	2EEE	64E3	7109	CCF3	804E	68E3	7109	CCF3	C023	38E4	E110	CCF1	A40D
022C:	38E0	3109	CC83	8074	38E5	7110	CCF3	E40C	388C	3400	D303	E40D	38E5	7110	CCF3	E40C
0230:	38C0	5910	FEF3	E40C	38B2	E110	8CF3	E40C	38B0	4910	20F3	E40C	3CB4	E110	8CF3	E40D
0234:	2CE3	7109	CCF3	C69B	2CE4	F104	CCF3	C7EA	2D13	7100	CCF3	E61C	38E0	0910	1093	E6ED
0238:	3865	702B	CCF3	C007	3C63	7106	CCF1	A4DF	38D5	7110	CCF3	E63D	38B0	0911	3493	E63D
023C:	3CE3	7110	CCF3	E40D	3CD3	7100	CCF3	E63D	38E2	2910	2213	E6ED	3865	7020	DCF3	E40D
0240:	3CA3	7107	CCF1	98FA	3A85	711E	CCF3	E409	041B	2110	DCF3	ED0C	38E0	0910	10F6	E40D
0244:	2914	F110	CCF5	E51C	3873	7110	DCF1	A40D	38A3	7104	CCF6	4C0A	1E70	0AC4	FE72	EC83
0248:	3A66	30C5	CC92	E42B	38E0	0929	0ED0	6801	44B0	08C4	0491	AC82	1C70	0AC4	0F17	4A62
024C:	38E0	0929	0053	C07D	2E65	70C4	CC62	EC83	38E5	7129	CCF1	80BE	84E5	6104	DD12	AC82
0250:	84E5	7104	CC66	AC83	38E8	0904	0463	AC82	38E0	4909	00F2	E600	38A0	0904	0469	EC83
0254:	3855	70C6	CCF1	6553	3A75	70C4	CCF2	EC82	18E8	B513	D58A	244C	38EB	3485	D543	E58A
0258:	38E6	7024	CC62	EC82	3860	1910	CAFA	2405	38A5	7110	CCF3	A40C	38E6	7024	CC62	EC82
025C:	38E5	7111	CCF2	240C	38E5	7129	CCF1	80BD	3A65	70C4	CCF2	EC83	38E5	7129	CCF1	807E
0260:	3A65	70C4	CCF2	EC83	44E0	0904	0496	CA63	38E0	6909	8317	C671	3A85	74C4	DAB3	D3E7
0264:	BDFA	F510	D673	E73C	84E4	F109	CCF3	DC80	38B3	7110	D0F3	E6DD	39D0	0910	06DD	240D
0268:	3910	6910	26FF	E71D	38A0	6910	2F73	E6DD	44A4	7110	8CF3	E6DC	3908	090B	00F3	805E
026C:	28E3	6149	80F2	E001	3885	7104	CCF1	AC83	39D5	712A	DCC2	E000	2CE0	0B04	0E64	AC83
0270:	38E1	8979	00F4	E6F3	38E5	7110	CCF3	EA7C	3B4A	3510	D731	E40D	392B	B4C1	D754	A40C
0274:	3935	7426	D724	E71D	38E0	4910	0063	A43C	38A0	1911	20F3	E670	38A5	7110	CCF3	E50C
0278:	38F6	3028	CCF3	C081	38E8	E504	5857	8B1F	3861	F110	CCD3	E670	1C70	0B0A	0E62	F801
027C:	9CE2	F104	CCF7	6C82	0665	70C4	CC62	EC83	3CE8	B759	0C94	0802	3B06	3510	C361	E40C
0280:	38E1	F179	DCB3	FFEE	58E4	E129	CCFB	0000	38E3	7110	DCF6	A40C	5D74	B111	DCF5	E40D
0284:	1CE3	7104	CCF6	445F	BCE8	F509	D871	8100	38E0	6905	B0D7	6622	38E1	C910	80D7	640D
0288:	28E2	F159	CCF2	CE5C	3BF5	7110	CCF4	640D	38E0	F0C3	9CD1	980C	3CE5	7105	CC93	E637
028C:	38F5	7510	D8A2	FC0D	1CD3	8C2D	0093	FC00	1CE0	0B05	0E63	E720	BCE8	F109	C2F1	C8B6
0290:	38D1	C8C4	8272	EC83	0665	70C4	CC62	EC83	4510	0910	06F3	E71D	2874	7110	CCF3	E6DC
0294:	38E5	7169	CCC3	C001	2F43	712A	CCF2	E000	2CE0	0B04	0E63	EC83	38E5	7105	CCF4	A5C3
0298:	39D0	191A	0062	E000	38E5	7104	CCF3	EC83	38E0	2913	009A	65B9	38A0	08C4	10D2	EC82
029C:	18E8	B309	88F1	EC00	38D5	7510	CCDB	6C0D	3870	08C4	1062	EC82	38E5	7149	CCF3	C101
02A0:	2ED5	70C4	CC62	EC82	3870	08C4	6062	EC83	3A92	48C6	00FA	668B	38A0	0904	406A	6C82
02A4:	3A95	70C4	FC62	EC83	38D0	08C4	2072	EC83	3862	E8CD	00F3	DC04	38E5	7105	CC63	E749
02A8:	3A64	0910	14F2	E40C	3A85	6024	8063	EC82	38E5	7585	D623	E533	38E0	5979	5ED0	FFE7
02AC:	38E0	6910	26F7	240C	38E5	7510	DBC3	EB4C	38E5	7510	D646	E40D	0200	0000	0408	0002
02B0:	0000	0000	033E	0200	0000	0408	0002	0000	0000	0343	0308	0004	0000	0000	0344	0200
02B4:	0000	0408	0006	0000	0000	033F	0200	0000	0408	0002	0000	0000	0342	0300	0100	0000
02B8:	0000	0358	0200	0000	0180	0000	FFFF	0F05	0200	0000	0003	0500	0300	0000	0000	0000
02BC:	38E5	7110	CCF3	E65D	38E0	0909	0863	0071	38E4	F109	8CF3	804F	38E0	3110	CC83	B40C
02C0:	38E5	5111	BC63	340C	388C	3400	DC23	E40C	38CA	31B0	CCF3	E40C	3905	7100	CCF4	2C0D
02C4:	38E8	9510	9BE3	E2DC	38E5	7109	9CD3	D601	2CE8	A110	ACF3	E40C	3831	68C6	40D0	E6F7

ADDRESS																
02C8:	38E8	B1B0	CCF3	E66C	39D8	3179	CCF1	BD81	84E5	7110	CCF3	E40D	3A80	382B	BCD3	C002
02CC:	38EF	F904	E833	C31E	38E0	6910	26F7	240C	38E5	7510	DBC3	EB4C	38E0	0910	E0F3	E40D
02D0:	38A5	7110	CCF3	E50C	38E0	1910	E0F3	E63D	38A5	7110	CCF3	E50C	38E5	7510	D676	E40D
02D4:	38E6	B159	CCD4	0E5C	38E5	7110	CCF6	265D	39BC	350B	DBC0	C040	40E8	9759	AD54	0E5C
02D8:	38E5	7109	CC24	0601	41E5	7110	CCC1	A40D	4024	7110	CCF3	C800	2D03	6110	C0F3	E70D
02DC:	38E5	7110	CCF3	E40C	08F5	7110	DC23	E40D	78E5	6109	4CF3	E001	7C25	60C7	8CC3	E36A
02E0:	18E8	A710	AD63	E40C	38E0	0909	1092	E000	38E0	0909	4053	81A0	38E5	7110	CCF7	656D
02E4:	3815	70A5	FC80	479E	34E5	7129	CC81	A801	34E5	7104	CD93	EC82	3490	0911	1093	A57C
02E8:	38D5	70C7	CCF3	D78F	38EA	38C0	00D3	E66C	38E5	7000	CCF3	E40C	0600	0000	0200	0000
02EC:	0503	0000	0000	003C	FFF6	003C	0405	0009	0009	8000	033F	0200	0000	0502	0000	0000
02F0:	0030	FFEA	0405	000D	000D	C000	0343	0200	0000	030E	0100	0000	0000	0358	0200	0000
02F4:	0503	0000	0000	001C	001C	001C	0180	0000	3E00	0000	0000	0008	7411	0000	0000	0003
02F8:	1800	0001	0000	000D	0405	0001	0000	C000	034A	0200	0000	030E	0014	0000	0400	0367
02FC:	0180	0003	0000	0000	BCEA	F509	D677	C669	84E4	F109	CCF7	DC81	9DF3	7110	CCF3	E73D
0300:	39F5	7110	CCF3	E6DC	38E5	7104	CCF3	C99F	18E0	3B10	00FA	240C	28E5	6109	88F3	C1A1
0304:	38B5	7110	CCF3	A98C	38E5	7110	CCF3	E57C	20E5	71C0	CCF3	E40D	28A8	3045	CD93	E585
0308:	38E1	8909	04F4	01A0	38A0	083F	0708	6676	38F0	1880	20F3	E45C	38AB	B550	C1F2	E40F
030C:	80E5	7105	DCD3	E42C	80EB	3709	2084	4034	39BC	F510	C183	A40C	38E5	7110	BCD7	656D
0310:	2A19	3510	C921	E579	20E7	F110	CD93	E44C	3911	3111	CCF4	640C	4404	F11D	CCF3	C0B8
0314:	2CB3	71B4	CCF1	AC82	38E6	3510	D0F3	E40D	38E0	0909	3292	C008	2CBA	B710	2DF3	A40D
0318:	34E0	F025	DC93	FC1C	38E5	7113	CCF2	FCFD	38F5	7510	D182	244D	38E5	7509	D1E1	DC01
031C:	38E1	C900	0073	E40D	60EA	B539	C4A1	C001	38EC	F510	CE33	E40D	38E5	7509	CE84	4149
0320:	3875	711E	CCF3	E509	38EA	3110	CCF3	E40D	38E5	7509	CE84	4146	38B5	7510	C213	E50C
0324:	60D2	0B08	00D3	C201	34B0	0AC4	40F3	DF12	38A1	08C4	00F3	D80E	38A5	7510	D1B3	E50C
0328:	28E5	7110	0CF3	E40D	7172	6375	A072	F36F	5CA3	E07E	BCF3	AC04	3D21	301E	CC73	DC05
032C:	2121	301E	CC73	DC04	5CE5	7110	0CF3	E40D	7172	6375	A072	F36F	5CA5	611E	BCF3	AD24
0330:	5525	611E	0CF3	DD15	20E2	6105	8C73	FCAB	70E2	2375	A073	E461	4CA3	E31E	9CF3	F405
0334:	3CE1	3010	CC73	E40D	70E2	2375	A073	E461	2935	611E	0CF3	E804	3C72	251E	9333	F404
0338:	28E5	7110	0CF3	E40D	7172	6310	A072	F00C	5CAC	271E	93B3	AD25	2121	311E	CCC3	DC04
033C:	38E5	7010	CCF3	E40D	5CE5	7110	0CF3	E40D	7172	6310	A072	F00C	5CA5	611E	BCF3	AD24
0340:	5525	611E	0CF3	DD15	20E2	6105	8C73	FCEA	70E2	2310	A073	E40D	4CA5	631E	9CF3	F404
0344:	3CE1	3105	CCC3	E41D	70E2	2310	A073	E40D	2935	611E	0CF3	E804	3C72	251E	9433	F405
0348:	28E5	7110	0CF3	E40D	7172	6365	A072	F37A	38A0	6110	9CD1	A4FD	3D71	3063	CC73	E8FD
034C:	48F5	7210	98F1	A78C	5C65	511E	BCF3	BD25	5CC5	711E	6CF3	BD20	20E1	3110	CC73	E40D
0350:	3888	7110	CCF1	A78C	5CE5	7110	0CF3	E40D	7172	6365	A072	F37A	5CA5	611E	BCF3	AD24
0354:	5525	611E	0CF3	DD15	20E2	6105	8C73	FD2B	70E2	2365	A073	E46C	4CE3	E311	9CF1	EA8D
0358:	3CB1	311E	CC73	F4F5	4CF5	5210	D8F1	A78D	28E5	7110	0CF3	E40D	7170	A375	A072	F38A
035C:	5CA5	6062	BCF3	AD2D	48EB	5705	F683	E5A3	3888	E110	0CF3	E4F7	38B8	7110	CCF1	E7AC
0360:	3888	7110	CCF1	E7A5	7170	6375	A072	F38A	5525	70D2	CCD3	E92C	5C7F	9662	D653	BD2D
0364:	4955	7077	ACF3	F9A2	3888	E110	0CF3	E4F7	38B8	7110	CCF1	E7AC	3888	7110	CCF1	E7A5
0368:	3875	7149	CCF3	C000	2D20	707E	CC73	EC04	5CA5	600E	BCF3	BD25	5CE5	7110	0CF3	E40D
036C:	7172	6375	A072	F36F	5CA5	611E	BCF3	AD24	555F	F0D2	CCF3	F91C	2126	6112	02F3	E93D
0370:	2157	E310	36F1	AA9D	5CBE	6662	D733	BD2D	48EB	5705	B763	E5DA	488E	9310	DCF3	E4F7
0374:	38B8	7110	CCF1	E7AC	3888	7110	CCF1	E7A5	3865	7149	CCF3	C001	2D20	707E	CC63	EC05
0378:	5CA5	600E	BCF3	BD25	28E5	7110	0CF3	E40D	70E5	6375	A073	E461	3958	B512	C571	E8FD
037C:	38E5	7010	CCF3	E40D	70E5	6175	0C73	E460	38E3	F105	CCF3	E41D	5CE5	7110	0CF3	E40D
0380:	7172	6375	A072	F38F	5CA3	E07E	BCF3	AC04	3D25	601E	0C73	DC04	2125	601E	0C73	DC05
0384:	28E5	7110	0CF3	E40D	7174	6310	A072	F00C	5CA3	A11E	BCF3	AD24	0123	701E	CC83	DC05
0388:	5CE5	7110	0CF3	E40D	7172	6310	A072	F00C	5CA5	611E	BCF3	AD24	5525	611E	0CF3	DD15
038C:	20E4	6105	8C73	FE1B	28E5	7110	0CF3	E40D	7174	6310	A072	F00C	5CA3	A11E	BCF3	AD24
0390:	0523	711E	CC83	DC05	5C73	911E	ECF3	BD21	00E3	7010	CC83	E40D	5CE5	7110	0CF3	E40D
0394:	7172	6310	A072	F00C	5CA5	611E	BCF3	AD24	5525	611E	0CF3	DD15	20E4	6105	8C73	FE3F
0398:	28E5	7110	0CF3	E40D	7170	6310	A072	F00D	38A7	E349	36F4	0000	5C73	9112	FCF3	BD2C
039C:	487B	56D7	FAA3	FEAA	008E	F1B0	C4F3	E409	2D03	6210	8483	E40C	0085	7110	CAF3	E404
03A0:	2CE3	6210	8483	E40C	7170	6310	A072	F00D	38A7	E349	36F4	0000	5523	B0D2	CCD3	DD2D
03A4:	5C7F	9662	DA63	BD2D	4955	7077	ACF3	FAAB	008E	F1B0	C4F3	E409	2D03	6210	8483	E40C
03A8:	0085	7110	CAF3	E404	2CE3	6210	8483	E40C	4120	707E	CCF3	EC05	5CAE	E60E	9A83	BD25

ADDRESS				
03AC:	5CE5 7110 0CF3 E40D	7172 6310 A0D2 F00C	5CA5 611E BCF3 AD24	555F F0D2 CCF3 F91C
03B0:	2126 6112 02F3 E93D	2157 E310 36F1 EA9C	5CBE 6662 DB43 BD2C	48EB 5705 FB83 E6E3
03B4:	008E B1A0 CAF3 E4C4	48B8 731D 9CF2 2AD2	0085 7110 C4F3 E409	48B8 731D 9CF2 2AD2
03B8:	3875 7149 CCF3 C000	48B5 6270 DCF1 E69D	5CB5 600E FCF3 BD25	3948 8510 3BD3 E5BD
03BC:	70E4 2305 E0F3 E71B	38E1 4105 1C03 E41C	70E2 2221 A073 C1BD	4CE7 A106 ACF4 26F7
03C0:	3C74 111E CCF3 F406	4CAE 239E 9EF3 F407	38C5 7505 C575 2722	40E1 4305 2003 E41D
03C4:	3948 8710 1C83 E5AD	70E4 2310 E0F3 E40D	4CAE A31E 9EF3 F407	38C8 3110 CCF5 240D
03C8:	38E1 4105 3C03 E41D	70E2 2223 A073 C1AD	4CE7 A706 BC04 2723	3948 A510 7CD3 E5BC
03CC:	70E4 2305 E0F3 E74B	38E1 6105 5CC3 E41C	3948 A510 7CD3 E40C	70E2 2306 E0D4 275F
03D0:	3948 A710 5D43 E5AC	70E4 2310 E0F3 E40D	4CAE A31E 9EF3 F407	38C8 3110 CCF5 240D
03D4:	38E1 6105 7CC3 E41D	3948 A710 5D43 E40D	70E2 2313 E0D4 240D	4CA5 631E 9EF3 F406
03D8:	3CE4 3110 8CF3 E40C	4CA5 631E 9EF3 F406	40C1 6105 4CC5 241C	5D74 B111 DCF5 E49D
03DC:	24C3 7047 CCF3 F944	1CE2 7000 CCF3 FC0D	5CE4 B110 DCF3 E49D	24A3 7111 CCF5 E49D
03E0:	24C3 7047 CCF3 F944	1CE2 7000 CCF3 FC0D	5D74 B111 DCF5 E49D	24C3 7047 CCF3 F944
03E4:	1CE0 7000 CCF3 FC0C	74EA F4F5 D086 E528	3805 7510 C07D 24CC	38E5 7175 CCF6 E460
03E8:	74EA F510 D163 E40D	3B45 7510 C071 E4FD	04E1 3105 CCC3 E41C	00E1 6104 4CC3 E2EE
03EC:	00E1 6105 4C03 E41C	38E1 2105 4C13 E41D	00E1 3104 CCC3 E2EF	38E1 2175 4C73 E461
03F0:	38E3 F105 CCF3 E41D	04E1 6105 4CC3 E41D	04E1 6105 4C13 E41C	38E5 7105 CCC3 E41C
03F4:	38E5 7175 CCF3 E460	38E3 F105 CC73 E41C	38ED B575 C183 E46C	38E5 7510 DF51 A4FC
03F8:	38E5 7105 CC13 E41D	00E1 0304 4413 E2EF	00E1 0304 4013 E2EE	00E1 0304 4813 E2EF
03FC:	00E1 3105 FC03 E41D	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C	

ADDRESS				
0400:	380F F424 CDF3 D3CE	395A 8110 3C92 E40D	3987 0110 1C97 240D	0415 7108 C2F4 4002
0404:	4CD2 211C 8C73 EDCB	2CA5 611E 8CF3 F407	4CA5 631E 9EF3 F406	4CE5 5305 DEF3 E42F
0408:	380F F425 CDF3 E7CE	395A 8110 3C92 E40D	3997 0105 1C97 2627	4477 F710 F932 E40C
040C:	002F 311D EC73 C551	00EC 31F6 CC78 24DF	38E5 7510 D233 FE4D	3CE5 71A0 C901 DC03
0410:	3CE5 7103 CA91 DC03	3CE5 7105 C6F4 644E	80E3 6311 8504 640C	04EB 0711 91A2 FE4D
0414:	3A0A 3447 D431 DC63	3CF5 7110 CA93 E406	3513 6309 C4F4 4004	38E5 7511 D142 FE4C
0418:	4112 7027 CCF0 010E	3915 7513 D410 000D	34EE F1D0 CEF3 9C0D	34E5 6200 8503 E40D
041C:	391A 348E D433 E458	8203 7081 CCF4 6C5D	4515 7027 C2F0 010F	28E8 9110 E4F0 000D
0420:	40A2 74C0 D223 A40D	38E5 7510 D1C3 E45D	80E5 7105 C890 2507	3CE5 71B0 C501 DC02
0424:	3CE5 7103 C691 DC03	3CE5 7105 CAF4 649F	80E5 E311 8504 640D	08EB 0713 52E2 FE4D
0428:	3A0A 3447 D501 DCB3	3CF5 7110 C693 E40A	3515 E309 C4F4 4005	38E5 7513 D282 FE4D
042C:	4112 7025 CCF0 0142	3915 7511 D4E0 000C	34EE B1D0 CEF3 9C0C	34E5 6200 8503 E40D
0430:	391A 348E D503 E459	8205 F081 CCF4 6C5C	4515 7025 C2F0 0143	28E8 9110 E4F0 000D
0434:	40A2 74C0 D363 A40D	38E5 7510 D303 E45C	80E5 7105 C490 253B	38EF 5110 C293 FE4C
0438:	3CE5 7510 D464 640C	28D5 71B5 C891 E4F2	3CE5 7113 C4F1 DC03	34E3 6306 84F4 64FB
043C:	3A0A 3447 D431 DCFF	3D13 7109 CCF4 4004	38E5 7511 D3C2 FE4C	4112 7027 CCF0 010E
0440:	3D15 7113 C890 000D	3D15 7111 C4F0 440C	3513 6511 9F30 880C	0118 B110 CCF0 440D
0444:	3915 7510 DF30 880D	3915 7512 C1F0 01DC	28D5 71A5 C491 E526	3CE5 7113 C8F1 DC03
0448:	34E5 E306 84F4 652F	3A0A 3447 D501 DD33	3D15 F109 CCF4 4005	38E5 7513 D492 FE4C
044C:	4112 7025 CCF0 0142	3D15 7111 C490 000C	3D15 7113 C8F0 440D	3515 E513 9F30 880C
0450:	0118 B112 CCF0 440C	3915 7512 DF30 880C	3915 7510 C1F0 01DD	380F F425 CDF3 E622
0454:	395A 8110 3C92 E40D	3987 0105 1C97 264F	1C10 7108 CCD4 4002	04D5 711C C294 2C0B
0458:	4D32 2112 8C73 F40D	3505 6310 DEF3 E80C	4CB5 631E DEF3 F406	38E5 7511 D608 240C
045C:	380F F425 CDF3 E622	395A 8110 3C92 E40D	3997 0105 1C97 264E	1CE0 7106 CCD8 2627
0460:	001F 711D CD03 C951	3915 75E2 D713 E54C	38EE F105 CCF3 E5AA	38E5 7510 D673 E40D
0464:	3CF5 7110 C6F3 E405	38A0 70C7 9CD4 2E0A	3A0A 3427 D850 0202	3CF5 7110 C4F3 E407
0468:	38A3 10C7 DC74 2E0B	3A05 7427 D640 0203	3CE5 71A5 C493 E5BB	3CF5 7110 C6F3 E406
046C:	3CA5 70C7 C494 2E0B	3A0A 3427 D850 0202	38F3 1110 DC73 E404	38D0 70C7 9CD4 2E0B
0470:	3A05 7427 D6B0 0203	38EE B105 CCF3 E5E6	38E5 7510 D763 E40D	3CF5 7110 C4F3 E407
0474:	38A0 70C7 BCD4 2E0B	3A0A 3425 D850 0203	3CF5 7110 C6F3 E405	38A3 10C7 FC74 2E0A
0478:	2A05 7425 D730 0203	3CE5 71B5 C893 E5F6	3CF5 7110 C4F3 E404	3CA5 70C7 C694 2E0A
047C:	3A0A 3425 D850 0203	38F3 1110 FC73 E406	38D0 70C7 BCD4 2E0A	3A05 7425 D7A0 0202
0480:	80E5 4110 4C13 E79C	80E5 4105 8C23 E7C7	8105 4110 2CF3 E79D	28E5 4110 6C13 E40C
0484:	30E5 4105 AC23 E7C7	80E5 4110 2CF3 E40D	28E5 4110 6C13 E40C	30E5 4104 AC23 C07E
0488:	38E1 3105 CCF3 E7C7	052F F0D2 C273 FD9D	5D56 7112 CCF3 F93C	541E 6108 ACF4 4003
048C:	28D7 911C FCF3 EC0A	0551 F110 C2F3 EC0C	54B3 2110 8C74 280D	5D29 A71E D9D3 EE55
0490:	5D29 971E D9D3 DC04	38E5 7070 CCF3 E66C	38E5 7000 CCF3 E40C	092F F0D2 C273 FD9D
0494:	5D56 7112 CCF3 F93C	542E 6108 ACF3 C003	28D7 911C FCF3 EC0A	0951 F110 C2F3 EC0C
0498:	54B2 6110 8C74 280D	5D29 A71E D9D3 EE55	5D29 971E D9D3 DC04	38E5 7070 CCF3 E66C
049C:	38E5 7000 CCF3 E40C	387F B462 DA03 FA6D	610A B4D2 C573 F80C	48E5 5305 DCF3 E55C
04A0:	38E8 3110 CCF3 E40C	555F F0D2 CCF3 F80C	5D26 7112 0CF3 E93C	540E 6110 AC73 E403
04A4:	2817 A190 ECF3 E402	0071 F112 C2F3 ED3D	0551 F110 C2D4 2A9C	392F F51E CDB3 EC04
04A8:	5412 711D 8C93 C150	7520 0B1E 08F3 E804	3CE5 71E3 C673 E93D	5CE5 7105 DCD3 E6DA
04AC:	000A F510 D814 640D	34FF F910 FE91 E7A4	390A 3511 DC64 269C	340E 7027 DC93 C799
04B0:	3805 4456 1C80 2703	40FB 7710 9CA1 E407	38BE 5110 CB03 E7AD	3805 7427 DC83 C798
04B4:	340B 7656 BCC0 2702	80F3 5510 DAE1 E7A5	0008 F510 DC54 640C	34FF F910 FE91 E407
04B8:	40BB 7710 9CA3 E7AD	340E 7027 DC93 C799	3805 7456 DC80 2703	38FA 3510 DC61 E7A4
04BC:	390E 5111 CB04 269C	3805 4627 1C83 C799	340B 7656 BCC0 2702	80F3 5510 DB81 E407
04C0:	44E5 4110 3CF3 E40D	34A0 7110 CCF0 455C	3975 41F6 3C02 E7C7	34F5 7310 98F3 E404
04C4:	8103 651E DF13 FC04	38F5 7510 DBC1 E7A4	44A5 6110 ACF0 240D	28E5 4105 4C13 E47D
04C8:	1C7F FA28 BEF3 C080	38A8 3110 CCF1 E40C	38EF B0F5 FCD3 F95D	4079 A600 E573 E40D
04CC:	38EF B0F5 FCD3 F95D	40E9 A600 E573 E40D	555F F0D2 CCF3 F80C	5D26 7112 0CF3 E93C
04D0:	055E 7193 C2F3 E80D	38BF F512 CDB3 E93C	0401 F110 C2F4 6405	015F B110 C702 EA1D
04D4:	5D29 A71E DDE3 EC05	0578 F511 DD91 E40C	3CA5 71B0 C690 640D	4478 A667 BDB3 FFBF
04D8:	38EA 3511 C1F0 440C	3CE5 7111 C492 FE1C	0515 74C7 FD63 EB6F	80D5 7110 CCF3 E7AC
04DC:	38D5 702D CCF3 EBD1	38E5 7510 DF33 E7CC	38E8 30F5 CCF3 E55D	555F F0D2 CCF3 F80C
04E0:	5D26 7112 0CF3 E93C	055E 7193 C2F3 E80D	38BF F512 CDB3 E93C	0401 F110 C2F8 2407

ADDRESS																
04E4	015F	B110	C502	EA1C	5D79	A711	DDE1	E40C	0528	F5BE	DEA3	FC04	3C75	7067	C6F3	FFBE
04E8	820B	2627	BED3	EBBB	38EA	3511	C1F0	440C	3CE5	7111	C4F2	FE1C	820B	2627	BED3	EBBB
04EC	38E5	75B1	DE70	440D	38E5	7510	DF13	E7CD	38E5	7510	DF13	E7DC	38E5	70F5	FC73	F95D
04F0	3CB8	30D7	C6F3	FD5C	74E0	0B55	06F3	E7CE	28EB	7710	3F63	E40C	74AA	F110	CCF3	E40D
04F4	2CA3	B0B0	CCF3	E403	38E5	7510	C073	E4DD	5CE5	6110	8C93	E40D	58EB	6710	A493	E40D
04F8	38D5	7510	C075	E4AC	3A80	582B	D6D3	C004	38E0	6904	8313	CB2F	38E0	6910	26F7	240C
04FC	38EF	F910	E433	EB4D	38E0	0904	0573	CAF2	3A50	6910	62F3	E401	38E5	6110	CCF3	EB2C
0500	38E5	7110	CCF3	EADC	38E0	6979	B2F0	FFF9	38E0	5910	F2D3	EB4D	38E0	6904	8D13	CAF2
0504	38E5	7104	CCF6	CBF7	380A	8024	3C98	14E2	3936	8110	1C92	C40D	4CB5	631E	DCF3	F404
0508	54D5	4110	8C73	FC0C	4CB5	631E	DCF3	F404	3585	5110	CC76	E40D	5D29	971E	D314	5C04
050C	5D29	971E	D314	5C04	54E4	41B0	8CF3	FE3D	38E5	7505	D193	E4A3	380A	8025	3C98	24E3
0510	3956	8110	1C92	C40D	5D29	A71E	D314	6C04	54D5	4110	8C73	FC0C	5D29	A71E	D314	6C04
0514	3595	5110	CC76	E40C	5D29	971E	D314	5C04	5D29	971E	D314	5C04	54E4	41A5	8CF3	E49E
0518	04EB	0710	9203	FE3C	38EA	3511	F1D2	FE3D	3CA3	7427	D190	006E	3A05	7110	CCF0	440C
051C	3A05	7510	D380	880D	3A05	7112	CCF3	C00D	38B5	7112	CCF0	440D	38C5	7512	C1F0	880C
0520	3805	7110	CCF3	A40C	38EA	348E	C1F4	2459	3CE3	7081	CCF4	2C5D	4105	4112	4C13	8005
0524	40B5	40C0	8C23	E40C	40E5	4110	2C03	E45D	38A5	74C5	D383	E487	08EB	0710	52A3	FE3C
0528	38EA	3513	F1D2	FE3C	3CA5	F425	D280	006F	3805	7110	CCF3	A40C	38EA	348E	C1F4	2448
052C	3CE5	F083	CCF4	2C4C	4105	4112	4C13	8005	40B5	40C0	8C23	E40C	40E5	4110	2C03	E44C
0530	38A5	74C5	D383	E4AF	48EF	F110	CCD4	380D	38E8	3179	CCF3	FFF9	5CB4	B0DE	DCF1	B098
0534	24E3	7043	CCF3	EC9D	4575	7111	8CF5	D88D	2905	62A7	DCF3	ED45	38E8	3110	CCF3	E40C
0538	74EA	F110	CCF3	E40C	38A3	B0B0	CCF3	E403	38E5	7510	C073	E4DD	719F	F8F5	F0F6	E529
053C	28AA	B428	D493	C008	38E5	7104	D0F6	F403	28E9	B725	0DF3	E421	08EE	F545	C083	E42D
0540	4804	A045	8C98	2517	295F	B110	CCF2	C63C	2C0B	2624	9203	D3CF	38EA	3511	F1D2	FE3D
0544	3CA3	7427	D430	006E	355F	B110	CCF2	C63D	280B	2624	D2A3	D3CF	38EA	3513	F1D2	FE3C
0548	3CA5	F425	D470	006F	30EC	B509	C084	000B	28EB	7745	0083	E42C	410B	7667	2083	F060
054C	410B	7667	2083	F060	0C05	3025	CCF8	26BE	0049	3508	D537	01FF	0828	9108	2CF6	C1FF
0550	041B	0710	D5B3	E63D	38EA	3511	F590	463C	3CA3	7427	D510	0163	0828	9108	2CF6	C1FF
0554	0C1B	0710	5623	E63D	38EA	3513	F570	463C	3CA5	F425	D550	0162	3A05	4107	5C13	C18A
0558	3A05	7510	DAF0	CDEC	3A05	4112	7C13	C00D	38B5	7512	C1F0	CC0D	3805	7110	CCF3	A5EC
055C	38EA	348E	C1F4	2458	3CE3	7081	CCF4	2C5D	410F	4112	4C13	8005	40BC	00C0	8C23	E40D
0560	40E5	4110	CC33	E45C	3905	74C7	DAF0	0172	3805	7110	CCF3	A5EC	38EA	348E	C1F4	2448
0564	3CE5	F083	CCF4	2C4C	4105	4112	4C13	8005	40B5	40C0	CC33	E40C	40E5	4112	2C03	E44D
0568	38A5	74C5	DAF0	258E	74E0	0AF5	10F6	E528	38A0	0825	0A83	E5F2	28AB	7687	2083	F060
056C	34EB	7711	2084	26B1	3429	B710	0181	8801	10F5	5110	00F1	663D	2C42	E025	8CF1	E51C
0570	3D09	9710	0473	EABC	3CB9	3511	D7A3	E6BC	38F4	0908	0097	01FF	3CB3	6310	84F4	640D
0574	14F2	F110	CCF1	4C0C	1822	D025	4CF2	E51D	3D09	8710	8473	EABC	3CB9	3511	D7B3	E6BD
0578	34F3	631D	84F8	0148	14C5	3508	D926	C1FE	10E5	7105	CCD3	E570	08E5	7105	CCD3	E570
057C	39A5	74F5	CDF6	E529	5404	4045	8C98	260A	292F	B110	CCF2	C63D	2C0B	2624	9203	D3CF
0580	38EA	3511	F1D2	FE3D	3CA3	7427	D800	006E	352F	B110	CCF2	C63C	280B	2624	D2A3	D3CF
0584	38EA	3513	F1D2	FE3C	3CA5	F425	D840	006F	74E0	0B10	10F6	E40D	38A0	0824	0893	D37F
0588	28E9	B6F5	0183	E528	34DB	7667	2083	F060	34EB	7711	2084	26B1	3429	B625	0181	E51C
058C	08F5	4110	80F3	E63D	2CE2	E119	8CF1	8000	3D09	8710	8474	2ABD	3D93	5310	84F4	640D
0590	3D09	3511	D7B3	E6BD	54F4	4108	CC56	C1FE	44FB	2110	ECF2	C63D	38BC	31A5	CCF3	E65E
0594	2C0B	1625	55B8	26BE	38EA	3511	F9A2	FE3D	3CA3	7427	D950	0267	38EA	3513	F9A2	FE3C
0598	3CA5	F425	D970	0266	7600	0B05	0EF0	CEC2	3A05	7112	CCF3	C00D	38B5	7112	CCF0	440D
059C	38C5	7512	C1F0	CC0C	74E0	0B10	10F6	E40D	38A0	0824	0893	D33B	28E9	B6F5	0183	E528
05A0	34DB	7667	2083	F060	34EB	7711	2084	26B1	34E9	B719	0181	8000	0CF7	0110	C0F3	E63D
05A4	2C32	L025	8CD1	E51C	3D09	8710	C473	EABD	3CB9	3511	DAE3	E63D	3D93	5310	84F4	640D
05A8	3CF4	C108	8C56	C1FF	4558	A310	C0F2	C61C	2C0B	1625	55B8	26BE	38EA	3511	F1D2	FE1C
05AC	3CA3	7427	DAB0	02B7	7600	0B05	0EF0	8AC2	0CE5	7105	CCD3	E571	74E0	0BE0	0EF3	E40D
05B0	2A0B	7710	3B60	440D	74AA	B310	F693	E40D	2CA3	B0B0	CCF3	E403	34E8	F110	CCF3	E4ED
05B4	38E8	7110	CCF3	E40D	38E5	7510	C073	E4DD	5CE5	6110	8C93	E40D	58EB	6710	A493	E40D
05B8	38D5	7510	C075	E4AC	6140	8B10	00D1	E40C	28AB	30C0	CCF3	E40C	70E4	1210	E0F3	E40C
05BC	4CA5	631E	9EF3	F406	60C5	7305	80B5	241D	6140	8B10	00D3	E40D	70A2	22C0	E073	E40D
05C0	4CE7	A010	ACF3	E40D	3C74	111E	CCF3	F406	4CA5	631E	9EF3	F406	60C5	7305	80B5	241D
05C4	6140	8B10	00D1	E40C	28A8	B020	CCF3	E40C	70E4	1210	E0F3	E40C	4CA5	631E	9EF3	F406

ADDRESS

05C8:	60C5 7305 80B5 241D	6140 8B10 00D3 E40D	70A2 2220 E073 E40C	4CE7 A010 ACF3 E40D
05CC:	3C74 151E DC23 F407	6141 0B05 00D1 E6EB	6141 0B05 00D3 E6FE	6141 0B05 00D1 E716
05D0:	6141 0B05 00D3 E72B	000A E230 94F3 E4CC	70E4 3305 60F3 E41D	4CA5 631E 9EF3 F406
05D4:	38C5 7105 CCF5 241D	7002 2230 A073 E4CC	4CE7 A105 ACF3 E41C	3C74 111E CCF3 F406
05D8:	4CAE 239E 9EF3 F407	38C5 7505 C575 241D	0008 E2B0 94F3 E4CC	70E4 3305 60F3 E41D
05DC:	4CA5 631E 9EF3 F406	38C5 7105 CCF5 241D	7002 22B0 A073 E4CD	4CE7 A105 ACF3 E41C
05E0:	3C74 111E CCF3 F406	4CAE 239E 9EF3 F407	38C5 7505 C575 241D	70E5 6349 A074 0000
05E4:	2C05 6110 4C93 E0DD	40A5 40D2 4CF3 EFBC	34EB 6103 BCF3 E79C	38E8 7010 CCF3 E40C
05E8:	1CE4 3510 DF13 E79D	70E2 2349 A074 0000	2C05 6110 4C93 E0DD	40A5 40D2 4CF3 EFBC
05EC:	34EB 6103 BCF3 E79C	38E8 7010 CCF3 E40C	4C75 531E DEF3 F406	3CE4 1510 DF13 E79D
05F0:	70E4 23E5 A0F3 E41D	4CA5 631E 9EF3 F406	38C5 7105 CCF5 241D	70E2 23E5 A073 E41C
05F4:	28A5 611E 0CF3 F407	3D34 211E 8CF3 DC07	4CA5 631E 9EF3 F406	38C5 7105 CCF5 241D
05F8:	5CE5 7110 0CF3 E40D	7172 6310 A073 F00D	5CA5 611E BCF3 AD24	4D25 611E 0CF3 DD15
05FC:	20E4 2105 8CF3 E7C7	60E2 0B09 00D3 D000	34B0 0B10 20F3 E50D	34A0 2AC5 00F3 E569
0600:	34A2 0AC4 00F3 DF13	38A0 C824 00F3 CC6E	38A5 7510 8823 E50C	38E5 7109 CCF3 8036
0604:	38E0 1800 6083 E40C	7174 2310 A072 F00D	54A0 411E 8CD3 F407	5CA5 611E BCF3 BD25
0608:	0407 F149 CCF1 C000	4C8E D3A0 DE73 E00D	1CB7 A112 ECF3 DD3C	48EE 3309 9CF3 C007
060C:	5094 3467 D193 ED5C	3935 7105 CCF5 1C1D	7174 2310 A072 F00D	5412 4110 8CD3 E54D
0610:	5CA5 611E BCF3 BD25	3870 091E 06F3 F406	4895 7267 9CF3 E95D	2003 4149 4CF1 C001
0614:	4CA5 53A0 DED2 E00D	20B9 4102 4C83 DD3C	3876 7112 CCF3 FD3C	4CEE 3190 8CF3 E40C
0618:	50A4 3105 CCF5 241C	38E3 B510 C063 E4DC	70E4 2349 A074 0000	04A7 F11E CCF3 F407
061C:	410E E1A0 4CF3 C00D	2CA7 A112 ACF3 ED3C	4D4E 1390 DEF4 240D	5074 3105 CCF5 241D
0620:	3905 7510 C065 24DC	7014 2310 A0D3 E54D	04A9 611E 4CC3 F407	2806 7549 D261 C000
0624:	4CAE F3A0 9E73 E00C	1CB7 A112 ECF3 DD3C	393E 7180 CCF5 1C0C	3948 7110 CCF5 270C
0628:	38E5 7510 C063 E4DC	000B 344E D2C3 E408	28B9 3111 C0F3 E40D	00A5 6105 FCC3 E4C6
062C:	38E0 0810 20C3 E79C	3805 6025 4C93 E4CA	3805 704E CCF3 E408	28B5 7040 DCF3 E401
0630:	28B5 6110 4CC3 E40D	38C8 7110 CCF0 279C	34E0 0810 20C3 E79C	70E5 6110 0CF3 E58D
0634:	38A5 7108 CCF3 C01F	38B5 710B CCF3 805E	38EE 3110 CCF3 E40C	3888 301D CCF3 C395
0638:	70E5 6105 0CF3 E4D2	3885 7113 CCF3 E40C	38B5 701D CCF0 0005	70E5 6105 0CF3 E4D2
063C:	3885 701D CCF0 0003	70E5 6105 0CF3 E4D2	3885 701D CCF0 0000	70E5 7109 CCF4 00E1
0640:	28A5 6109 80F3 8041	28E5 6179 80F4 7FE0	7489 E579 9461 FFFF	0105 702F CCF3 E800
0644:	2915 7220 F4F3 EC0C	30E1 1210 C803 E40D	00E5 702E CEF3 E008	2915 7220 F4F3 EC0C
0648:	30E1 1210 C803 E40D	70E5 7110 CCF3 E40C	28A5 6108 80F3 C01E	28B5 6109 80F3 8040
064C:	70E9 2110 8AF3 E40D	288C 3179 4CF3 FFFE	2CA5 7508 D533 4E01	2805 709F FCF3 E821
0650:	28B5 729C F6F3 E42E	30C5 4310 48F3 E4CD	38B8 711D CCF0 6B92	2805 7110 CEF3 E40E
0654:	28B5 769F D503 E821	70E1 3375 E073 E461	38E3 F105 CCF3 E41D	70E6 F305 E0C3 E41D
0658:	70E1 4305 2003 E41D	7405 7110 C493 E54D	28E5 706F CA73 E7E7	38B5 728C FAF3 C055
065C:	38B5 7110 CCF3 E40C	3875 711D 86D3 EC06	2CB5 70C8 CAF3 FF01	28B1 321D 8403 E001
0660:	38B5 74C5 CD73 E591	70EB 3510 C683 E40C	38E5 70AE CCF4 0009	3805 7102 CCF4 25AD
0664:	3905 70FF CCD3 E407	70E5 628C BCF3 EC54	38E5 6110 8CF3 EC0D	38D5 701D CCF0 279C
0668:	70E9 4311 2093 E4CD	38B5 751D C073 C392	34E1 3010 CCF3 E40D	70E1 2305 60C3 E419
066C:	70E1 0305 0403 E419	70E1 0305 0003 E418	70E1 0305 0803 E419	00E1 4304 6013 E2EE
0670:	04E7 F0EF CC93 E7E6	38B5 728C FAF3 C055	38B7 C110 0CF3 E4BD	0405 711D CCF3 EC07
0674:	38A1 311D 8C03 EC06	2CA8 B51C C643 E803	38E5 7010 CCF3 E40D	040A B0C0 CDD3 E54C
0678:	0415 7105 EC13 E5A0	2C07 C31D 00F1 E40C	3815 74FF D8C3 E7E6	00E1 7105 EC03 E41D
067C:	00E9 4111 6C93 E4DD	38B5 751D C073 C793	34E1 3010 CCF3 E40D	0401 431C E0F0 88D1
0680:	28B5 7010 CC33 E78D	00E7 F0EF CC93 E7E7	38B5 728C FAF3 C454	38B7 C110 4CF3 E4DD
0684:	3815 711D 8C03 EC07	3805 711D CCF0 6C06	2C05 751C CF53 C3C0	0807 F042 CD03 E4BD
0688:	08B7 C225 00F1 E5A0	0405 731C E014 2401	1CEB 631C BCF3 EC04	38E5 64EF 9913 EFE6
068C:	1CE5 628C BCF3 EC54	38E1 E11D 8202 EF6A	80DA A22D 80F0 679C	00B5 7045 EC03 FD90
0690:	38E5 7010 CCF3 E40D	40E5 7110 CCD3 E40D	3818 31FF CCF3 E7F7	0401 711C ECF0 2401
0694:	28B5 7010 CC13 E78C	0409 411C ECF4 08E1	38AC 3110 CCF3 E79C	3908 7110 DCF3 E78D
0698:	28E1 0210 C4F3 E40C	38ED 3525 C083 E42D	34DB 3426 0083 F02C	74E8 F510 DAA3 E40D
069C:	0C17 F0A0 CCF3 E40D	040A B112 CCF0 253D	0431 B11D EC13 C540	0C25 731C E0F3 E401
06A0:	2CA5 7110 CC92 E40C	38E5 71EF CCF3 E7E7	28B5 428C 7AF3 C055	28B5 7110 CCF3 E40D
06A4:	38BA B02D 8CF3 E8D3	38E6 3100 8CF3 FFFD	28B1 338C E033 E401	28B5 7010 C0F0 A78C
06A8:	38BE F42D DA53 E803	38E5 7505 DA73 E697	0418 8310 04F3 E54D	3835 7510 C683 E54D

ADDRESS				
06AC:	0405 731C E0F0 2401	2835 711D CC13 C541	0C25 731C E0D2 E401	38E5 71FF CCF3 E7E6
06B0:	0405 628C BCF3 EC55	38E1 E11C 8290 2C00	34E7 B310 E023 FC0C	2CB1 B22D E033 EB62
06B4:	34B7 C24D C0F3 E801	0CE1 3100 CCF3 E40C	00E7 B310 E003 E7FC	3421 B210 E013 E78C
06B8:	38E1 6105 5CC3 E41C	38E1 6105 7CC3 E41D	38E1 4105 1C03 E41C	38E1 4105 3C03 E41D
06BC:	38E1 4105 5C13 E41C	38E1 4105 7C13 E41D	38D3 B062 CCF3 C7BC	38E3 B513 C073 E79C
06C0:	0001 026D 8013 C40C	38E3 B100 CCF3 E7CD	38E8 7110 CCF3 E7DC	00E5 70EF CC93 E7C6
06C4:	38B5 728C FAF3 C454	38B5 7020 CCF0 679C	38E5 7110 8CD3 E7DC	38E5 7010 CCF3 EB8C
06C8:	00EB 3510 D873 E40D	0408 30D7 CCD3 CB3A	38E6 71FF CCF3 CBF6	00E5 628C BCF3 EC55
06CC:	38E1 E110 8213 E4CC	34E1 3010 CC23 E40C	0816 74FF DCB3 E7E6	38E5 7509 CE84 4149
06D0:	3BF5 5110 CC97 E40D	38E0 0909 2873 8061	38D6 8910 26F3 E50D	3885 7110 CCF2 280D
06D4:	68E2 0AA5 00F3 EF62	38E5 60C0 8CF3 E80C	3B45 6111 8CFD 240C	39F5 6106 8CF7 E74B
06D8:	1CA0 F0C0 DCF3 E40D	3B45 7511 DE4D 240C	38E5 7509 CE84 401E	39F2 D048 FCFF E782
06DC:	60FA B110 C0F1 EC0C	BCF9 F512 DE1F E58D	BCE1 F109 CDF3 C00F	3CB3 5110 CCF3 E50D
06E0:	60E2 0B05 00F3 E608	28E6 B139 CCD3 FEFF	BCE6 6349 C9F4 1802	38E3 7510 C9E3 E40C
06E4:	38E8 3110 CCF3 E40C	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C	

ADDRESS																
0700:	7578	F4F5	D062	F128	38ED	F510	C186	E40D	39A0	E0C0	4CF3	E40D	5CA3	E072	BCF3	BC0C
0704:	38F5	7010	CCF1	E79D	3885	7010	CCF1	E79C	38EC	B510	C086	E52C	39A4	E0C0	4CF3	E40C
0708:	5CA3	A112	BCF3	BD2C	00E3	7010	CC83	E40D	74EA	F109	C2F4	0007	38E5	74F5	D1A3	E528
070C:	7508	F512	D133	F13C	3815	7108	CCF6	C1FE	5CB5	7175	0C72	E461	01A0	F0C2	CCF3	EC0C
0710:	5CA3	E072	BCF3	9C0D	38F5	7010	CCF1	E79D	3885	7010	CCF1	E79C	3825	7108	CCF6	C1FE
0714:	04B4	F595	D162	E420	04A4	F108	CCF6	C1FF	5DA5	70C2	0CF3	EC0D	5C13	8072	7CF3	A80D
0718:	00E3	7010	CCF3	E40C	00E5	7010	CC83	E40D	74E8	F509	D254	4004	381D	B508	C1B6	C1FE
071C:	0002	F111	CCF3	E6BD	3815	7108	CCF3	C1FE	39A3	F027	CCF3	EC87	3D71	3110	CC73	F0FD
0720:	38F8	711D	CCF1	AF8B	5C03	C112	3CF3	A12C	5C05	4113	2CF3	A12D	20E1	3110	CC73	E40D
0724:	3888	7110	CCF1	A78C	411B	76D7	0083	F02C	09A4	F110	CCF3	E4DD	0CB0	1B10	FEF3	E40C
0728:	2CAB	2710	B2B6	E40D	04E3	7110	CCF3	E40C	0173	7010	CCF3	F00D	5C25	4112	BCF3	AD2D
072C:	04E3	7110	CC83	E00C	5C25	4113	ACF3	A12D	00E3	7010	CC83	E40D	28E5	7110	0CF3	E40D
0730:	7172	6310	A073	F00D	5CA5	611E	BCF3	AD24	3925	711E	DCF3	DC05	20E1	6110	8CF3	E40D
0734:	28E3	7010	CC83	E40D	5CE5	7110	0CF3	E40D	7172	6310	A072	F00C	5CA5	611E	BCF3	AD24
0738:	5525	611E	0CF3	DD15	20E2	6105	8C73	FCC7	28E5	7110	0CF3	E40D	7172	6310	A073	F00D
073C:	5CA5	611E	BCF3	AD24	3925	771E	D333	DC05	5CE5	7110	0CF3	E40D	7172	6310	A072	F00C
0740:	5CA5	611E	BCF3	AD24	5525	611E	0CF3	DD15	20E2	6105	8C73	FCF2	28E5	7110	0CF3	E40D
0744:	7172	6310	A073	F1BD	5CA5	611E	BCF3	AD24	48E5	5310	DCF3	E40C	20E1	4305	2003	E41D
0748:	5CE5	7110	0CF3	E40D	7172	6310	A072	F1BC	5CA5	611E	BCF3	AD24	5525	611E	0CF3	DD15
074C:	20E2	6105	8C73	FD17	70E2	2310	A073	E40D	4CA5	631E	9EF3	F406	3CE1	4010	0CD3	E40D
0750:	70E2	2310	A073	E40D	2935	611E	0CF3	E807	3C72	251E	94E3	F407	28E5	7110	0CF3	E40D
0754:	7172	6310	A073	F1AC	5CA5	671E	9463	AD24	5CE5	7110	0CF3	E40D	7172	6310	A072	F1AD
0758:	5CA5	611E	BCF3	AD24	5525	611E	0CF3	DD15	20E2	6105	8C73	FD56	70E2	2310	A073	E40D
075C:	4CA5	631E	9EF3	F406	3CE1	4010	2C03	E40C	70E2	2310	A073	E40D	2935	611E	0CF3	E807
0760:	3C72	251E	95C3	F407	5CE5	7110	0CF3	E40D	7172	6310	A072	F00C	5CA5	611E	BCF3	AD24
0764:	48E5	5305	DCF3	E5B2	28E5	7110	0CF3	E40D	7172	6310	A072	F00C	5CA5	611E	BCF3	AD24
0768:	5525	611E	0CF3	DD15	20E2	6105	8C73	FD8E	70E2	2309	A073	801F	4CA5	631E	9EF3	F406
076C:	20E5	7110	CC73	E40D	28E7	F0EF	CC93	E7E7	38B5	728C	FAF3	C055	38B7	C394	10F3	D9D3
0770:	1C05	711D	CCF3	EC07	70E2	2310	A073	E40D	2935	611E	0CF3	E807	3C72	251E	96B3	F406
0774:	28E5	7110	0CF3	E40D	7172	6310	A073	F00D	5CA5	611E	BCF3	AD24	3925	711E	CCF3	DCC4
0778:	20E9	4111	2CF3	E40C	38B8	711D	CCF3	C393	20E1	0010	2CF3	E40C	5CE5	7110	0CF3	E40D
077C:	7172	6310	A072	F00C	5CA5	611E	BCF3	AD24	5525	611E	0CF3	DD15	20E2	6105	8C73	FDDB
0780:	70E2	2310	A073	E40D	4CA5	631E	9EF3	F4C6	3CE9	4111	2CF3	E40D	38B8	711D	CCF3	C393
0784:	3CE1	0010	2CF3	E40D	70E2	2310	A073	E40D	2935	611E	0CF3	E807	3C72	251E	9813	F407
0788:	38EC	F509	CE33	8144	5CE4	B109	DCF0	014C	2083	7110	CCF3	E44D	2D73	7109	CCF5	C005
078C:	3895	70C7	CCF3	C22A	38E0	1909	4A03	8145	38E5	7110	CC82	244D	3895	70C7	CCF3	C23B
0790:	38E5	7104	CCF3	F527	38E4	0904	01C6	4B53	38E2	4109	04F3	8060	38E5	7311	B4D2	240C
0794:	3CA0	7110	DCF4	240C	B0E5	7110	C1C3	C109	38F6	74A4	C363	CC6E	38E5	7104	CDF6	80CA
0798:	38E0	0910	0CF3	EA7C	38E2	E159	8CF2	E001	38E0	4908	0074	4131	3CE2	F110	DCF3	E10C
079C:	44E6	7104	CCF7	4C0E	2919	3509	D9F4	4101	38E5	7104	CCF3	EC83	3910	08C7	0CF3	DE72
07A0:	68EB	1310	C8F3	E90C	38E6	3149	CCF4	1203	3CE1	F110	DCF3	E40C	38E3	6110	8CF3	E5DC
07A4:	38E6	B105	CCD3	E679	38EC	F510	CE36	E5CC	38E0	6909	A6F3	8012	28E4	F109	CC02	F800
07A8:	38E3	7110	FCF3	E73C	00E2	F110	CCF3	FE7D	3805	7110	CCF4	240C	3CE5	7109	CCF3	C0E1
07AC:	20B8	F510	DC33	E50D	38E5	7109	CCF4	5000	2CEB	342F	DB13	E407	44D2	E280	C4F3	EC5C
07B0:	3915	7579	DAE4	7FF0	3CE5	7109	CCF3	C1E0	38B5	7110	CC73	E50C	38EF	F939	D602	C001
07B4:	1CE5	7104	CCF3	CC0F	2879	3509	DB81	C101	A4E0	0D04	8193	EC82	38E5	70C6	CCF0	02DF
07B8:	38E0	4959	0064	5554	387F	F8C7	D603	EAD3	3905	7110	CCF3	E73C	3915	7110	CCF3	E6DD
07BC:	39B5	7111	CCF6	E40C	3905	7110	CCF3	E63D	39B5	7111	CCF6	E40C	3D05	7110	CCF3	E63C
07C0:	38E5	7110	DC93	E40D	3CE5	7104	CCF3	C82B	38E5	7510	DC23	E40D	38EB	3510	DB13	E63D
07C4:	6CE5	7109	C693	CF95	2CA4	E06B	8CF3	C012	80C0	3105	C0F3	E75C	38E5	6110	8C93	E40C
07C8:	38E0	1909	00F3	8165	38A8	3110	CCF3	E50C	38E5	7509	DFA3	C401	38E5	7509	DD43	CC01
07CC:	38E5	7510	DD63	E40D	38E5	7510	DDA3	E40D	38E5	7510	DDD3	E40C	38E6	F509	DE03	81CB
07D0:	38E6	B509	DE03	81CA	38E5	7511	DE94	2571	38E6	7509	DEC3	8031	38E6	7509	DF13	81CA
07D4:	38B5	7110	CCF3	E50D	38E5	7510	8822	640C	6CE0	1B10	FEF3	640D	28E5	7110	CA93	E40D
07D8:	20E0	1B09	FEF3	C030	34B5	6305	8483	A7CE	6CE0	1B10	FE93	640D	20EF	EB09	00F3	C031
07DC:	34B5	6305	8483	A7CE	20E5	7110	CCF3	640D	38A0	0910	30F2	240C	38A5	7105	CCF3	A7CE
07E0:	6CE5	7109	CA73	81C8	20E5	7110	C2F3	E40D	208E	E110	8CF6	E44D	6CA5	71B0	C473	E73C

ADDRESS				
07E4:	38B5 7100 CCF3 E6BD	2075 7100 DC83 E6DD	38F5 711D CCF3 DED2	38E5 7109 CCF3 8030
07E8:	38E5 7105 8C83 E609	34EA B509 DF33 8030	8CE6 B149 C4D4 1403	38E6 7105 CC83 E4AC
07EC:	38E5 7109 CC83 81CA	38E5 7110 CCF3 E40C	20E5 7149 C0D4 1E03	38E5 7110 CC82 240C
07F0:	38E6 7105 9CD3 E4AC	38E5 7149 CCD4 1E03	3885 7425 8823 E4AD	38E5 7109 CC83 C401
07F4:	68E5 6310 C8F3 E40D	38A5 7024 CCF3 CC6E	38E5 7510 8823 E40C	38ED 3525 C083 E42D
07F8:	00E3 B109 C6F3 81CA	294B 3511 DFC5 270D	38E3 B110 CC82 240C	38E5 7509 DFF3 803C
07FC:	38E3 B111 CCF2 245C	0406 F110 CC82 2405	38E5 7109 CCF3 8039	38E5 7105 CC83 E41D
0800:	0417 F0CD CCF3 CD02	38E5 7105 CC93 E42F	0405 731C E0D4 2401	04E1 3129 CCF4 4F00
0804:	40E5 7110 CAD2 E40C	3905 7110 8AF3 E407	38B9 F11E 46F4 2809	28B9 F31C F8F4 2426
0808:	309E F11C CCF3 E40A	3106 231C F2F0 2402	4401 E010 8C13 E78D	38E1 3010 CCF3 E40D
080C:	00E7 F175 CCF3 E461	380E F025 CCD3 E45B	38E5 7529 D154 0701	0005 7110 CCF3 E40A
0810:	28B9 F11E C6F4 6588	28B9 F31C F8F4 6427	309E B11C CCF3 E4FB	3116 631C F2F1 E403
0814:	4001 E010 8C03 E79C	0005 7712 F103 E40A	38E1 3510 C071 E4FD	7411 02AD C4F3 CCE1
0818:	0405 401C ECF3 C800	2831 304D CCF3 EC0B	38E5 7010 CCF3 E38D	0C25 470C 7193 C000
081C:	0C11 F110 CC94 676D	2C07 A11C ACF3 CBE1	44DF FB78 80F4 3F80	04A7 C382 C0F8 2C01
0820:	34C0 0B05 7E74 24CE	2A00 0D10 8113 6400	4600 0ADB 7E73 D801	0915 70CD CC03 C053
0824:	39E5 7105 CCD3 E0DF	2C77 B30A E0F2 C080	2CE1 B29E E0F3 E828	38E5 629C BAF3 EC27
0828:	8406 31EC 4C94 6031	2CE8 A300 84F3 CD4C	3515 751D D363 C548	30C9 F11E CCF4 6448
082C:	38B9 E31C B8F4 6426	749E B31C 5493 E4BA	2D16 731C 72F0 64B3	4101 E11C 8C14 695C
0830:	2CD8 83ED 04F3 EB93	38E5 7410 C673 E40D	3915 74A5 C673 E599	4600 0B12 7EF3 65E0
0834:	2A00 0C49 8113 D800	38D5 74CD D243 C853	38E1 3510 D390 E4DD	383E 3110 CCF3 E79D
0838:	0401 3110 CC33 E4DC	38B8 7110 CCF0 A78D	3815 7549 D1C0 4000	0C11 F110 CCF4 676D
083C:	282F FA2D 8093 EBE3	451F FACD 80F3 C002	04D7 C305 C0F3 E8DB	04B0 0B69 7EF4 0080
0840:	0CA5 730A 48F2 C080	00A5 711D CC91 E002	38E5 715F CCF3 E407	28B5 528C FAF3 C854
0844:	28B6 3510 D2B3 E44D	0C11 F0C0 CC73 E76C	2C0F FA25 8093 E546	0818 9308 C4F2 C07F
0848:	38B5 750A D364 4080	3871 7178 CCD3 FF80	04D7 C312 C0F3 EC0D	0CB0 0B19 7EF4 3F80
084C:	2820 0D6F 80F3 E408	4405 628C BCF3 EC54	4407 A11C ACF3 EC40	38E1 F110 CCD3 FC04
0850:	2CD6 3105 CCF3 E4AF	38E5 7110 CCF3 E4BD	0CE1 3110 CCF3 E4BC	3815 7510 C693 E78C
0854:	04E8 8349 04F3 C001	3815 751D D0B3 EF91	2C08 7110 CC13 E78C	04E5 7149 C093 C101
0858:	2CEB 6579 9742 EE00	04D7 F0A9 CCF3 EA01	04B0 0B05 7E93 6DA7	28B0 0D13 80F3 640C
085C:	2805 709E CCF3 E429	28B5 729C FAF3 E427	30C5 731C F2F3 E403	28B5 731C E090 27E0
0860:	28B5 751C D6E3 EFC1	04EA 8579 5742 EE00	04A7 F0A9 CCF3 EA00	04B0 0B05 7E93 6DA7
0864:	28B0 0D13 80F3 640C	2805 709E CCF3 E429	28B5 729C FAF3 E427	28B5 731C E090 27E0
0868:	28B5 751C D6E3 EFC1	3895 7065 CCD4 3DC8	3400 0C9E 80F3 E428	28B5 709C C0F3 E420
086C:	30C5 731C E090 27E1	28B5 711C CCF3 EFC0	28D8 B110 CCF0 679D	38D8 7110 CCF3 E78D
0870:	38E8 7110 CCF3 E7DC	350E 82C5 04F3 E590	38A5 7405 E643 E591	38A8 711C CCF0 67CE
0874:	38E1 7110 CCF0 240D	38E8 7110 CCF0 640C	70EB 2509 1793 C07F	2C0B 231E 88F3 E409
0878:	38BB 211C BCF3 E409	38C8 7110 CCF0 279C	70EB 2509 17D3 C07E	2C0B 231E 88F3 E409
087C:	38BB 211C BCF3 E40A	38C8 7110 CCF0 279C	70EB 2509 1813 C07E	2C0B 231E 88F3 E409
0880:	38BB 2110 BCF3 E400	38C8 7110 CCF0 279C	70EB 2509 1853 C07F	2C0B 231E 88F3 E409
0884:	38BB 2110 BCF3 E403	38C8 7110 CCF0 279C	70EB 2509 1893 C07F	2C0B 231E 88F3 E409
0888:	38BB 211C BCF3 E40C	38C8 7110 CCF0 279C	70EB 2509 18D3 C07E	2C0B 231E 88F3 E409
088C:	38BB 211C BCF3 E40F	38C8 7110 CCF0 279C	70EB 2510 1923 E40C	38A5 7108 CCF3 807E
0890:	0405 708E CCF3 E458	28B5 728C F8F3 E453	30C1 3105 CC13 E6B3	70EB 2510 1973 E40C
0894:	38A5 7108 CCF3 807E	0405 708E CCF3 E458	28B5 728C FAF3 E452	30C1 3105 CC13 E6B3
0898:	70EB 2510 19C3 E40D	38A5 7108 CCF3 807E	0405 708E CCF3 E458	28B5 708C C0F3 E450
089C:	30C1 3105 CC13 E6B3	70EB 2510 1A13 E40C	38A5 7108 CCF3 807E	0405 708E CCF3 E458
08A0:	28B5 708C C2F3 E451	30C1 3105 CC13 E6B3	70EB 2510 1A63 E40D	38A5 7108 CCF3 807E
08A4:	0405 708E CCF3 E458	28B5 728C F4F3 E453	30C1 3105 CC13 E6B3	70EB 2510 1AB3 E40C
08A8:	38A5 7108 CCF3 807E	0405 708E CCF3 E458	28B5 728C F6F3 E452	30C1 3105 CC13 E6B3
08AC:	38B8 7110 CCF0 278C	38E5 7110 CC93 E40C	7005 6510 5B72 E40C	28B5 728C F8F3 E456
08B0:	30C1 3110 CC33 E40C	38B5 7110 8CF0 A78D	38A5 7110 CCF0 678D	38E8 7110 CCF0 3F8C
08B4:	7005 6510 5B72 E40C	28B5 728C FAF3 E457	30C1 3105 CC33 E6C3	04A5 7108 CCD3 807F
08B8:	0C28 308E CCF3 E458	3805 7510 C070 64CC	38E5 7510 C073 E4DD	38E5 7510 C073 E4CC
08BC:	00E5 7125 CCF3 E420	28E3 B104 CCF3 F526	00E5 7125 CCF3 E420	28E3 B104 CCF3 F526
08C0:	00E5 7125 CCF3 E420	38A5 7020 CCF3 E4CC	38E5 7109 CCF3 8035	38E5 7104 CC83 F526
08C4:	38ED 3525 C083 E42D	34E9 3709 2081 D804	38ED F510 C183 E79D	38E0 182F 2063 DE76

ADDRESS				
08C8:	38F1 088D 0497 0858	3810 0910 7073 E74D	38EB B579 C1F3 FFFD	0008 F51D DCD3 E274
08CC:	38E5 7510 DCF3 E4DD	38EB B426 DCF1 DF33	38D1 3710 DCD3 E67C	3860 8910 00F3 E50D
08D0:	38F1 311D CCF3 EA78	38C5 7110 CCF0 64CC	38F5 7104 CCF0 B527	38E0 0904 0673 FC52
08D4:	38E0 0910 1036 E5CC	00E0 3B04 0047 4C0E	28E9 3510 DF92 E5DD	38E1 8910 04F3 E40D
08D8:	38E5 702F CCF3 EA77	38E5 7080 CCF3 E45D	38EB B510 C1F3 E40D	10E5 5304 3843 EC83
08DC:	1834 0B10 00F3 E6BC	28EB 3509 DF92 E000	0CF0 F110 DCF6 E40C	10E0 0910 30F3 E40C
08E0:	38F5 7110 CCF0 E80D	2CB2 F111 CCF3 E6BD	38E3 7113 FCF3 E79C	3CE5 7179 CCD1 A010
08E4:	38A5 7044 CCF3 F526	00FA B510 DED1 640C	0CE0 0910 0CF3 E6DD	28A2 F111 CCF3 E6BD
08E8:	3851 3110 CCF0 A40C	38F3 7110 CCF0 240D	3CB5 7148 CC13 C00E	0CB4 F110 CCF3 E6DD
08EC:	38E3 7104 8CF3 F526	38E1 8910 04F3 E40D	38E5 702F CCF3 EA77	38E5 7080 CCF3 E45D
08F0:	38EB B510 DF73 E40C	38E5 7104 CCF3 EC83	0CE0 0910 0EF3 E40C	38A1 3110 CCF3 E6BD
08F4:	0CE4 F149 CCF3 C001	38F0 091D FEF3 EED3	00E3 6304 88F3 F526	0CE4 F110 CCF3 E40D
08F8:	38E3 7105 8CF3 E47D	A4E5 7110 CCF0 640D	38A5 70A0 CC23 E7BD	38E5 7113 CC00 240C
08FC:	38E5 7104 CCF3 F527	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C	

ADDRESS																
0900:	3915	70C5	C3F4	693F	80A5	7598	D931	DFFE	3BB5	7710	D691	E72D	B3B0	0904	C093	E97A
0904:	B3B0	0904	C093	E9F7	3BA8	0905	0323	E62F	3BA8	0905	0528	25CA	B3B0	0904	E093	E97B
0908:	B3B0	0904	E093	E9F6	B0E0	0905	7E93	E612	3CE7	3025	CD23	E56B	38E5	7510	D583	E40D
090C:	38A5	7510	DCA1	A5CC	38E5	7529	D611	C000	44E2	F104	DCF7	A82E	3875	7440	DFC3	E40E
0910:	B360	0905	7E92	E78F	B360	0905	E322	E79E	18E7	7109	C2FC	8601	28E6	F109	C2FD	8E01
0914:	28E5	7109	C2FC	4401	28E0	0B09	1E8E	5400	39A0	0910	10FB	E40C	2860	F0A0	CCF3	E40E
0918:	28E0	F105	DCF3	E69B	38FF	F910	F0F6	A40C	A5DB	311E	4CF1	E409	28BB	3109	4CF1	C101
091C:	387F	51E0	86D3	DC00	38F8	3110	CCFE	EEAC	3862	A108	DCFE	800F	2CF5	7110	CD64	66BC
0920:	3DD5	710A	CDJC	E001	38E0	0911	07A2	E5DE	3FB4	E110	9CFF	2A2C	38F5	7105	8274	272E
0924:	1C65	702F	C263	E407	3CE5	7080	CD43	EC52	38E1	E11C	CD88	699F	2CBC	0A28	00F3	C008
0928:	38A5	5110	8293	E5AD	38B8	F56B	92B8	0000	3875	74B5	D343	E52A	3B9C	08C5	09D2	FF2F
092C:	18D5	70B0	C273	E403	28E5	7105	C273	E52A	38EC	0910	41D7	DD8D	34EA	F510	D343	E40C
0930:	1F25	70C7	C292	FF2F	1B83	70C5	CCF2	E72E	38E3	5110	8CF3	E59D	34E8	0C24	0093	E976
0934:	90E0	0B10	1E93	E40C	90EA	B509	D40F	9E00	34A8	F445	D3A7	A90E	38DE	7425	D4A3	E4EA
0938:	1A10	0A34	8073	E8A6	9BE5	70C5	CC62	E72A	3B25	70C5	F262	E72B	93CC	0B10	0143	E72C
093C:	44A0	F110	CCF3	E6DD	4200	0AC5	80F3	E543	2918	B511	D104	640D	391B	3510	D4F3	E40C
0940:	90CE	7510	D483	E59C	34A8	F4A5	D9A3	E92B	38D5	7425	D383	E66B	B8E6	7129	CD42	C000
0944:	3BA5	7509	DCA1	8011	A0EA	F509	D391	80C1	9B50	08C5	3122	E72B	38E5	7509	D5A2	C801
0948:	2CEA	B510	D4A7	A58D	34DB	3435	DA13	E687	8CEF	FB09	C0F2	D801	30A4	F025	CCF3	E536
094C:	B7A3	7509	DCA1	8010	38E5	7585	D9AA	6687	391B	3509	D4F4	4007	3916	F111	CCF3	E6BD
0950:	64E0	0B11	10F4	6E3D	38F8	A510	9A61	A5F7	38BC	0909	0293	D200	38F4	E110	CCF4	25DB
0954:	2CB0	3108	C072	DE01	9507	7113	CC64	259D	1C60	0A49	070D	FFDC	3CE8	3104	CCD3	EAD3
0958:	44E4	F105	CCF3	E72E	18E3	7105	CCF3	E53E	88E0	0B05	1E73	E72E	8875	710B	CCF3	805F
095C:	38AF	F828	3E73	C020	3885	710D	CCF1	9803	3885	751D	DCA1	AC00	38EF	35E5	D4F3	E717
0960:	39E5	7445	DC83	E4E6	4475	70C8	CCD4	000F	38E5	7511	DD84	200D	3CE5	7110	C6F2	E40C
0964:	3C7C	310D	C491	A993	38D2	F11D	9CD1	9D82	3905	7579	DCA4	3FFE	390E	7425	D653	E53B
0968:	38E5	7105	CCF3	E58E	9910	1910	0063	EADC	B864	08A4	0093	E976	A8E4	9179	CCF3	C000
096C:	9105	62C8	C8F1	C801	2BE3	70C5	DCF2	E72A	A07A	F425	D713	E5C8	3907	34A5	DD83	E447
0970:	BB20	1CC5	4062	E72A	38EF	F825	EC73	E6BB	38E0	0929	3E94	007E	38E5	7510	DD83	E40C
0974:	38F5	7108	CCF3	C008	2F78	B435	D7F3	E5DA	B0E0	0825	7C62	E72A	38E5	7110	CCF2	E65C
0978:	38DB	3529	DCA1	8081	3A05	7529	DCA1	80C0	88E5	7105	CC83	E72B	38E5	7110	CCF3	E66D
097C:	38E5	71F0	CCF3	E40D	38EF	3400	D8D3	E40D	38E6	F104	CCF3	EAFA	44E4	F025	CCF2	E72F
0980:	18E5	7105	CAD3	E72F	B060	0910	BCF3	E405	2CA3	7305	84F1	A72A	3905	7510	DCA1	A5CD
0984:	B0E0	0825	E0D4	2763	34E0	0825	3E62	E72A	3BAB	3529	DCA1	8040	38E5	7105	8C63	E72B
0988:	3875	7549	DCA1	8000	3905	7529	DCA1	8009	38D5	7510	DCA1	A5CD	2CE7	3109	CC98	0021
098C:	38EE	F510	D732	E5FD	44E4	F105	CCF3	E72E	38E3	5129	8491	80BF	44E2	E105	9CF3	E72A
0990:	34E9	1129	ECF1	807F	3CE5	7105	4C73	E72B	3915	70C6	CCF4	5D3E	40B7	F11D	CCF1	E1F0
0994:	3B7E	F426	DD83	E656	80E5	7110	CCF3	E40C	8F84	E1B0	8CF3	E72D	BBE3	51A0	CCF3	E5CC
0998:	2D15	7111	CCF3	E6DC	9B20	08C7	1062	EB2A	38EC	3509	DCB2	D601	18E0	0B09	E0D2	C600
099C:	38E6	F105	F263	E72A	9BE5	72C5	8462	E72B	A318	F510	DAF2	E5FD	3B5C	3510	DCB2	E5EC
09A0:	18E0	0B05	8063	E72B	98E0	0939	1122	C001	38A5	7105	CCF1	A72B	93C4	0B10	00F3	E72D
09A4:	A0EA	F110	CCF3	EADC	38E6	7509	D5A2	C801	BCE0	0B09	0677	6000	38EB	8911	0063	E5B1
09A8:	A4BF	F91E	F0F3	E408	28B7	F1AC	4CF3	E59F	BEF8	A310	C8F6	A40D	18E1	F025	CC62	E72F
09AC:	6352	0B10	00F3	E670	2875	74C4	CF33	C9CB	3875	74C6	DA61	E6BB	38E5	7025	FC6A	672B
09B0:	38EC	3509	DCB2	C401	B060	0910	BE63	E553	38EC	35F5	DCA2	E6D7	B0E0	0905	7E63	E72A
09B4:	3B50	08C5	C062	E72B	38E5	71F5	CCF3	E6F6	38EC	3509	DCB2	C401	18E0	0B10	08F2	E5FD
09B8:	28AB	30C5	CCF3	E72E	38E5	7529	DB61	8032	3A9F	3434	DC83	E95A	38EC	3510	DCA2	E5EC
09BC:	3CE6	F595	D013	E686	38E5	7110	CCF3	E65D	3B5C	3510	DCB2	E40C	18E0	0B79	C0F8	7BFE
09C0:	28E0	0D05	6063	E72A	3A1C	34C6	DCB8	6707	38E5	7110	CCF3	E66D	18E0	0F05	2263	E72B
09C4:	38E8	3110	CCF3	E40C	38EC	3509	DCB2	C401	1C67	3030	DC73	E5F3	2CAA	F424	DC53	EADA
09C8:	BB20	1CC5	4062	E72A	38E5	7510	DA63	E5FD	3B35	710D	CCF7	3E52	39D5	711D	DC67	3E52
09CC:	A460	5D10	04F3	E74C	90EC	0B10	0153	EA7D	18EB	3509	CCBD	4C00	386B	B0E6	CCF1	A73B
09D0:	3865	7426	DCF1	A74D	3B50	1911	20F3	E670	38E0	8910	02F3	E90D	94E5	7111	DD53	EA7D
09D4:	38E0	0910	3AF3	E271	B0F0	18CD	87B1	AA64	38F8	A510	CF11	A40D	ACE8	31C5	CCF6	E53E
09D8:	B3B0	1910	83D3	E65C	3B14	A0C5	CCF2	E72F	B4AB	12CD	88F1	9A65	47E8	B104	CC73	EAF6
09DC:	2D18	317B	CCF1	8000	386A	90B0	8E94	240C	1910	4909	0064	4201	350A	B0A0	C0F3	E400
09E0:	1910	1909	0064	4801	3873	5110	FCF4	640D	44E6	5025	8DD3	E52A	38E8	0905	7E73	E72F

ADDRESS												
09E4:	38EA	9510	98A2	E40C	38E5	7509	DD84	00DB	B0E0	0905	E2D2	E61B	38E5	7105	F09A	672E
09E8:	38E5	74A5	DD83	DBE7	18D0	0B10	08F8	25EC	38A5	74C5	D773	E7BF	38E5	7549	DCA1	8005
09EC:	3BA5	757A	D778	3FC0	38E5	7549	DCA1	8005	3A05	74C6	DD88	27BA	B0E0	0825	BE62	E72B
09F0:	B0E0	0905	7E63	E72A	38E5	7510	DCB3	E5FC	1866	3105	C891	E72E	18D5	70C5	C513	E6DA
09F4:	3865	74A5	DCB3	E6DA	1865	52A0	C073	E40D	38B5	7445	DCB3	EFE6	1875	70C5	C4F3	E6DA
09F8:	8468	A6A5	BB63	E6DA	3B75	717B	CCF3	FFE0	2D0B	3510	DC53	E400	3CB8	7445	D013	E53F
09FC:	1D1F	E901	00F3	DF3D	3D15	7111	CCF3	EB3D	38A5	7510	D4F3	E6DC	80B5	70B4	CCD3	E807
0A00:	38E8	0824	0062	E72B	1EAE	0B11	0073	E63D	3905	6028	8E73	D001	38A1	6909	0091	C201
0A04:	38E1	0909	00F4	1201	40A0	1AC4	FE62	E72B	3D05	70C4	CC62	E72A	3B95	70C4	8C62	E72B
0A08:	9BB5	5110	CC83	E5FC	A8D4	A0C4	CCF2	E72B	94E3	7104	DCF3	E69A	38EC	0911	049A	AABD
0A0C:	3916	3110	F4F1	A40D	3FB8	A310	8913	E72D	38E5	7113	DF13	DADC	38F5	7110	8D04	240C
0A10:	38A5	7024	CCF3	E977	84E5	7109	CC63	E001	8509	B0CD	AC98	6C04	390C	3128	8D11	D1FF
0A14:	8615	70C5	CCD3	E46B	A105	7040	CCF3	E40C	2CAA	A104	C0F3	E7FF	84BC	30B4	CC63	E806
0A18:	B8E5	6224	8462	E72B	38E5	7104	CCF3	E806	410A	B528	D1E1	F1FE	8109	9718	B241	8001
0A1C:	3905	7138	CCF3	F1FE	38EC	2110	CC73	E40D	8355	70C4	CCD2	E72E	1DE2	0C20	0073	E40D
0A20:	1870	0B1A	C0F1	E000	28E0	0D04	4463	E72B	BB20	08C4	0462	E72B	38E5	7104	CCF3	E806
0A24:	18EB	0585	D263	E47B	3905	7528	D1E1	F1FE	38E5	7110	CCF7	A40C	3905	7538	D1E1	F1FF
0A28:	38E5	7105	CCD3	E4C8	3B28	70C4	F262	E72A	3900	0910	150A	6406	2CBF	F8CB	FAD1	801D
0A2C:	390C	0910	0294	2409	811B	3311	8924	663C	1908	B508	DB44	01E1	3DF5	751D	D958	3E55
0A30:	850B	76A5	B283	E4A2	2B85	70C4	CCD2	E72E	1B95	70C4	CD12	E72E	4504	F028	CCD3	C801
0A34:	18E6	F105	8CD1	E4E7	2903	7358	94F1	BFFE	A874	F110	4CF1	E40A	84E5	F110	CCF3	E40C
0A38:	18E3	5110	CCF1	E40D	8A9B	3435	D583	E526	3F55	70C4	C872	E72F	1900	0B08	C0F4	2000
0A3C:	3905	6104	9C63	672B	420B	3510	D404	2400	389B	34A4	97B3	E53A	3915	7509	D7B4	4005
0A40:	3DFA	B110	CCF1	A402	1D0F	E91D	0074	3C00	1905	7110	C293	E40C	2C90	CAA5	0063	E51F
0A44:	290B	3424	D7B3	E4CE	3918	B509	97B4	4005	3915	7510	D7B4	5C0C	19FB	3510	D7B3	E40E
0A48:	38B5	7434	D463	E4CF	41FA	B510	D4D1	A403	38E6	A104	CE93	E6C2	1A05	7025	C293	E5C3
0A4C:	38D6	F434	D703	E4CE	1A06	B025	CEF3	E5C3	18A6	F044	C293	E4CF	3FB9	F110	C6F3	E62C
0A50:	39EC	20C5	8C73	E54E	28EF	E904	0073	E6C3	3915	7510	D7B4	5C0C	B0E1	1905	FE73	E687
0A54:	3A25	7510	F9E4	663C	2CE5	5224	C873	E806	38E5	7110	CCFA	640D	39F5	75A5	D521	A52F
0A58:	43B9	3510	D8C3	E62C	39E5	74C5	D8F3	E686	3A25	7510	F9E4	663C	3CF5	7108	CD07	81E1
0A5C:	38E5	7105	CCF3	E497	34E6	7024	CDD3	E52A	2C74	B508	D923	C01E	43AD	FB79	FEF8	3FC1
0A60:	38A5	71B4	F094	25DE	8505	7128	CCF1	D1FE	3908	A6A4	9113	E802	42AE	B511	D663	E63D
0A64:	28E9	2021	80F1	A40C	38E8	0824	0262	E72A	3CE5	7024	CC92	E72E	3907	D108	80F7	01FF
0A68:	B0D0	0910	BE68	26CC	3915	5111	8473	E63D	1CF9	1510	D6C2	E589	38C5	5110	CE72	E595
0A6C:	4600	0910	0503	E74C	3875	751D	DA83	FED3	38E5	7024	CC72	E72B	B0E0	0904	7E63	E72B
0A70:	41F9	35B5	D734	25DA	B0E0	0824	BE62	E72A	B0E0	0904	7E63	E72B	3BB8	8510	97A3	E62D
0A74:	39E5	7425	DA13	E5EB	38E5	7105	CCF3	E5EB	3FB5	7110	C663	E62D	38E5	70A0	D073	E80F
0A78:	3868	B579	97A1	BE01	3905	7425	DA13	E5F2	3915	7105	CCF4	5EC3	38E6	F104	CCF3	E53A
0A7C:	3915	7510	D7B4	580D	2C74	B508	D923	C01E	43A2	0D79	00F8	3F81	38A0	0904	3C94	25DF
0A80:	42AA	B511	D251	A6BC	8506	7108	CCF7	01FE	38F8	A710	9118	26CD	3905	50A0	BD03	E401
0A84:	38F9	2100	C0F1	A585	38C5	7510	D881	A407	39E5	7110	BCF3	E40C	2CA8	B0C5	CCF3	E626
0A88:	B8E5	5110	8C63	E40D	38E5	7104	CC73	E72A	3905	7178	CCF4	1FFE	3BB5	7510	DA83	E62D
0A8C:	39E8	B510	98E4	240C	38B5	74C5	D8F3	E687	38E5	7511	DB01	E401	3B50	08C4	C062	E72A
0A90:	B0E0	0824	7E62	E72A	3AAB	3510	DF03	E6BC	3516	6110	CD23	E65D	38E3	61B4	FCF3	E763
0A94:	3C75	7108	CCF4	011E	38EC	08C5	0493	EA63	3905	7424	DB03	E977	3CEB	3510	D5D3	E40C
0A98:	44E2	F113	DC91	A66D	3CA5	7111	CCDA	A6BD	3915	7110	F4F3	DABC	3CF8	A310	8914	25EC
0A9C:	3915	7089	DF18	800B	38F5	7008	8D07	81E0	38AC	0911	049A	AEBD	3CCB	3512	D5D3	E2BD
0AA0:	3CF8	3110	CCD4	240C	391C	0910	0493	E40C	3CE0	F113	CCF1	AC0D	3CA4	F0D7	CCF3	ED77
0AA4:	18F8	A310	80F1	A40C	2CCF	EB08	1E63	C1E1	38B0	0827	20F3	E176	18E3	6000	8CF3	E40D
0AA8:	3905	7028	BCD3	C801	8104	F00D	CCF3	E19B	B0E3	6310	C0F3	E250	4208	A396	C8F8	26B6
0AAC:	2CE3	7113	CC93	E591	80E4	E110	CCF3	E66C	38E3	7179	8CF3	D000	3A08	2024	CC93	E977
0AB0:	4105	702F	C2D3	E407	38E5	7080	82F3	EC53	2CE9	A179	8CF3	C005	3918	2110	CC94	6A3D
0AB4:	47B6	7110	CCF3	E72C	38E5	7513	D5D3	EADD	38E5	7104	CCF3	E6F7	3B65	70C4	CC62	E72A
0AB8:	3B50	08C4	D262	E72A	B360	18C4	0062	E72B	38E5	7104	CCF3	E6F7	3B15	70C4	FC62	E72B
0ABC:	38F5	7104	CCF3	E722	38E6	F110	CCF3	E40D	47B5	7110	CCF3	E72C	38E5	7113	D073	EADC
0AC0:	B8FE	E109	8C64	0008	2B25	70C4	4C62	E72B	9BE0	18C4	0062	E72A	93CC	0BA0	00F3	E72D
0AC4:	3905	7100	CCF3	EADD	3905	7110	CCF3	EEDD	38E5	7104	CCF3	E69B	38E0	1979	FEF2	C001

ADDRESS																
0AC8:	38A6	3104	CCF3	A72F	21A0	0B04	1E7B	FC3B	3871	F110	CDE3	A43D	2865	708E	C3C3	E459
0ACC:	38B5	708C	CCF1	A45D	3870	0910	1E73	A5FD	18E6	F11E	CCF3	E408	2897	F3AD	F4F3	DC45
0AD0:	38E5	6129	C0F2	C000	38A0	1910	FEF7	6801	38A0	2804	0093	A72F	18E0	4B10	0097	5804
0AD4:	18D5	70C4	CCD2	E72E	18E0	0B09	20F2	E600	38A5	7024	8C63	E44B	38D5	7024	F2F3	E44A
0AD8:	B0A5	7104	CC91	A72F	38D5	70C4	CCF2	E72E	38E5	7040	84F3	D80E	18E0	0A24	0E93	EBB3
0ADC:	38E0	1824	FE92	E72F	38E5	5110	84F3	D80B	38B5	70CD	8C64	2802	38D1	6904	00F3	A44B
0AE0:	38A5	70C4	CCF2	E72F	3865	7108	CCF3	800F	2D05	708E	CCD3	E458	28B5	708C	CCF4	245D
0AE4:	38E5	7113	CCF3	A80D	41D6	F11E	CCF1	E409	28E7	F3A0	F4F3	E44D	3895	7048	CCF3	FFF0
0AE8:	38B5	7110	CCF3	A40D	1CE0	0B09	F0F3	81FF	3895	711D	CCF7	6803	2CE0	4D04	0063	E44B
0AEC:	38D5	710B	CCF3	804F	38E5	7024	CCF2	E72F	1885	7110	C4F3	E40C	2CE5	6224	8463	EB77
0AF0:	38F5	7108	CCF3	E000	2D1B	3510	D7B4	5C0C	B0E0	0824	BC62	E72B	3865	70C4	CCF2	E72F
0AF4:	98E0	08A0	80D3	D80C	B0E0	0824	E262	E72A	B0E5	7024	C082	E72A	B0E5	7104	8C63	E72A
0AF8:	B0E0	0804	E283	E72B	B0E5	7104	D063	E72A	38E5	7510	D7B3	E40C	38E5	7509	CE84	4149
0AFC:	38E5	7110	CCF3	E40C	2085	711C	DC82	2401	38E5	7105	CCF3	A71E	38E5	7110	CCF3	E40C
0B00:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B04:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B08:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B0C:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B10:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B14:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B18:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B1C:	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004	0001	0002	0003	0004
0B20:	39D5	711D	DC87	3E52	A4E0	5D10	04F3	DB4D	38E5	7110	CCF3	EA7C	18EB	3509	CCB8	1804
0B24:	38EB	B0E6	CCF1	988E	38E5	7426	D241	9B4C	3A00	1910	20F3	E67C	A4E5	71C0	DCF3	E90D
0B28:	38F0	88C7	02F3	EB30	38E5	7111	CCF3	EA7D	38E0	0910	3AF3	E271	38F5	70CD	CCF1	AA64
0B2C:	2CFB	3110	CCF1	A40D	3861	31C8	FCF3	C038	38B8	30C5	CCF3	E750	74E0	0B09	0EF3	D668
0B30:	2CE4	6109	8CF3	C011	74E0	2310	C8F3	E40D	38A5	7025	DCF3	E4F8	28E8	3349	F78A	0000
0B34:	38E6	3105	CCF3	E4F3	38E7	3105	CCF3	E4F2	38E5	7105	CCF8	F8F2	38E5	7105	CCF3	E4F2
0B38:	38E0	0905	06D3	E76F	38E0	0905	08D4	279A	38E5	7105	CCF3	E4F8	38E5	7105	CCF3	E4F8
0B3C:	38E0	09B4	0673	FC53	54E5	4110	0C03	A7EC	5575	4179	8D32	FFFA	4955	4310	1CF1	C80D
0B40:	5405	411E	4C13	FC05	48A5	631E	9CF3	FC04	4875	531E	DCF3	FCA4	38E2	4139	0CD4	7FFE
0B44:	38E5	7110	CCF2	C69D	3CE5	7109	CC83	8009	44F5	6310	8892	240D	3910	091D	0704	2C04
0B48:	3892	7179	9CD3	BFFA	80EF	F911	FF02	FE9D	3C95	7109	CC83	8008	34F5	6310	8492	240C
0B4C:	8108	B51D	D484	2C03	38BF	3585	DAE2	E6BE	0040	0B08	7EF3	C07E	28B0	0D0A	8003	C081
0B50:	00BF	FB10	8101	240D	3846	6198	4D18	3F80	38E5	7000	CCF3	E40C	7608	F510	D764	640C
0B54:	350B	3425	DF73	E7DE	0405	7110	CC92	E40C	38E5	716F	CCD3	E406	28B5	528C	3AF3	D455
0B58:	28B5	7110	CDAA	E40C	38E5	7110	8CC2	BC0D	0405	7110	CC92	E40C	38E5	716F	CCD3	E406
0B5C:	28B5	528C	BAF3	DC55	28B7	A11C	4CC2	B801	AAB1	B11C	CDAA	E401	38E5	7110	8D43	E40D
0B60:	1445	7110	CC92	E40C	38E5	716F	CCDA	6407	28B5	428C	BAF3	CC55	28A7	A11C	4CC2	B800
0B64:	AAB1	B11C	CDAA	E401	38E5	7110	CCF3	E40C	9067	B110	8D42	E40D	1E91	B11C	CC9A	6400
0B68:	38E5	71EF	CCD3	E407	28B5	428C	BAF3	CC55	28E5	7110	CCF3	E40D	28A5	7110	CAF1	E407
0B6C:	9297	A110	8C93	E40C	2E31	B110	CCF2	A40D	38A5	711C	CCF3	F800	2877	A11C	4CD2	EC01
0B70:	AAB1	B11C	CC04	2401	84E6	7110	CCC3	A43C	8201	F138	CCF3	FFFF	2E05	6110	8CF3	E40C
0B74:	80A7	E369	C0F3	C101	38E5	6105	CD03	E71F	40DB	3425	DF73	E697	38E5	4110	8AF3	CC07
0B78:	2CE5	6310	8424	040B	38E5	4110	86F4	0407	40E5	6310	8413	C00B	40E5	6310	C404	1C07
0B7C:	38D5	4112	CA33	D00D	38E5	511C	8AF3	ED5C	40E5	6310	84D4	140A	38E5	5110	86F4	1407
0B80:	40E5	6310	8453	D00B	40E5	63F8	C443	8035	3A05	7110	CC93	E44D	80E7	E310	C0F2	DC0A
0B84:	1056	711E	CCF3	E409	0015	628C	BCF3	EC55	38E5	711C	4284	2000	34E5	7110	CC59	BC0D
0B88:	38E5	611C	FC44	680D	38E5	71EF	CCD3	E407	28B5	428C	BAF3	CC55	28B5	7110	CCD2	E40D
0B8C:	38E7	B31C	ACD2	FC0D	1901	B31C	AC74	3D90	38EA	A529	992A	8000	0837	B11C	8CD2	FC01
0B90:	0101	911C	CC74	0400	4445	7110	CC59	A40C	3906	51FF	CCF3	E406	0015	628C	BCF3	EC55
0B94:	38E5	611C	8359	6C00	34E5	7110	CC64	7C0C	38E5	71EF	CCD3	E407	28B5	428C	BAF3	CC55
0B98	28B5	7110	CCF3	E40D	9655	631C	ACF3	ED90	38EA	A511	99D3	E40C	4465	731C	ECF1	AC01
0B9C:	38E5	7110	CCF4	680C	3A65	5110	8C91	240D	8201	F138	CCF3	FFFF	28D5	61E0	ECF2	E43C
0BA0:	46A5	7100	CCD4	680D	38B5	7179	CC94	7F81	1055	702F	CCF3	E406	38E5	628C	BA04	2C56
0BA4:	44E5	7105	CD03	A71E	6107	3025	CC33	E7DE	8C44	7108	CCF3	C07F	44E3	6110	CCF3	E40C
0BA8:	14A3	7110	CCF3	E79C	18E3	7110	CCF3	E40D	1CE3	7110	CCF3	E40C	3830	0840	14D3	CFE0

ADDRESS

0BAC	8CE5 7010 CCF3 E7FD	3955 7512 C700 288C	3A87 7110 CCF0 000C	38D5 711D CCF3 FC02
0BB0	34B8 B025 CCD3 E7DF	3905 70C0 CC03 E40C	38E5 7110 CC91 240D	1877 811C ACF3 CC00
0BB4	0045 4110 02F3 D002	2C51 A11C ACF3 C400	3A8F F8DE F8F3 E809	28E9 B110 DCF3 E47D
0BB8	3846 351D D4E3 C140	8201 F138 CCF3 FFFF	2CE5 6110 ACF3 A40C	38A1 C910 00F3 6801
0BBC	40CB 3462 DFA3 E80C	38EB 3505 9FA3 E7EB	1877 B31C E0D2 E400	1051 B31C E0F3 E400
0BC0	28B5 709E CC44 2428	28B5 729C FA44 2427	0837 B11C 8CD2 FC01	1101 811C 0C04 0401
0BC4	28B5 731D 84F3 FC32	80B7 F11D CCF3 E802	80B5 6025 8303 E7DF	0109 F11E CCF3 E408
0BC8	28B9 731C F804 2426	38E7 B11C 9CD2 FC01	0101 B11C CCF3 E400	28B5 731C FA04 2407
0BCC	6095 711C CC53 E40A	80B5 411C 0CF3 E40A	2CB7 E310 B490 2B6C	8CE4 7110 CC73 E40C
0BD0	3803 4110 0CF3 E79D	40E3 7110 CCF1 BC0C	3903 711D 8CF3 FC03	18B3 711D CCF3 EB82
0BD4	34D8 A235 C4F3 E75A	38E5 7410 DD63 E5AD	3850 0910 10D3 D7E0	38EE E040 8CF3 EC0C
0BD8	38E1 2010 8CF3 E7FD	395E F112 CCF0 1C8D	38E5 7105 9CD3 E5C1	38EC 8513 0084 240D
0BDC	5575 6110 8C02 F00C	38E5 7110 CCF3 E4CC	410B 3711 BE51 C29D	5CB5 611E FCF3 BD25
0BE0	4905 52C1 DCF4 240C	3885 7149 CCF4 4000	38C4 4025 0CF3 E77A	5C05 4112 3CF3 BD2C
0BE4	4513 701D CC83 F392	38E5 7010 CCF3 E79D	04ED 3525 C083 E42D	5555 6110 8CF0 000C
0BE8	28A5 7510 DF10 64DD	74A1 12AD C4F3 DC01	1CE7 F010 CCF3 E40D	40D5 7062 AC73 DFBC
0BEC	2CE8 B513 DEF4 279D	0475 7425 DF13 E41D	38E5 7510 DEB3 E40C	38EE F510 C073 E76C
0BF0	3905 701D CCF3 EF91	5D05 7110 0C72 EA9C	5CB5 611E FCF3 A925	4122 411E 0C73 FC05
0BF4	5CA5 611E BCF3 9D24	4885 5310 DCF1 E40D	2105 7001 CC94 240C	8CE4 702F CCF3 E406
0BF8	38E3 7080 CCF3 E45D	38EF 3410 DAB3 E79D	8C04 7108 CCF3 C07E	80E3 6110 CCF3 E40D
0BFC	64A3 7110 CCF3 E79D	0823 711D CCF3 C403	0CB3 701D CCF3 CF83	38E5 7110 CCF3 E40C
0C00	3816 3508 D1A1 83FF	3805 7508 D1A1 83FF	38E5 7105 CCF3 E4F8	38E5 7105 CCF3 E4F8
0C04	38E5 7105 CCF3 E4F8	38E5 7105 CCF3 E4F8	3807 3508 D1A1 83F0	38E5 7105 CCF3 E4F8
0C08	38E5 7105 CCF3 E4F8	BC00 0D04 03F1 C3E3	3805 7105 CCF1 E410	3805 7510 D423 E54D
0C0C	38E5 7511 D3C3 E40D	60EA B510 C4A6 E40D	38F1 8975 0474 6461	38A0 0910 0E73 E67D
0C10	3918 0909 02D4 1800	28F1 3125 CCF3 E421	1D10 F0C0 DC63 E40D	38E5 7145 CCF3 E421
0C14	18E2 F109 DCF2 C01B	40F4 F110 CCF8 E5BD	3CE5 7129 CD14 4001	3910 00D7 0293 FCF8
0C18	380B B510 C1F7 640C	34E8 3110 C293 E5CC	34E0 0B04 3C97 0C0A	1875 50D7 90F3 DB3C
0C1C	8465 6188 9C97 40FF	38DF 3510 D504 25E3	35B2 F110 CCF7 A40D	38E3 7110 FCF3 E40C
0C20	3CEB 3510 F393 E40C	38E5 7113 CCF7 A80C	3CE7 F395 8873 E4A3	30E4 0B29 00F2 E001
0C24	3878 A568 9374 0001	44E8 A310 C0F3 E79C	3DD8 F425 D2E3 E4BB	38E5 7511 D373 E79C
0C28	451B 1710 F373 E79C	38B1 31B6 CCF4 2CDE	3CE0 0B39 1EF2 E001	4505 6027 9CD3 DCBB
0C2C	38EB 2511 F2E3 E79D	44E0 0910 04D3 E79D	38EF F910 FAF3 E40C	34D4 F190 CCF3 EB3D
0C30	1800 0B10 0E63 E6DC	38E5 7104 CCF3 EC83	38E3 7110 BCF3 E40D	3943 B1E1 CCF5 270C
0C34	38E0 8910 00F3 E4CD	38A5 7110 CCF3 E67D	38E5 7510 C073 E40C	38D5 7110 CCF3 E73C
0C38	39E5 7510 D333 E6DD	38E5 71FF CCF3 E407	38E5 7080 CCF3 E45D	38E5 7105 CCF3 E47A
0C3C	08BD 3427 C0B3 F021	04A7 F1B4 CCF1 8C0A	3823 B508 D444 00FE	4508 B5A5 D332 C503
0C40	38E5 4024 0C63 EC83	38E5 7510 D333 E79C	04E5 7145 CCF4 2420	28A0 0B34 FE91 8C0A
0C44	18EE 91A9 90FA 3FF6	84E0 E027 9CF3 DB3D	38EE B509 D4A7 6000	38F4 09FF 00F3 E407
0C48	38F8 A080 DCF3 E45D	38F4 0826 00F3 E512	40F8 B51D D3F4 6804	38D8 A5A5 D332 C132
0C4C	38E1 3024 CCF7 6C83	3A05 70B0 CCF3 E40D	2E88 F426 D4CA 253F	3865 7510 D330 65EC
0C50	18F5 5109 86F3 C035	2D03 210B 8CF3 805F	38E5 7110 CC73 E40D	3886 0913 00F3 E40C
0C54	3CA3 62C4 C8F2 EC82	38E0 0979 0872 FFF1	34E4 D110 CC93 E40D	38E3 70C6 CCF2 FD5A
0C58	38E0 0909 10D2 DC01	38E2 0904 0093 EC82	34EB 1379 88F4 7BFE	38E1 3509 D332 C201
0C5C	38E0 2904 0093 EC82	18E8 B309 88F1 EC00	38D5 7510 D334 2F9C	3870 08C4 1062 EC82
0C60	38E5 7149 CCF3 C101	2D05 70C4 CC62 EC83	3870 08C4 6062 EC83	3912 48C6 00F4 658E
0C64	38A0 0904 4064 6C83	3915 70C4 FC62 EC82	38D0 08C4 2072 EC83	3862 E8CD 00F4 5C04
0C68	38A5 70C4 CC62 EC82	3915 7512 D333 DF9D	70E5 70F5 CCF6 E529	38A5 6109 4C73 81CD
0C6C	70EB 3509 D724 0401	388D F508 C186 C03E	1C80 4978 00F3 FFC0	38A3 F110 CCF3 EEBD
0C70	38E5 7110 CCF3 E40C	38F5 7010 CCF1 E79D	38E5 4125 0CF3 E421	3905 7110 CCF3 EABC
0C74	38E5 7110 CCF3 E40C	38F5 7108 CCF6 C03E	1CF0 4978 00F3 FFC1	38A5 7110 CCF3 EEBC
0C78	38E5 7110 CCF3 E40C	38F5 7010 CCF0 279C	70E5 70F5 CC96 E529	38A5 6109 4C73 81CD
0C7C	70EB 3509 D824 0401	3885 7108 CCF6 C03F	1C80 4978 00F3 FFC0	38A5 7110 CCF3 EF3C
0C80	3805 7110 CCF3 EABC	38C8 7110 CCF3 E6DC	3105 7110 4C73 C2BD	38E3 B110 CCF3 E40C
0C84	38FD 3508 C086 C03F	38F5 7178 CCF3 FFC0	3875 7110 CCF3 EF3D	3813 B110 CCF3 E40C
0C88	38C8 7110 CCF3 E6DC	38E0 0910 1E76 E40D	38E0 4910 0093 E40D	38E5 7110 CCF3 E40C
0C8C	1CDF F910 FE73 DEBD	3075 7109 CCF3 81CC	30F8 B510 D8C2 240D	38E5 7184 CCF3 F526

ADDRESS				
0C90:	38E5 71A4 CCF3 F24F	38E5 7104 CCF3 C0CA	38E6 B104 CCF3 F226	38ED B510 C1B3 E40C
0C94:	38E5 7109 CCF3 8077	3985 7110 CC86 E40C	50E4 F119 CCF3 E000	2CE3 7109 CCF3 8071
0C98:	3941 8910 D493 E70D	38E5 7510 DE43 E40C	38F5 7110 8C73 E40D	78E5 7105 CC93 E793
0C9C:	1CE5 7310 8073 E40D	7CE5 7109 CC93 D8D5	2CE4 F110 CC83 E44C	38EB 1109 CCF3 C021
0CA0:	34E4 E310 C083 E40D	50E4 F119 CCF3 E1A1	2CE3 7109 CCF3 8071	3941 8910 F493 E70C
0CA4:	38E5 7510 DE43 E40C	38F5 7110 8C73 E40D	78E5 7105 CC93 E793	1CE5 7310 8073 E40D
0CA8:	7CE5 7109 CC93 D8F4	2CE4 F110 CC83 E44C	38EB 1109 CCF3 C021	34E4 E309 C083 804F
0CAC:	50E4 F109 CCF3 C01D	38E3 6109 CCF3 DA07	38E4 E110 CCF3 E40D	38E0 3109 CC83 8077
0CB0:	38E5 7110 CCF3 E40C	388C 3400 DB33 E40C	38E5 7110 CCF3 E40C	38C5 7110 CCF1 E40C
0CB4:	38B5 7110 CCF3 E403	2CE3 FB05 FE93 E792	3875 7168 8C74 0002	38E0 0905 3A93 E792
0CB8:	1CE5 7309 8073 C03A	38E5 6109 C0F3 8046	3901 A91D 0694 2802	2CE0 0E20 02F3 DC0D
0CBC:	34E4 F110 CCF4 280D	38E0 3110 CCF1 E4FC	40E4 F110 CC83 E40D	38E5 7110 CCF3 E40C
0CC0:	38EC 3400 DC13 E40C	38E0 5910 20F6 E4FD	3942 E111 8CF5 270C	38E0 0909 08F3 8075
0CC4:	3CE3 6309 84F3 C5FF	38B5 7110 CC83 E6BD	38E5 7109 CCF3 C422	38F5 7110 CCF3 EF3C
0CC8:	38E5 7110 CCF3 E6DC	38E5 7110 CCF3 E6DC	38E0 0910 7EF3 E6DC	38A5 7110 CCF3 E6DD
0CCC:	38E5 7010 CCF3 E40D	70E8 B0F5 CC96 E528	00EC B509 C082 C400	2CA5 6110 4C73 EEBD
0CD0:	70DB 3110 CCF3 E2BD	38F5 7508 DD46 C03E	70C8 B178 CCF3 FFC1	38B5 7510 DD83 DE3C
0CD4:	38F5 7108 CCF6 C03E	38C5 7178 CCF3 FFCD	38BD F510 C183 DE3D	38E3 F104 CCF3 CCD3
0CD8:	3CE1 3010 CC03 E40C	38E0 48F5 0096 E528	70A5 7110 CCF3 C2BD	38D5 7110 CCF3 EABC
0CDC:	70FB 3108 CCF6 C03E	34CC 2178 4CF3 FFC1	38B3 B510 DE23 EB3C	38F5 7108 CCF6 C03E
0CE0:	34C5 6178 4CF3 FFC0	00B3 B510 DE33 EB3C	04E3 B145 CCF3 E421	38A8 7110 CCF3 E6DC
0CE4:	38E5 7110 CCD3 E65C	34EB 3510 DE93 E40D	34EA F109 CCF3 C003	2CD5 7310 80D3 E402
0CE8:	2CB5 70C5 CC93 E79A	38E8 B110 CCF3 E66C	38E8 3110 CCF3 E40C	38EC 8509 0083 8078
0CEC:	3944 E111 8CF5 270C	38E3 B109 CC83 807A	38E5 7010 CC83 E40C	38E5 7109 CCF3 807D
0CF0:	38E5 7104 CC83 F526	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C	

ADDRESS				
0D00:	38F0 0909 2A93 DA0A	2CE4 F109 CCF3 8033	38E0 3110 CCD3 E57C	34F2 F06B CCF3 C021
0D04:	38E8 2105 BC73 E400	3CE5 7513 D443 EABD	3CE5 7513 D463 EABC	BCEF EB05 FFF3 E527
0D08:	3945 7511 D405 270D	38E0 0904 0693 E313	38E2 0904 0097 5E97	38E5 7104 CCF3 DE22
0D0C:	38E5 7104 CCF3 E34F	3945 7106 CCF5 252A	38E5 7104 CCF3 E2F3	38E5 7104 CCF3 E2FA
0D10:	38E5 7104 CCF3 E302	38E5 7104 CCF3 DFDF	38E6 3104 CCF3 F227	38E6 7104 CCF3 F24A
0D14:	38E5 7104 CCF3 F3AF	38E5 7104 CCF3 F3BE	38E5 7105 CCF3 E4F8	38E5 7105 CCF3 E4F8
0D18:	38E5 7105 CCF3 E4F8	38E5 7105 CCF3 E4F8	38E5 7105 CCF3 E4F8	38E5 7105 CCF3 E4F8
0D1C:	38E5 7105 CCF3 E4F8	38E5 7104 CCF3 F603	38E5 7104 CCF3 F64B	38E0 0904 0AD6 F733
0D20:	38E5 7104 CCF3 F802	38E0 0904 0873 F883	38E1 3104 DCF6 F9DF	38E5 7110 CCF3 E40C
0D24:	38E5 7110 CCF3 E40C			

```
ADDRESS
 0D40:    BCE0 1D10 01F3 E40D    60EA B110 C0F3 E40D    BCE9 3510 C953 E40C    38E8 7110 CCF3 E40D
 0D44:    3BF5 7175 CCF2 E860    38F5 7510 D491 ECFC    3BF5 7125 CCF2 E821    34F4 F110 CCF3 EEDC
 0D48:    0003 7512 D49F E4CC    3945 7511 C065 270D    2CD4 3111 CCF3 E63C    38E0 0949 0E91 C000
 0D4C:    60FA B11D C0F3 EED3    38E6 8549 C9E4 1802    BCD8 5310 C5F3 E50C    38E5 7110 CCF3 E40C
 0D50     38E5 7110 CCF3 E40C
```

ADDRESS				
0D80:	50E4 3109 DCF3 81D3	38E0 0910 08F6 E40D	3882 E178 8CF3 FFC1	2C80 4908 12F6 C03E
0D84:	3D60 E179 8C73 FFFE	3942 A111 CCF5 240C	38FB F828 FEF3 C400	3D23 6219 88F3 FFFD
0D88:	38EF F910 F4F3 EE2C	48A0 0910 46F3 DEBC	3CE0 A110 AC73 E40D	38F5 712A CCF4 0000
0D8C:	38F5 70C0 CCF2 E40C	2D03 7110 DCF3 E6DD	28AF F910 F2F3 DF2D	38A9 20A1 8CF3 E40D
0D90:	38E8 7110 CCF3 E6DD	38E5 7010 CCF3 E6DD	3945 7111 CCF5 270C	38ED 3510 C0B6 E40C
0D94:	38ED 7109 CCF3 C009	38E5 7510 C083 EEBD	38E5 7109 CCF3 81D3	38E5 7110 CCF3 E40C
0D98:	3885 7178 CCF3 FFC1	2C80 4908 12F6 C03E	48FF F910 FCF1 EABD	38B2 A179 8CF4 3FF5
0D9C:	48F0 0828 46F3 C400	3CE2 A105 ACD3 E6EB	38F5 7110 CCF4 640D	3D05 7110 CC93 E6BD
0DA0:	38EB 2110 8CF3 E40C	38FA B558 DA54 3FFF	34E5 7110 DC93 E40D	38E4 B110 9CF3 E40D
0DA4:	38E3 7110 CCF3 E40C	38D4 B179 8CF2 FFFE	2CE3 7020 CCF3 EC0C	1CEF F8A5 EEF3 E6C2
0DA8:	2902 F110 CCF3 E6DD	3915 7178 CCF4 77FE	3D16 0B08 00F3 C800	28EB 3025 CCF3 EEC3
0DAC:	3875 7510 DB03 E73D	3D10 0D10 04F3 E6DC	28E3 7110 CCF3 E40D	38E5 7510 DC63 E40C
0DB0:	38D5 7179 CCF1 FFF2	2CE9 3109 CCF3 C00C	2C14 B020 8CF3 E40D	1CEA B0B9 CCF1 C801
0DB4:	34E4 B105 8CF3 E6EB	04E3 7020 CCF3 FC0D	38E5 7105 CCF3 E6EE	3825 70C8 CCF3 FF80
0DB8:	38E5 7105 CCF3 E6EE	38E5 7105 CCF3 E726	38E5 7179 CCF1 F7FF	5820 B108 CCF4 007F
0DBC:	08AF FB10 80F3 E40D	28FC E912 80F5 AC0D	28BB 3579 DC43 FFFF	38E5 7110 CCF3 EEAC
0DC0:	38E5 70B0 CCF3 DC0C	38F5 711D CCF3 DED4	38E5 7030 CCF3 DC0D	38F5 711D CCF3 DED2
0DC4:	3905 711A CCF0 2601	38E5 7010 CCF3 E79D	38E5 7109 CCF0 4028	38E5 7109 CCF0 8401
0DC8:	38B5 7010 CC03 E79D	38E5 7109 CCF0 8402	3825 7110 CCF0 640D	3825 7010 CCF3 E79D
0DCC:	38E2 E109 BCF3 81D3	396D F51D DE63 EAAE	3C80 4978 04F3 FFC0	2C83 2108 8C97 003F
0DD0:	58E4 B319 ACF1 FFFE	3D49 3511 DE65 270D	3CF3 5338 C8F4 3FFF	3CB8 0AC5 00D4 6F7A
0DD4:	38E0 3909 F863 81DB	3505 311D CCF3 DC04	2C83 7378 84F3 FFC0	1880 E108 CCF6 C03E
0DD8:	38E0 E110 BCF3 E79D	38F5 7110 CCF5 640C	38F5 7110 CCF6 640C	39A5 502D DCF3 EC01
0DDC:	28E2 0D00 00F3 E90D	38E8 7110 CCF3 E90C	38E5 7109 CCF3 81D0	44E5 7109 C0F4 4401
0DE0:	2C85 6178 8CF2 FFC0	2882 E108 CCF6 C03F	3110 0910 0663 FEBC	3D09 3579 DE63 FFFE
0DE4:	38F4 08A7 00F3 EF9A	3CE3 6305 84F3 E756	38E5 7111 CCF3 E79D	38E5 7010 CCF3 E40D
0DE8:	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C		

ADDRESS				
0E00:	38E5 7109 CCD3 81E1	38E0 0910 2AF6 E40D	38E0 0910 26F3 EA3C	3880 E178 8CF3 FFC0
0E04:	3CB5 7109 CCF4 440D	2FFF F910 FE73 E80D	28FF F910 FC90 2C0D	3885 7108 CCF6 C03F
0E08:	3875 7110 CCF3 E6BC	34E2 F110 CCF0 640D	44E2 F110 CCF4 640D	3D45 7111 CC75 270C
0E0C:	3CF5 7110 CCF3 E40C	0405 611C CCF0 A801	28E5 7129 CC33 F800	38E5 6109 CC92 CA4C
0E10:	2CA5 411C ECF0 0801	28E5 7110 CC13 E40C	28E9 3579 D173 FFD2	2C0B 1110 CCF0 A40D
0E14:	1CE8 B110 DC73 E40D	38E5 7110 CC73 E40D	3815 7110 CCF0 E40C	596F F979 F6F3 FFF9
0E18:	28B4 B110 CCF3 E72D	38E5 71EF CCF3 E406	28B5 728C FAF3 DC55	28B5 7110 CCF3 E40D
0E1C:	28E5 7110 CCF0 3C0D	38E5 7110 8CF3 E40D	2805 411C CCF3 CAD1	28E3 7010 CCF3 E40C
0E20:	28EA B110 2CF3 E40C	38E5 7510 C083 E40C	1CE9 3709 3213 C0C9	0405 711C CCF0 2C00
0E24:	38E5 6110 8C13 EC0C	28E5 7109 CCD1 C0C9	3815 70FF CCF3 E406	1CE5 628C BCF3 EC54
0E28:	38E5 6110 8263 E40D	34E5 7110 CC13 E40D	00E5 7110 CCD3 E40C	1805 70FF CCF3 E406
0E2C:	1CE5 628C BCF3 EC54	38E5 6110 82F3 E40D	38E5 7109 CCF3 81E0	3545 7111 CC05 270C
0E30:	3880 4978 00F3 FFC0	2C82 E108 8C56 C03F	38E5 6110 9CF3 E40D	3CE5 7311 F8F3 EABC
0E34:	28E5 7310 F8F3 E40D	28F5 7025 CCF4 65BE	38F0 0847 16F3 E9C7	14F5 70A7 CCF3 E9C8
0E38:	28F0 0910 04F3 EABD	38E4 E109 8CF4 0011	44F5 7308 FAF1 87FF	28B5 731C FA73 E408
0E3C:	2D63 7379 F8F3 FFF0	2CA4 B0C0 CCF3 E40C	38E3 5110 CCF3 E58D	40E5 7110 CC43 E40D
0E40:	28E2 D109 4CF3 C005	38B5 7110 CCF3 EA3D	3CE0 7B11 FE63 EEBC	28E5 7310 F8F3 E40D
0E44:	38E5 711C CCF4 2808	3CF5 411C 6CF0 0000	28B5 7110 CC13 E40C	410B 3509 D4D3 C004
0E48:	2CE2 D110 4CF3 E40C	38E5 7111 CCF3 EABD	38E5 7110 CCF3 E40C	3CF5 411C 6CF3 C000
0E4C:	38EA A510 93F3 E40C	08EB 3510 D733 E40D	0CE5 708F CCF3 E406	3885 708C CCF3 E858
0E50:	38B5 7149 CCF1 8000	40E5 7310 FAF3 E40D	44A4 D11C 4CF3 E40A	1853 6110 CCF3 D23D
0E54:	3913 411C 8CF3 E40A	38B3 411C 4CF3 E40B	38B3 4108 0CF3 C7FE	3CBF 8B10 00F3 E40D
0E58:	40BB 351D D6D3 EAD2	3905 7109 CCF3 C004	2CE2 D110 4CF3 E40C	38E5 7111 CCF3 EABD
0E5C:	38E5 7110 CCF3 E40C	3CF5 7110 CCF3 E40C	0405 611C ACF3 EED1	38E3 6110 8CF3 E40C
0E60:	0CE0 0B10 20F7 240C	08E0 4825 10F3 E9AF	3833 2108 8CF3 C01E	38BF F8CB FCF3 C01F
0E64:	38E5 6105 8CF3 EDAB	28B5 7110 CCF3 E40D	38E5 6113 8CF3 EC0D	38E5 62C5 B4F3 ED9A
0E68:	38E5 6110 8CF3 EC0D	38E5 6110 8CF3 EC0D	3CE3 6310 88F3 E40D	1510 0909 12F1 E001
0E6C:	38E4 E110 8CF3 E40C	38E3 5185 CCF3 E5D7	38E5 7010 CCF3 E79D	38E5 7109 CCF0 0007
0E70:	38E5 7010 CCF3 EF9C	38E5 7109 CCF0 0032	38E5 7010 CCF3 EF9C	38E5 7109 CCF0 0034
0E74:	38E5 7010 CCF3 EF9C	38EF F909 FEF0 0009	38E5 7010 CCF3 EB9D	394D 7511 DA55 270D
0E78:	38E0 5909 6EF3 C014	2962 F11D CCF3 EEAF	38E0 4909 3873 81DE	3CE8 B509 DA51 8013
0E7C:	38F1 3045 CCF3 E697	2CE5 7109 CCD4 0008	1C82 F178 CCF3 FFC1	2C80 4908 0037 003F
0E80:	3D05 7110 CCF3 E63C	0CA5 2109 8CF4 0010	3D03 711D CCF3 DA2F	38B3 7111 8CF3 E62C
0E84:	3CB3 7111 CCF3 E62C	3CB3 7111 CCF3 E62C	3CB3 7111 CCF3 E62C	3CB3 7111 CCF3 E62C
0E88:	3CE3 7109 CCF3 C7FE	3D03 70A7 CCF3 EE8E	38E0 0909 28F3 81E0	39C2 E110 9CF7 640D
0E8C:	3880 F108 7CF7 003E	3C85 7178 CCF3 FFC0	2CF0 4910 0AF3 E80C	2BF3 3110 CCF2 EC0D
0E90:	28AF F911 FCF3 E6CC	38A5 C910 0093 E6CC	3CF5 711C CCD1 BC01	34E5 7109 C092 CA4D
0E94:	2CA5 711C ACF1 5801	28F9 3510 D991 640D	2855 7110 CCD1 A40D	38F5 7029 CCF1 7FD2
0E98:	38F5 7111 CCF1 640C	39D5 7110 CCD7 240D	38E5 7110 8C43 CECD	38E5 71EF CCF3 E406
0E9C:	28B5 728C FAF3 D454	28B5 7110 CCF3 E40D	1065 711C 8CF4 7C00	0CA3 3110 DCF3 E6DC
0EA0:	44F3 7110 CCF3 E40C	2CE3 7110 CCF3 E40C	3CE3 7109 CCF4 0010	1CE4 F110 CCF3 E40C
0EA4:	40E3 7110 CCF3 E40C	38E5 7010 CCF3 E40D	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C

ADDRESS				
0F00:	38DD F434 C183 F036	38E3 F109 CCF3 8165	38E5 7110 CCF3 E40C	3885 7010 CCF1 E79C
0F04:	38E5 7110 CCF3 E40C	38E1 8909 04F8 6001	38A0 0910 A903 E67D	14E5 7104 CCF7 4C0E
0F08:	28E9 3559 D0B2 E000	3A15 7104 CCF2 EC83	80E5 7104 CE63 EC82	1450 190B 0053 C301
0F0C:	BCB5 70C5 CCF3 E41E	38A6 34B4 C023 C852	386C 0824 41D3 E69B	18E9 1109 8CFE 1200
0F10:	38E8 3129 CCF2 C010	38E5 7024 CCF7 672E	3861 F110 CDE7 65F5	28E5 7104 C3C3 E44A
0F14:	1C78 B022 2CF3 F00C	38E8 3110 CCF3 E65D	1CEB 7710 31A3 E40C	5D74 B111 DCF5 E49D
0F18:	24C3 7047 CCF3 F92D	38E5 7510 D143 E66D	5D70 B179 CCF5 FFFF	58EB 6710 8483 B08D
0F1C:	38F5 7510 D142 266D	38E5 7110 CCF3 E40C	74E0 0B09 1EF3 C010	2CA8 A228 88F3 C00C
0F20:	38E5 7505 C3E3 E4F8	38EF F904 FA93 DA67	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C

ADDRESS												
1000:	70E5	7109	C2F3	E00E	2D54	E110	8CF3	E40C	392F	F0D2	CCF3	ED9C	3956	7112	CCF3	F93D
1004:	38E0	3190	CCF6	E5FD	70E7	F310	F6F3	E5ED	74F5	7000	CC94	258D	38E5	7104	CCF3	C58B
1008:	38E5	7104	CCF3	C4EA	38E5	7105	CCF3	E4F8	38E5	7105	CCF3	E4F8	38E5	7104	CCF3	C95A
100C:	38E5	7104	CCF3	C286	38EE	F5A5	D5B3	E537	38EE	F510	D573	E40D	3940	0911	54F5	270D
1010:	4105	70D7	C2F3	E8F8	28E5	7109	C2F3	E02B	2D04	E11C	8CF3	E40F	38B0	307C	CCF3	E54F
1014:	40E5	7000	CC93	E40C	38E5	7105	CCF3	E4F8	38EE	F5A4	C3E3	C283	38E5	7105	CCF3	E666
1018:	38E5	7105	CCF3	E666	38E5	7104	CCF3	C80E	38E5	7104	CCF3	C80E	38E5	7104	CCF3	C80E
101C:	38E5	7104	CCF3	C80E	38EE	F5A5	D203	E4C3	38EE	F5A4	DFF3	E002	38E5	71B4	CCF3	E006
1020:	3976	7125	CCF2	F021	3905	70B0	CCF3	E40D	38E6	0113	02F2	FCBC	5402	6174	8C74	411A
1024:	38E5	71F5	CCF3	E49B	487B	56D7	F433	FD0E	3CE5	7110	CC93	DC0C	5CB3	E072	FCF3	BC0C
1028:	44D8	F100	CCF1	E40D	3888	5110	DCF1	E40C	38E0	6110	8CF1	DC0A	5CA5	607E	BCF3	BC04
102C:	38F5	7100	CCF4	6406	3885	7110	CCF4	6406	38E5	7110	CCF3	E40C	44E5	5210	C473	E40C
1030:	3976	7125	CCF2	F021	3905	70B0	CCF3	E40D	38E6	0113	02F2	FCBC	5402	6164	8C74	4127
1034:	38E5	71F5	CCF3	E4DA	487B	56D7	F433	FD0E	3CE3	F110	CC93	DC0D	5CB3	E072	FCF3	BC0C
1038:	38D5	7100	4CF1	E40D	3880	6110	9CD1	E40D	5CA5	607E	BCF3	BC04	44F8	F100	CCF1	A40D
103C:	3888	7110	9CF1	A40D	38E0	6110	8CF1	DC0A	5CA5	607E	BCF3	BC04	38F5	7100	CCF4	6406
1040:	3885	5110	86F4	6406	28E5	5310	C473	D80B	44E5	6210	C463	E40D	3875	7149	CCF1	C001
1044:	48B5	631E	DCF3	FC04	38E2	5000	CCF3	E40C	5D74	B111	DCF5	E49D	24C3	7047	CCF3	F944
1048:	1CE2	7000	CCF3	E40D	5CE4	B110	DCF3	E49D	24A3	7111	CCF5	E49D	24C3	7047	CCF3	F944
104C:	1CE2	7000	CCF3	E40D	60E5	7145	C4D3	E42C	38E0	0910	06F3	E40C	294B	7710	2081	E40C
1050:	60E5	7110	C8F4	0405	60B5	711D	CC14	6802	4C25	5118	EC22	FFFE	3825	7128	CCF3	C001
1054:	2CB5	7110	CCD3	E40D	42C8	B6C7	B673	ED9F	3935	7510	D643	FF0D	38ED	0525	4083	E42C
1058:	28E8	8710	2073	E4BD	38E6	7110	CCF3	E4BD	38E5	7110	CCF3	E40C	38ED	3525	C083	E42D
105C:	38E5	7129	CCF3	C000	04E5	6310	C8D3	E40C	60E5	7110	C8F4	00B5	38B5	411D	4C94	68B2
1060:	60A5	7118	C4F2	FFFE	38A5	7110	8CF3	E40C	42C8	A6C7	F673	E99E	3935	7110	CCF3	FF0C
1064:	4CB5	631E	DEF3	F406	38C5	7110	8CF5	240C	38A5	7010	CCFB	240D	38E0	5910	2473	E40D
1068:	38E5	7110	CCF3	E40C	3902	F025	CCF1	E510	38E8	B510	9783	DC0D	38B5	6111	FCF3	E6BC
106C:	40A5	7110	CCF3	E6BD	38F2	F0D7	DCF3	E9E3	2CEB	3509	D783	C525	2CF2	F110	CCF3	E40D
1070:	2CB8	B510	D726	E40D	3CEB	3510	C443	E40C	28A5	7110	CC73	E6BD	38E5	7110	CCF3	E40C
1074:	40F5	7110	C0F3	E40D	407B	6710	E443	EABD	38E2	F110	CCF6	E40C	38F5	7025	CCF1	E511
1078:	38D5	7110	8CF8	E40C	38A1	F110	CCD7	640C	3C72	F0F1	CCF3	DD90	2C70	E110	8C93	DD80
107C:	2CB2	F025	4CF3	E510	3CFB	3508	CC68	01FE	3CA5	7067	CC73	DD11	28D4	0D11	00F3	E6BC
1080:	28A3	7108	CCF3	DFFE	3CB7	F110	C0F3	EC00	38F5	6110	FC98	640C	38A5	53B0	F4F8	AC0D
1084:	38EB	2795	8443	E528	39D7	F110	8CFB	240D	38E5	7149	CCF3	E001	1E3E	E3AD	C8F3	DC03
1088:	38E8	A445	98F3	EE3E	34E5	7110	CCF3	FC0C	38B5	711E	CCF3	E807	4605	7110	CC86	240C
108C:	3A15	7110	CCF5	3F0D	3A15	7110	CCF4	E40C	3A28	7110	CCF3	240C	38E5	71A0	CCD3	E59D
1090:	3E37	F110	CCF2	E40C	5D74	B12E	DCF5	F098	24E3	7140	CCF3	E49D	38E3	6110	4CF3	FCAC
1094:	3973	6109	CCF5	C008	44E3	7110	CC35	AC0D	58E3	6110	ECF3	E46C	38EE	B549	C324	3E02
1098:	38E5	7559	C324	0003	38E5	7175	CC73	E481	6100	6B08	00F1	800F	60EF	9CC0	FEF4	68FE
109C:	38E8	A109	9CF4	4008	38E8	7110	CCF3	E40D	44EB	1310	88F3	E40D	38E5	7010	FC73	E40C
10A0:	38E0	0900	0A73	E5CD	34EA	F109	CCF1	C008	38E7	3104	CCF3	FC50	08E5	4309	04F1	4040
10A4:	5CE5	7025	CCD3	E8EF	38EF	7425	DF33	CAAB	3995	4110	C2F7	240C	5572	61F5	8C72	E6A8
10A8:	48AB	66D7	BF73	EFDF	38E7	C310	F6F8	1ECC	38E5	4025	42F3	C3F7	5572	61F5	8C73	E6B8
10AC:	48AB	66D7	BF73	EFDF	38E5	50B0	CD03	C59D	38EB	0510	9CD3	E58C	3CFE	71B0	C664	2406
10B0:	3CF5	7110	C464	2405	38E5	4110	3C03	E40C	38E5	4110	5C13	E40D	38E5	4110	BC23	E40C
10B4:	38E5	41C5	DC33	E75A	390B	04C7	1C03	DBEA	38EE	3425	DB93	CB2F	38EE	B510	DBB3	E58D
10B8:	38E5	7510	DB03	E5AC	80E2	71A0	DD03	E59D	28E6	B305	9CF3	E6BE	3A0A	3511	DBE8	26CC
10BC:	38BF	351E	DAF3	FDB5	38C5	751E	DAF3	FC06	38E5	71D0	CCF3	E40C	394B	3579	C1F5	3FFE
10C0:	38E5	5025	4C63	CBEE	38E5	71C5	CCF3	E7B6	38EE	B5B5	DC83	E712	38F5	71B0	CCF4	2406
10C4:	38F5	4110	BC24	25A4	38E5	41C5	DC33	E793	390B	04C7	9FB3	DBEA	38EA	35B5	DBE3	E713
10C8:	3A05	7111	CCF8	26CD	38BF	351E	DC33	FDB5	38C5	751E	DC33	FC06	38E5	71C5	CCF4	179A
10CC:	38EE	3595	DD43	E73E	38E5	71C5	CCF4	179A	3CEE	7110	C663	E40C	3CE6	3110	C463	E40D
10D0:	38E5	4110	3C03	E40C	38E5	41C5	5C13	E7B7	390B	04C7	1FB3	DBEB	38EA	3595	DBE3	E73F
10D4:	80E2	7110	DD03	E40D	28E6	7305	9CF3	E73B	18E5	5110	02F3	E65C	5572	61F5	8C73	E766
10D8:	48AB	66D7	BF73	EFDF	3C65	7030	CCF3	D26C	80E2	7100	CCF1	A806	38EF	3110	CCF1	A005
10DC:	38E8	3110	CCF3	E40C	40E5	5110	02F3	E65D	5572	61F5	8C73	E782	48AB	66D7	BF73	EFDF
10E0:	3D05	7030	CCF3	D26D	80E2	7100	CCF4	2807	38E5	7110	CCF4	2005	38E8	3110	CCF3	E40C

ADDRESS				
10E4:	38EF 3110 CCF3 E40D	38E8 3110 CCF3 E40C	40E5 5110 02F3 E65D	5572 61F5 8C73 E7A6
10E8:	48AB 66D7 BF73 EFDF	3D05 7030 CCF3 D26D	80E2 7100 CCF4 2807	38E5 7110 CCF4 2005
10EC:	38E8 3110 CCF3 E40C	18E5 5110 02F3 E65C	5572 61F5 8C73 E7C3	48AB 66D7 BF72 EFDF
10F0:	3C65 7030 CCF3 D26C	38E5 6110 8481 A807	80E2 7000 CCF3 E40D	38E5 7110 C2F2 F40D
10F4:	4D85 6110 8C77 240D	4CA5 631E 9EF3 F406	38E5 7510 DA93 E40C	5C75 7149 CCF1 C000
10F8:	48B5 631E DCF3 E804	38E2 5000 CCF3 E40C	390C 3112 CCF3 DB9C	38E1 3110 CCF3 E4AC
10FC:	38E5 7010 CCF3 E40D	38E5 51C5 4C63 E7B6	38E5 7510 DC33 E40C	38E0 0910 0873 E5BC
1100:	38E6 7104 CCF3 FC50	3805 4024 42F3 C4D2	5572 61F5 8C72 E413	48AB 66D7 B373 ECDF
1104:	38E5 4110 82F1 1E9D	54E2 61F5 8C73 E41F	48AB 66D7 B373 FCDE	3905 50B5 CC53 E436
1108:	5D95 7110 CCD6 E40C	48E5 7110 CC93 CDC3	5555 6110 CCF4 640D	48A5 6072 ACF3 FC0D
110C:	3915 7549 D104 4000	4D85 7110 CC96 E5DD	5D35 7110 CCD4 640C	38E5 7110 CCF2 F40C
1110:	38EA C11C 8CF3 C54E	38F6 3110 CCF1 A409	08EB 0710 5243 E40C	3CEE 7110 C673 E40D
1114:	3CE6 3025 C473 C0D3	38E5 4110 3C03 E40C	0CE5 5111 C2F0 440C	44E2 E1D6 8CF0 8867
1118:	34AC 230E 9CF3 FC05	34C5 731E 5EF3 FC06	3C35 70B0 CCF3 DC0C	38E5 6110 8671 E805
111C:	38E2 51A5 4CF3 E47E	38E6 9185 CA63 E48A	38E6 3105 CCF3 E453	3876 D11D 1C43 DAD2
1120:	38EA 3580 D323 EA9D	10E5 7305 9CF3 E453	38E2 5025 5C53 C0D3	28E6 7305 9CF3 E44F
1124:	3CEE 7110 C673 E40D	3CE5 7100 C473 E40C	38EB 0510 1341 9C0A	38E5 4110 3C03 E40C
1128:	08E5 5111 C2F0 440D	44E2 E1D6 8CF0 88AE	34AC 230E 9CF3 FC05	34C5 731E 5EF3 FC06
112C:	3C25 70B0 CCF3 DC0D	38EE A110 8671 E804	38E5 7510 D263 E5AC	3876 D11D 1C43 DAD2
1130:	38EA 3510 D323 EA9D	10E5 7305 9CF3 E49A	1145 7379 9CF3 FFFF	38BB 3510 C1F5 240C
1134:	38FE D1B0 1C43 E4E4	2865 731D 9CF3 EED2	38E5 7010 CCF3 E40D	3875 7149 CCF1 C001
1138:	48B5 631E DCF3 FC04	38E2 5000 CCF3 E40C	397D 3525 C082 F02D	38EB 0579 15E3 FFC7
113C:	2CE9 8510 15E3 C5C2	54E2 61F5 CC73 E4FF	48AB 66D7 BA53 FE96	38E5 5110 CC93 E40D
1140:	3CE7 3113 CCF3 81DC	38E7 C310 76F4 2808	3CEE F110 C673 E40C	3CE6 B085 C473 E54F
1144:	38E0 0919 40F3 E000	28E8 97A0 F4A3 EC5C	38E0 0905 60F4 2D0F	40E3 6310 84F3 E40C
1148:	34E2 7110 DC93 E40C	38A5 751E D423 FDB5	40EE D310 C4F3 E44C	38E3 6109 8CF4 4073
114C:	3CEE F110 C673 E40C	3CE6 B085 C473 E55B	3910 0910 60F3 E45D	287B 56D7 F5E3 ED7B
1150:	387E F510 D4D4 240B	34E2 7110 DC93 E40C	38A5 751E D4C3 FDB4	38E5 7109 CCF3 C040
1154:	38B5 70C2 CCF3 DC0C	38E5 7109 CCF1 C060	38E5 7105 CCF3 E78F	38E0 0909 60F3 C073
1158:	287B 56D7 F5E3 ED7B	3CE7 71A1 C493 E79D	00E8 C113 4CF3 E79C	40E3 5310 C4F3 E40D
115C:	00EA C110 4CF3 DC0A	34E3 6310 C4F3 E40D	74EA F110 CCF3 E4CC	38E3 B110 CCF3 E7ED
1160:	38EF 35C5 C6B3 E5B9	38E5 7010 CCF3 E40D	38E0 0910 0873 E40C	38EF F904 C693 FC50
1164:	38E7 4110 04D2 C9D4	38EB 2425 9A13 EE87	38E7 0110 0718 4807	28BB 76D7 BA13 FE86
1168:	34D9 B465 5A13 E287	8617 7112 AD08 3C0C	3875 7110 BCF1 A40C	28A5 4110 42F1 440D
116C:	54E2 61F5 8C73 E5BB	48AB 66D7 BA53 DA96	3CE5 70B0 CD33 D40D	38E5 6110 8671 E805
1170:	38E5 5105 CD43 E78E	38E8 8595 5743 E5DB	3CE5 7110 C4F3 DC0B	2CE3 6305 84F3 E5DB
1174:	3CE5 7110 C8F3 DC04	2CE3 6310 84F3 E40C	38EE B180 FD28 A40D	38E5 7113 CD28 A40C
1178:	38E5 411C C2F3 CD4E	54E2 61F5 8C73 E5EE	48AB 66D7 BA53 DA96	38E5 4110 42F1 5E9C
117C:	54E2 61F5 8C79 1DFB	48AB 66D7 BA53 DA96	38E8 C110 4CF1 1C0D	38F6 7110 CCF2 E586
1180:	38E0 0909 72D4 0061	38E5 5109 7C54 4040	8205 7110 CCF3 E40C	2CAB 66D7 BD47 EEA2
1184:	38E5 71C5 CCF1 FE6E	8215 7110 CC92 E40C	8605 6310 94F3 E40C	2CAB 70D2 6CF3 FC0C
1188:	30C5 7110 CC92 E40D	38E5 7110 CCF2 DC0C	38D5 70C5 CCF7 FEE2	8615 71C5 CCF3 E66F
118C:	2CAB 66D7 BD47 EEA2	38E7 71C5 CCF3 E66E	3A2E B445 D933 E65F	88E7 F110 CCF3 D69C
1190:	38E5 7105 CCF1 FF7B	38F5 7110 CCF3 E408	3875 771D F9B3 EED2	88E7 F111 CCF3 D69D
1194:	2CE5 5305 9CF1 FF7B	38F5 7110 CCF3 E40B	38B5 7510 D9B3 DEDF	38EE F511 D9B3 D69D
1198:	2CE5 5310 9CF3 FC0C	38F5 6110 CAF3 E405	38B5 711D CCF3 EAD2	9245 7110 CCF3 E69D
119C:	38B5 70C7 CCF3 EA7E	8CF5 7110 C8F3 E405	38B5 751D DA13 EAD3	90E4 7110 CCF3 E40C
11A0:	8CE3 7110 CCF3 E40C	74E8 F110 CCF3 E4EC	38E3 F110 CCF3 E7EC	38EF 35C5 C6E3 E5AD
11A4:	38E5 7010 CCF3 E40D	3875 7149 CCF1 C001	48B5 631E DCF3 D804	38E2 5000 CCF3 E40C
11A8:	10E5 5310 9CF3 E65D	39FF 3512 DB13 A40D	3CEE 7110 C673 E40D	3CE6 3085 C473 E773
11AC:	38E5 7110 CCF3 E45C	391E 74C7 DAB3 DEC3	38E2 5080 1C43 E59C	38A5 751E DAA3 D805
11B0:	38E7 3110 CCF3 E44D	3CEE 7110 C673 E40D	3CE6 3085 C473 E773	38E5 7110 CCF3 E40C
11B4:	1D0B 76D7 BDB3 DF6E	38EE 7510 DB23 E45D	38E2 5080 1C43 E59C	38A5 751E DB13 D805
11B8:	10E5 5310 9CF1 FE5C	39FF 3510 DC63 A40C	3CEE 7110 C673 E40D	3CE6 3085 C472 DF73
11BC:	38E5 7110 CCF2 EC5A	391E B4C7 DC13 DF16	390E 7510 DBB1 E5AD	38E2 5080 1C43 E59C
11C0:	38A5 751E DBA3 D804	3905 518D 5C53 FED3	28E6 D305 9CF3 E6EE	38E2 5080 1C43 E59C
11C4:	38A5 751E DBA3 D804	38E7 3110 CCF1 FC46	3CEE 7110 C673 E40D	3CE6 3085 C472 DF73

ADDRESS																
11C8	8CE5	7111	C8F3	D40C	90E8	A110	ECF3	E80C	2CE5	5310	C532	FC0B	1D0B	76D7	BDB3	DF6E
11CC	38EE	B580	DD03	E45C	38E6	B105	CCF3	E71F	38E2	5080	1C43	E59C	38A5	751E	DC63	D805
11D0	3875	518D	5C53	FED3	28E6	D305	9CF3	E71F	38E2	5080	1C43	E59C	38A5	751E	DC63	D805
11D4	10E5	5310	9CF1	FE5C	39FE	B510	DD83	A40C	40E5	7085	CC72	DF63	38E6	B110	CCF3	E45C
11D8	38E5	5085	CAF2	DF72	3906	D11D	5C53	EAD2	38A5	751E	DD63	D854	38E5	7110	CCF3	E5CD
11DC	38EC	3510	DDD3	E66C	38E8	3110	CCF3	E40C	38E0	0910	60F3	E40C	1C7B	21A9	ACF3	C021
11E0	2C78	3109	CC71	C032	38E5	71A9	CCF1	C0F6	38E8	3109	CCF1	C0FA	38E0	0910	80F3	E59C
11E4	1CEB	6110	ACF3	E5AD	38E8	3109	CCF3	C092	1C7B	6179	ECF3	FFE0	38E8	2110	CC73	E58C
11E8	3870	08C9	60F3	FF0A	38E5	6000	8CF1	E80D	3876	E0C9	8CF3	FF07	38E0	0800	A4F1	E80C
11EC	1C7B	6179	ACF3	FFCF	38E8	2110	CC73	E40C	38E6	7000	CCF3	E5AD	38E0	0910	80F3	E40D
11F0	1CEB	6110	ACF3	E40C	38E8	3109	CCF3	C092	1C7B	6179	ECF3	FFE0	38E8	2110	CC73	E40C
11F4	3870	08C9	60F3	FF0A	38E5	6000	8CF1	E80D	3875	60C9	8CF3	FF06	38E0	0800	A4F1	E80C
11F8	1C7B	6179	ACF3	FFCF	38E8	2110	CC73	E40C	38E5	7000	CCF3	E40C	38E0	0910	40F3	E59C
11FC	3876	B0C2	CCF3	E80D	38E0	0800	56F3	E40C	3876	30C2	CCF3	E80C	38E0	0800	5AF3	E40C
1200	3876	F0C2	CCF3	E80C	38E5	7000	CCF3	E40C	38E8	3110	CCF3	E5CD	38E0	0979	0674	7FC7
1204	38E5	7104	CCF3	FC50	5CE5	7025	CC63	C13E	3917	0085	82F3	C13F	54E2	61F5	8C73	E428
1208	48AB	66D7	B533	D94E	38E7	4110	42F1	5E9D	54E2	61F5	8C73	E432	48AB	66D7	B533	D94E
120C	38E8	C11C	4CF3	C94F	38F6	7110	CCF2	E586	38E5	7113	CCF1	540C	38E5	5109	CC44	0060
1210	38E0	0909	72D4	4041	34E5	7110	C2F3	E40D	38A5	7108	CCF3	C00E	38EB	2625	D233	EC83
1214	38EB	2710	92C3	E40C	38EB	2710	9347	C00D	38E5	7104	CCF3	C6E2	38E7	7105	CCF3	E52A
1218	3CEE	7180	C673	E40D	3CE5	7000	C473	E58D	38E2	5000	1C43	E59D	38EE	B110	CCF2	DC0D
121C	38E8	3110	CCF3	FDAB	3878	D11D	5C53	EED3	38A5	711E	CCF3	D805	38E5	7000	CCF3	E40C
1220	38E5	7513	D187	C00D	38E5	5104	CC33	C6E3	38E5	75C5	D283	E53F	38E5	7113	CCF7	C00D
1224	38E5	7104	CCF3	C6E2	10E5	53C5	9CF3	E53F	38E5	7510	D183	E40C	38E5	5110	CC33	E40D
1228	390B	0710	11B1	E40C	38E5	4110	CC78	276C	38E5	7104	CCF3	C7EF	38E5	75C5	D3F3	E53F
122C	38E0	0905	4033	E463	3837	10C7	CC33	DCBE	38E7	7109	CCF1	C061	38E5	7104	CCF3	C7BF
1230	1D0B	76D7	B4E3	DD3A	38E5	7513	D1B7	C00D	38E5	7104	CCF3	C6E2	38E5	75C5	D3D3	E53E
1234	38E5	7113	CCF7	C00D	38E5	7104	CCF3	C6E2	10E5	53C5	9CF3	E53F	38E0	0905	4033	E463
1238	383F	34C2	D3A3	DC0C	38E5	7109	CCF1	C060	38E5	5104	CC33	C7BE	1D0B	76D7	B4E3	DD3A
123C	38E5	7510	D1B3	E40C	38E5	4110	CC78	276C	38E5	7104	CCF3	C78F	3A07	7045	CCF3	E516
1240	38E5	7110	CCF3	D69D	38E5	7110	CCF1	FC0D	38E5	7104	CCF3	C77A	38F5	7110	CCF3	E408
1244	38B5	751D	D4F3	DED2	38E5	7111	CCF3	D69C	2CE5	5310	9CF1	FC0D	38E5	7104	CCF3	C77A
1248	38F5	7110	CCF3	E40B	38B5	7510	D4F3	DEDF	38EE	F511	D4F3	D69D	2CE5	5310	9CF3	FC0C
124C	38F5	6110	CAF3	E405	38B5	751D	D4F3	EAD3	38E5	7110	CCF3	E5CD	34EA	F110	CCF3	E4DC
1250	38E3	B110	CCF3	E7ED	38EF	35C5	C6E3	E5AD	38E5	7010	CCF3	E40D	3875	7149	CCF1	C001
1254	48B5	631E	DCF3	D804	38E2	5000	CCF3	E40C	38E0	0909	0878	0041	38E5	7104	CD03	FC51
1258	38EB	0509	15C8	4049	38E7	4110	4508	48B5	38EA	8110	4CF8	08B6	38E7	3105	CCF3	E42C
125C	38E5	7113	CCF0	27ED	3975	4111	82F1	A40C	54E2	61F5	8C73	E582	48AB	66D7	B533	D94E
1260	38E5	4110	42F1	5E9C	54E2	61F5	8C73	E58E	48AB	66D7	B533	D94E	38E5	511C	CC43	C94E
1264	38F8	C110	4CF2	E587	38E6	7113	CCF1	540C	5CE5	7104	CC63	C9E3	38E5	7105	CCF3	E60F
1268	38E5	7109	CCF3	C008	40E0	1B1E	E0F3	EC08	38E5	6023	8293	EC0D	38E5	6109	8CF3	E4E0
126C	2CE4	E110	8CF3	E40C	38E0	3110	CCF3	E40D	38E5	7110	CCF3	E40C	38E8	3110	CCF3	E40C
1270	34DA	F435	D8A3	E646	38E5	7510	D9E3	E40D	34DA	F435	D9E3	E6E6	34DA	F435	DB93	E70F
1274	34DA	F435	DC33	E737	38E5	7510	DD53	E40D	38E5	7510	DEA3	E40D	34DA	F434	DF03	CC3F
1278	74E8	F510	D7D3	CE53	4D85	6110	CC76	E40D	4CA5	631E	9EF3	F667	4CE5	7110	CD38	F40D
127C	38E0	D000	CD13	E40C	3995	4110	C2F6	E40D	54E2	61F5	8C73	E603	48AB	66D7	B533	D94E
1280	48E5	7110	CD33	E66D	38E0	D110	CD18	D80D	38E2	5000	0CF3	E40C	38E8	C0B0	DC33	CC0C
1284	84F0	F000	DD14	2404	30F5	7110	CC74	2406	763A	F110	CCF3	E40C	8C78	131E	DCF3	EC04
1288	8C75	531E	DCF3	E004	38E5	7000	CCF3	E40C	3905	7108	CCF4	001F	38B5	7024	CCF3	CA0F
128C	3905	7110	CCF3	E40C	38B5	7024	CCF3	CC6F	38E5	7104	CCF3	CC1B	38E5	7104	CCF3	CC2B
1290	3905	7579	D8D4	3FFE	3900	0908	4DD4	001F	38B5	7024	CCF3	CA0F	3905	7110	CCF3	E40C
1294	38B5	7024	CCF3	CC6F	38E5	7104	CCF3	CC1B	38E0	0909	82F3	C0B5	2875	5077	ECF3	EE6E
1298	38E0	0909	C2F3	C0F5	2875	5077	ECF3	EE6E	1CE8	B710	B9D3	E40D	38E5	7104	CCF3	CC2B
129C	3905	7579	D944	3FFF	38E5	7104	CCF3	CC96	3900	0908	4154	001F	38B0	0824	6163	CA0E
12A0	3905	7110	CCF3	E40C	38BF	3424	DB33	CC6F	38E5	7104	CCF3	CC1B	94D8	96B0	FA93	C002
12A4	9A05	7110	CC71	E40C	38E5	7104	CCF3	CC2B	3905	7579	DA14	3FFF	38B5	7024	CCF3	CC6F
12A8	38E5	7104	CCF3	CC1B	98D8	96B0	FAD3	C003	3A05	7110	CCF1	E40D	38E5	7104	CCF3	CC2B

ADDRESS												
12AC:	3905	7579	DA74	3FFF	38E0	0910	08F3	E40D	34E8	A710	BB43	E40D	3A15	5110	CCD1	E40C
12B0:	38E5	7104	CCF3	CC2B	38E5	7105	8C73	E6D2	38B5	7024	CCF3	CC6F	38E5	7104	CCF3	CC1B
12B4:	38E7	3109	CCF3	C073	987B	56D7	FB83	EEE2	38E5	7104	CCF3	CC2B	3905	7579	DB24	3FFE
12B8:	38E5	7104	CCF3	CC96	3905	7108	CCF4	001F	38B5	7024	CCF3	CA0F	3905	7110	CCF4	640C
12BC:	38E5	7104	CCF3	CA0F	3608	F510	D8F1	E40C	38EF	7110	CCF3	E40C	4115	7110	CC73	E40C
12C0:	38B5	7024	CCF3	CC6F	38E5	7104	CCF3	CC2B	3915	7579	DC04	7FFE	3905	7108	CCF4	001F
12C4:	38B5	7024	CCF3	CA0F	3905	7110	CCF4	640C	38B5	7024	CCF3	CC6F	38E5	7104	CCF3	CA0F
12C8:	3608	F510	DCA1	E40D	38EF	7110	CCF3	E40C	40E5	7110	CC73	E40D	38E5	7104	CCF3	CC2B
12CC:	3915	7579	DC64	7FFE	3900	1908	0094	001E	38B5	7024	CCF3	CA0F	3905	7110	CCF3	E40C
12D0:	38DF	742D	DD43	EC04	3905	717A	CCF4	3E00	38E5	4110	FCF3	E40C	40E5	6104	8C33	CC6A
12D4:	38E5	7104	CCF3	CC6F	3900	1908	0094	001E	34B5	7024	CCD3	CA0E	3905	7024	CCF3	CC6F
12D8:	5CD5	702D	DC63	EC04	3908	F57A	9E54	3E00	38EE	F511	DDE3	D69C	2CE5	5310	98F3	FC0D
12DC:	38F5	6110	CAF3	E405	38B5	711D	CCF3	EAD2	4105	4112	A2F0	880C	54E2	61F6	8C73	E786
12E0:	48AB	66D7	B533	D94E	38E2	5110	0CF3	DE9C	38E5	511C	FC53	C94F	84F0	F110	CCF2	E406
12E4:	5CE5	7104	CC63	CC6E	4105	4112	62F0	440C	54E2	61F5	8C73	E7A2	48AB	66D7	B533	D94E
12E8:	38E8	C110	4CF1	1D8D	5CE6	7104	CC63	CC6E	3905	7108	CCF3	C01E	34BA	F509	DED4	0040
12EC:	3A0F	7110	CCF1	E40D	40E5	7110	CC73	E40D	38E5	7104	CCF3	CC2B	38E5	7104	CCF3	CC6F
12F0:	6100	2B08	00D4	401E	38B5	6024	CC93	CC6E	38B8	B425	9F83	E7DE	38E5	7104	CCF3	CA0F
12F4:	3905	7110	CCF1	E40D	38E5	7104	CCF3	CC2B	3915	7579	DF24	7FFF	34E5	4024	CC33	CC6A
12F8:	34E5	4024	CC33	C9E2	3915	7110	CCF3	E40D	38B5	7024	CCF3	CC6F	38E5	7104	CCF3	CA0F
12FC:	3905	7110	CCF1	E40D	38E5	7104	CCF3	CC2B	3915	7579	DFA4	7FFE	38E5	7110	CCF3	E40C
1300:	38E5	7175	CCF3	E460	3803	F110	CCF1	E79D	38E8	7110	CCF3	E40D	38ED	B575	C183	E46C
1304:	3813	F110	CCF1	E79C	3803	F510	C071	A78D	3C45	718E	C673	D804	38E5	4000	5C13	E58D
1308:	3C45	711E	C473	D994	38E2	5000	1C43	E40C	38EE	8110	9C22	DC0C	38E8	3110	CCF3	FDAB
130C:	3875	511D	5C53	EED2	28E6	D310	9CF3	E40C	38E5	7000	CCF3	E40C	40E0	0B10	1E93	E40D
1310:	38E5	6109	82F3	E62A	2CE4	E109	8CF3	C013	38D0	30D7	CCF3	EC6F	38E5	7110	CCF3	E40C
1314:	38E8	3110	CCF3	E40C	34DA	F435	D263	E4B7	34DA	F435	D303	E4C7	34DA	F435	D323	E4F6
1318:	34DA	F435	D3F3	E51B	34DA	F435	D4B3	E536	38E5	7104	CCF3	C9E2	38EA	3510	D1D3	E40D
131C:	38E5	7104	CCF3	C99F	38EE	F511	D213	D69C	2CE5	5310	9CF3	FC0C	38F5	6110	CAF3	E405
1320:	38B5	711D	CCF3	EAD2	80EF	7149	C473	C000	2E05	6310	8473	E4FB	3A15	711D	CCF0	2CF2
1324:	38E5	7513	C1F1	A40C	38E5	7110	CCF3	E5CD	38EE	F511	D2A3	D69D	2CE5	5310	9CF3	FC0C
1328:	38F5	6110	CAF3	E405	38B5	711D	CCF3	EAD2	38E5	7110	CC03	E4ED	38E5	71C5	CCF3	E5AC
132C:	38E5	7010	CCF3	E40D	3A1F	3510	D2F1	E40C	38E5	7104	CCF3	CC2B	38E5	7104	CCF3	CC6F
1330:	38E7	3104	CCF3	CC6F	38E7	7104	CCF3	CC6E	38E0	0904	4093	CC1A	38DF	34C2	D353	DC0D
1334:	38E5	7109	CCF1	C060	38E0	0909	6093	C073	287B	58D7	F253	EC97	34E5	5109	EC93	C20D
1338:	60EB	2310	C8F3	E5AD	38E6	F104	CCF3	C77B	38E7	C304	B6F3	CC2B	38EF	3510	D1B3	E40D
133C:	38D5	74C5	D1B3	E4C3	38E5	7104	CCF3	CA0F	3905	7104	CCF8	0C6E	60E0	2B04	00D3	CA0F
1340:	3908	B508	9444	401E	3905	7108	CCF3	C1E0	38E0	091E	08F3	EC08	38EB	2110	BCF4	6C02
1344:	3915	7109	CCF8	4041	38E5	7104	CCF3	CC6F	60E0	2B04	0093	CA0E	3905	7104	CCF8	4A0F
1348:	38D5	7020	CCF3	E40D	3905	7110	CCF8	640C	38E5	7104	CCF3	CC6F	38E5	7104	CCF3	CA0F
134C:	40E5	7104	CD03	CC6E	80E5	7104	CCD3	CA0E	3905	7108	BD03	C101	38EB	2420	9533	EC0C
1350:	3905	717A	CCF4	3E00	38E5	7110	CCF3	E40C	40E5	4104	FC33	CC6A	38E5	7104	CCF3	CC6F
1354:	38E0	B3F5	0096	E528	70AB	3510	D623	E6B6	2CE2	F179	CCF3	FFDD	74FA	F510	D5F3	EEBC
1358:	3CA0	F030	CCF3	E5C3	68F2	2A11	0093	E6DC	70CB	6025	ECF3	E729	38E1	3513	E073	EEDC
135C:	68DA	F0C0	CEF3	E79C	38F3	7045	CCF3	E679	38E1	3010	DCF3	E40C	3CF4	F02D	CCF3	EB93
1360:	34E3	7011	CAF3	E79C	3CE2	F510	C9E3	E5CC	059D	3525	C084	242C	39A7	F82D	FEF3	E800
1364:	38E2	0001	00F3	E90C	40B5	711D	CC13	E902	5415	711D	CC05	4009	38C8	7108	CCF6	41FF
1368:	74EA	F4F5	D6B3	E528	68E8	F529	D6C1	C001	60B5	6307	C4B3	E41C	60E5	6305	C8B3	E41D
136C:	60E5	5305	C4B3	E610	5D74	B12E	DCF2	F099	24B3	7140	CCF5	E49D	3945	6179	CCF5	3FFF
1370:	70E5	6110	AC93	F8AC	54B5	62A7	9CF3	FD48	39A5	7110	CCF6	240C	3584	3110	CCF3	E40D
1374:	38B5	7510	C066	240D	5D74	B12E	DCF2	F099	24B3	7140	CCF5	E49D	70E8	B113	CCF1	FC0C
1378:	2CA2	B51D	F7B4	3CAD	58B9	2647	E493	F948	3905	7510	C075	E40C	3D65	5247	E093	FD25
137C:	58A3	2647	A493	F948	38D5	7510	C075	E40C	5D74	B12E	DCF5	F098	24E3	7140	CCF3	E49D
1380:	74E7	F110	CA91	38AD	5CA9	2628	E523	E001	5CE0	B0F5	CC73	E529	35A7	F1A5	C068	665A
1384:	6155	7110	CD68	A40C	1997	F1A5	C068	E663	2927	F1A5	C0F9	266F	18E7	D1A5	8069	7A76
1388:	2977	F1A5	C06A	6683	3995	61A5	4D72	E692	1969	1510	9A61	640C	7478	F510	DA85	E40C
138C:	58B9	2647	FAB3	FAAB	5615	70CD	CCF3	FC00	4955	6072	ACF3	F9ED	6167	F100	C0F3	E40C

ADDRESS																
1390	4959	2647	FAB3	EEAF	4925	5112	2CF3	F80C	3A17	F9E8	FEF6	81FF	38A2	0901	00F3	E50D
1394	68BA	F11D	CCF3	E902	38EE	8410	C843	E40D	58E9	1713	C491	DEAC	38F5	7000	CCF8	640D
1398	38F0	9110	FCF5	640D	58EA	A113	BCF1	EAAC	18F5	7408	DB06	41FF	5959	1710	DB03	E40C
139C	38F0	9000	FC74	AC0C	5959	1710	DAF3	E40D	38F0	9110	FC72	AC0C	18B5	7000	CCF1	240C
13A0	58F9	1710	DAE4	240C	60BA	B118	CCF4	5E00	40AC	3168	CCB3	E1FE	44E5	6203	C4B1	DEAC
13A4	58F9	1710	DAD3	E40D	2CE5	7003	CCC1	DEAD	3950	9110	FC73	E40C	58F9	1600	DAC5	AC0C
13A8	1559	1710	DAC3	E40C	38F5	7000	CCF5	EC0C	38E5	7110	CCF3	E5CD	3A95	7110	CCF5	E40C
13AC	3855	7110	CCF5	A40C	9CE5	7110	CCC3	E40D	9A55	7110	CCB2	A40D	3A45	7110	CCF4	A40D
13B0	3A35	7110	CCF6	640D	3A25	75D5	C525	6525	7577	F310	F6F1	F00C	28B7	F3A7	F692	E6FA
13B4	2967	F3A5	F691	E6FB	38A5	61BC	4CF4	654E	6155	71D5	CC91	EAFB	4557	F3A5	F691	EAFB
13B8	28E7	F3A5	F691	FAFA	712B	35A5	DCA1	E6FA	34D7	F228	F6F3	C007	54EE	B4F5	DC53	E528
13BC	75A9	F510	DBE1	E80D	38E5	7010	CCF3	E40D	5D54	B172	DCF1	DC6C	34A5	7002	CCF3	F88C
13C0	1CB3	711C	CCF3	E55D	3578	31E6	CCF5	E70B	5D75	70AE	CC74	3C98	24E3	7021	CCF3	EC9C
13C4	3905	7512	C525	DCAC	5D94	B160	DCF1	E46D	24E3	7110	CCF1	E88D	24E3	7161	CCF5	EC6C
13C8	54E5	7003	CC05	DC8C	5578	3047	CCF3	FB0A	5CE2	B110	CCF3	B00D	396D	F4D7	C073	E81D
13CC	3CE5	7113	CC85	E08C	2CE2	B510	DCB3	E44D	3815	7110	CCF0	E40C	3835	7110	CCF0	679C
13D0	3805	7110	CCF0	A40C	3828	7110	CCF0	278C	0015	7110	CC13	E40C	2CE1	3010	CC03	E40D
13D4	00E5	7110	CCC3	E40D	38E1	2010	4C03	E40D	3805	7110	CCF0	640C	0425	7105	CC23	E74E
13D8	04E5	7110	CCC3	E40C	38E5	6010	4C13	E40C	38E5	7110	CC93	E40C	70E5	6510	1E02	E40D
13DC	28B5	728C	F8F3	E456	30C1	3110	CC23	E40D	38B5	7110	8CF0	678D	38A8	7110	CCF0	278D
13E0	00A5	7108	CCD3	807E	0818	308E	CCF3	E459	70E5	6510	1E03	E40C	28B5	728C	FAF3	E457
13E4	30C1	3105	CC23	E77B	0025	7108	CCD4	07FF	04B8	8269	04F4	7C01	40EB	3510	DEE2	E58D
13E8	38E5	7109	CC91	C201	441E	711E	CCF3	E408	28B9	931C	F8F0	6446	3090	4D10	00F3	E40C
13EC	2CA5	6110	0CC0	A79D	38E5	7010	8C03	E40C	38E5	7509	DEC3	C055	3805	7105	CCF3	E74E
13F0	0405	751C	CF53	C3C0	3805	7510	C073	E7AD	74E5	7309	F694	0005	60AA	B4A4	C4A3	FC01
13F4	390D	3526	C083	E42D	2D0B	76D7	0083	F02D	3815	6108	5CC6	C1FF	00E2	C110	CCF3	E40D
13F8	38E8	A110	7CC4	02BC	38E1	3711	CC73	C2BC	3CFB	4710	9FD6	E40C	3CFA	3510	DFE0	240C
13FC	2DB2	C51D	DF80	680A	38E1	301C	DCF3	E001	3905	7510	C1F0	240C	38E5	7110	CCF3	E40C
1400	74E0	0B04	E0F3	DB0B	74E0	0B04	C0F3	D05B	38E5	7104	CCF3	D19A	74E0	0B04	C128	96E3
1404	74E0	0B04	C148	9803	74E0	0B04	8168	9403	74E0	0B04	8168	9403	74E0	0B04	8168	945A
1408	74E0	0B04	8168	960E	74E0	0B04	8168	960E	74E0	0B04	8168	960E	74E0	0B04	8168	9C02
140C	70EB	3578	C3E3	7FE7	38A6	F0A5	D0F3	E0F9	28E9	B727	0DF2	E821	38EE	F545	C083	E42D
1410	70E2	891C	00F1	7D6E	38E4	6110	BCF8	240C	38E0	3110	CD08	67ED	3827	7110	CD1E	A40D
1414	38E5	7110	CDAA	640C	60E7	F009	CCF4	3FC6	04E5	7105	CC63	E420	4109	8465	9953	C258
1418	002B	3425	D533	E54F	555F	F0D2	CCF3	F99D	3926	7112	CCF3	E93D	08E5	7104	CC73	D15B
141C	38E5	41F9	3CF3	8006	00E5	6110	8C73	E5CD	40E2	7104	8C53	D17B	3C27	71C7	C4F8	2487
1420	28EC	2109	8073	8002	28DC	21E0	AC72	FE9C	30E5	7103	4072	EE9D	00F1	3100	ED04	65C5
1424	38C5	7110	CCD4	65D6	1C9A	0D10	0043	E553	44EF	F979	0674	7F8D	28A0	0824	0463	D107
1428	3841	3168	4CF1	1FFE	1860	0824	50F1	D107	28A0	08A5	12F1	E653	28A0	08A4	12F1	D0CB
142C	28A0	08A4	0EF1	D0CF	28A5	70A5	CCF1	E653	2840	097A	14F1	2001	28A0	08A4	20F1	D107
1430	38A0	08C5	08F1	E653	10A5	7104	CEF1	911B	3841	310A	FCF1	2001	3870	0911	0863	E40C
1434	38B5	7104	CCF1	D0E7	38FE	F5C0	D421	E5D7	38C0	0900	2091	E5C4	38EF	F913	A0F2	FE9C
1438	44A9	94D5	D941	DD17	10E5	6110	CEF3	E40C	28A5	611C	80F3	E850	28B5	611C	8043	EC01
143C	1E05	7226	84D8	2537	8097	F1F0	DD03	E552	14E4	7104	CCF3	D0D6	10E3	7110	CCF1	25ED
1440	3862	5104	7C53	90D7	3870	0904	0863	D0E7	38FE	F5C0	D461	E5D6	10C5	7100	CEF1	E5C4
1444	38E5	7112	CCF2	FE9C	10D5	70C5	CEF3	DE52	28A5	611C	80F3	E850	28B5	611C	8043	EC01
1448	8207	F026	DD08	2537	389F	F9F0	C093	E552	14E4	7104	CCF3	D10A	10E3	7110	CCF1	25ED
144C	3862	5104	7C53	910B	1095	70B4	CEF3	D142	28A5	611C	80F3	E850	28B5	611C	8043	EC01
1450	3C55	7080	C6F3	E71C	3848	B100	8CF3	EADE	3841	3110	CCF3	E6DC	74E0	0B10	20F3	E4DC
1454	38A5	7020	CCF3	E57C	38E5	7510	C073	E4DD	0C3E	7110	C273	E55C	5C77	31F0	CCF4	2553
1458	54BF	6111	ACF4	255D	28E7	9110	FCF3	E40D	0D01	F110	C2F4	2402	5575	611D	8CD3	EC0F
145C	42A5	631E	9CF3	EC05	38E5	7000	CCF3	E40C	1865	7110	C273	E55D	5C75	71F0	CCF2	E553
1460	54BF	6111	ACF2	E55C	28E7	9110	FCF3	E40D	18E1	F110	C2F2	FC02	5575	611D	8C93	EC0E
1464	48A5	631F	9CF3	EC05	38E5	7000	CCF3	E40C	050B	3465	D913	C65E	555F	F0D2	CCF3	F99D
1468	0026	7112	ECF3	E93C	38A5	411D	7CF0	C809	28C5	4104	4C74	515A	3800	7110	8CF1	A40C
146C	04E5	7104	CC73	517B	38D5	7111	CC38	261C	38EC	E919	0138	E060	34B0	7110	CCF1	240D
1470	38F5	70BD	CCF3	C9B9	08CA	F110	CCF3	E5A8	2CE0	0D09	6043	8009	3800	08B0	0263	E59C

ADDRESS				
1474:	38FF F900 ED01 E40B	38C4 7100 8CF1 E40D	04E8 F509 D7A3 8004	1875 7110 DCF1 E45E
1478:	189C 3512 F7C3 E93D	1C90 0B12 1EF3 E80D	3875 711C C0F3 DC01	28B5 611C 8091 EC01
147C:	8039 A51D 9960 E853	111E F026 4E44 660F	2893 71B0 CEF1 65AC	3A35 7110 CCF1 25BD
1480:	14E7 9194 ACF3 D1EA	3E06 7111 CC78 261C	38E5 7509 D7A3 8008	4070 7178 8CF1 E000
1484:	38E1 0179 4CF3 E000	7470 0A27 80F3 EE37	2878 B56B D8D1 6001	10E3 2913 0043 E79C
1488:	7450 0A24 40F3 D22E	8CA1 10C4 0C43 D236	28E2 0919 0043 E201	3448 B4D2 D8D3 EC0C
148C:	1046 8919 0041 3601	38E5 71A4 CCF4 523F	10F5 7110 C644 640A	3835 70C0 CCF3 E40D
1490:	44E3 5110 0EF3 E79D	74E0 0B10 20F3 E4CD	38A5 7020 CCF3 E57C	38E5 7510 C073 E4DD
1494:	38E6 7104 CCF3 DB66	38E6 3104 CCF3 DB67	38E5 7100 CCF3 E59C	60E4 0B10 00F3 E58D
1498:	28EB 3510 D9A3 E57D	38E5 7585 C6C3 E5B8	60E1 0D05 00B3 E647	98E5 7110 CCF3 E65D
149C:	A0E5 7110 CCF4 680D	9CA5 7104 CCD2 D3EA	2CE5 7110 CCD4 680D	A4E5 7110 CCF8 3C0D
14A0:	34A5 7104 CD02 D3EB	28E5 7109 CCD3 803C	3515 7104 CC62 D3DF	80B5 6310 94D7 640C
14A4:	2C65 7310 5592 E40D	3A05 70D4 4D84 53EB	28E5 7109 CCD3 803C	3515 7104 CC62 D3DF
14A8:	28B5 7110 CD69 A40D	2CE5 7109 CD73 801D	18E5 7104 CCD2 D3DE	98B5 7110 CCF9 266D
14AC:	A0A5 6310 D499 640D	B4A5 73D0 5159 E40C	30E7 F200 F6F3 E40C	39D5 7104 CCFA 177F
14B0:	94E5 7110 CCF3 E40C	9CA5 7110 CCF4 640C	98A5 7104 CCD2 D3EB	2CA5 7110 CCD4 640C
14B4:	A0E5 7110 CCF1 BC0D	34A5 7104 CD72 D3EA	2CA5 7110 CCD4 640C	A4E5 7110 CCF3 E40C
14B8:	34A5 7104 CD82 D3EA	28E5 7109 CCD3 803C	3515 7104 CD92 D3DE	18B5 6310 9567 25CD
14BC:	38E5 7104 4C53 D26E	3A55 7110 CCF8 E40D	28E5 7110 CD43 E40D	38D5 71E5 CCF9 675B
14C0:	95C5 6310 D161 A5DC	3265 7110 CD39 240C	2CA5 7110 CD58 A40C	15D5 7110 CD89 A40D
14C4:	2CA5 7110 CD79 E40D	1865 7104 CD9A 177F	90E5 7110 CCF3 E40D	98A5 7110 CCF4 85ED
14C8:	94A5 7104 CCD2 D3EB	2CE5 7110 CCD4 680D	9CE5 7110 CCF3 FC0D	2CA5 7110 CD52 E40D
14CC:	34E5 7104 CD63 D3EB	2CA5 7110 CCD4 640C	A0E5 7110 CCF3 E40D	34A5 7104 CD72 D3EA
14D0:	2CA5 7110 CCD4 640C	A4E5 7110 CCF3 E40C	34A5 7104 CD82 D3EA	28E5 7109 CCD3 803C
14D4:	3515 7104 CD92 D3DE	28B5 7105 CC74 26C2	A3F5 7109 CCF3 8101	9A65 6310 9488 A44D
14D8:	2CA5 7310 5488 644D	39D5 7110 4C89 244C	2CA5 7110 CC88 E44C	15C5 7110 CC89 A44C
14DC:	2CE5 7110 CC89 684D	1DD5 7110 CC8A 244C	2CA5 7110 CC89 E40C	3805 7035 CCF3 E7CB
14E0:	38E5 7104 CCF3 D77E	90A5 7104 CCD2 D3EA	2CA5 7110 CCD4 640C	98E5 7110 CCF3 FC0C
14E4:	2CA5 7110 CD42 E40C	34E5 7104 CD53 D3EB	2CA5 7110 CCD4 640C	9CE5 7110 CCF3 E40D
14E8:	34A5 7104 CD62 D3EB	2CA5 7110 CCD4 640C	A0E5 7110 CCFF E40D	34A5 7104 CD72 D3EA
14EC:	2CA5 7110 CCD4 640C	A4E5 7110 CCF3 E40C	34A5 7104 CD82 D3EA	28E5 7109 CCD3 803C
14F0:	3515 7104 CD92 D3DE	38B5 60D5 8DAB E71A	AAF5 7110 CD81 A40C	38A0 0910 0A79 E40C
14F4:	2065 7110 CCFA 245D	38A7 9185 FC72 27D3	18E5 7104 CD93 D77E	38E5 711E CCF3 E408
14F8:	28B5 628C A4F3 EC55	38E5 6000 8CF3 EC0D	3916 71FF CC93 E406	BEE5 628C BCF3 EC55
14FC:	38E5 600C 82F4 6C00	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C	

ADDRESS				
1500:	38A5 74C5 C089 64F9	38E0 0904 3E99 DDD3	34E5 6304 48CA 1DDE	3865 51A4 CDBA DE22
1504:	38E5 6104 4DFA 5E3A	7408 F510 D111 A7CC	B4E5 7104 CCFA 9E63	AC65 7104 CCDF 5E82
1508:	BCD5 70B9 CC73 C005	B4A5 7110 CCF1 E15D	38A5 70BD CCF3 E1CB	3875 711D CDE1 E15F
150C:	38E5 7104 FC63 DAD3	38EA 951C D37B 1D4F	389B 35E5 D783 999B	38E5 7510 D443 E40C
1510:	22D5 7106 CDE3 A51B	60EE 7B04 FEB3 DF73	BCE5 5025 CC91 9C1A	1CA9 06A7 B5B3 C81B
1514:	18D5 411D ADF3 DBDF	28E5 6105 ECC3 E41B	38A5 74C5 C089 64F9	38E0 0904 3E99 DDD3
1518:	34E5 6304 48CA 1DDE	1C65 71A4 CDBA DE23	38E5 6104 4DFA 5E3A	B405 7104 CCF1 9E63
151C:	ACE5 7104 CCD3 DE82	BCD5 70B9 CC73 FFFC	B4A5 7110 CCF1 E15D	38E0 08B0 04F3 E80C
1520:	3875 7110 CDE1 E95C	39B0 090B 0A63 8110	38EA 951C D27B 1D4E	3800 19E5 12F4 252B
1524:	39B5 71F7 CCF3 A973	38E5 7105 CCF3 E4CB	39B5 750B D353 8110	3806 51A5 C3F4 24AA
1528:	3AC5 74B5 D783 E4B3	38E5 7105 CCF3 E4B3	BEC8 F4B5 D7E3 E59A	1CEB 1710 B263 E40D
152C:	BCE6 7109 C273 8113	39B0 1911 1291 A40D	38EB 1513 D321 DC5C	38E7 9113 BC63 E44C
1530:	2095 7185 CC83 ACBE	38D5 7105 CCF3 A4B7	74E0 0B09 1EF3 C019	2CEB 2709 B813 C00B
1534:	2DBB 370B 7813 8112	3AB6 7104 CCF4 5EDA	38E5 7104 CCF3 DF62	3BD6 D1A5 C3F4 24EB
1538:	3AC5 74B5 D5C3 E4F2	38E5 7105 CCF3 E4F2	BEC8 F4B5 D623 E52A	1ADB 1710 F103 A40D
153C:	BED5 7110 C273 A5DC	28EB 3510 D443 E40D	39B5 7110 CCF1 A44C	207E B025 CCF3 E513
1540:	B8E5 6310 85E3 E40D	38E7 9110 BC63 E45D	2095 7186 CC83 A506	3AD6 D105 FC73 A4FA
1544:	74E0 0B09 1EF3 C019	2DBB 270B B813 8113	B8EB 3110 CCF3 E40D	3AB7 3104 CCF4 5EDA
1548:	38EF 7109 CCF3 C800	60E7 6304 C4B3 DF62	A4E5 7179 CC91 B000	9CE5 7179 CCD3 E001
154C:	227B 2310 C8F2 E40D	9657 302F CCF3 E407	28B5 728C F953 E457	2CA5 7108 CD64 401E
1550:	38E5 7110 8D73 FC0D	3905 60A9 CD83 FFE0	3645 7000 CD93 E40C	2E35 7110 CC92 E40D
1554:	8E05 7110 CCD3 D93C	8611 F02F CCF3 E407	28B5 728C F913 E456	2CE5 7110 CD23 E40C
1558:	38E5 7110 8D33 FC0C	2D15 70B0 CD43 E40D	34E5 7200 C553 E40C	60E0 8D04 E4B3 DF3B
155C:	21B5 7111 C8D1 A65D	2085 7110 CAF3 E406	2CE5 6311 8483 E450	38A7 911D BC82 2C0B
1560:	389B 3185 8CF3 A176	38E7 3105 CCF3 E606	BCE5 7110 C273 E65D	38EC 3509 D861 9FFE
1564:	21BB 1311 89E1 A5AD	3877 3106 CCF1 E507	A4E5 7110 CC93 E40C	9E75 7110 CCD2 E40C
1568:	9655 702F CCF3 E407	28B5 728C FB53 E456	28A5 6310 D163 E54C	32DE F310 9173 A40C
156C:	30E1 FB10 FF53 FC0D	3905 60A9 CD83 FFDF	3645 7000 CD93 E40C	2E35 7110 CC92 E40D
1570:	8CE5 7110 CCD3 E40D	8615 702F CCF3 E407	28B5 728C FB13 E457	2ED5 7110 CD23 A40C
1574:	38E5 7110 8D33 FC0C	2CE5 70C0 CD43 E40C	34E5 7000 CD53 E40D	38E5 7104 CCF3 DF3B
1578:	21B5 7111 CCD1 E65D	3885 7110 CCF4 2446	38E5 7113 8AD1 DC5C	2105 711D CCD2 2843
157C:	2CE5 70A5 CC83 DDE6	38E5 7105 CCF3 E606	BCE5 7110 C2F3 E65C	38AC 3511 D4A1 E40D
1580:	38E5 7509 D2D3 8113	38E0 08C0 0A63 E66D	38E5 7000 CCF3 E40C	38A5 74C5 C089 64F9
1584:	38E5 7104 CDE9 DDD2	38E5 7104 CCFA 1DDE	1C65 71A4 CDBA DE23	00E5 7104 CC6A 5E3A
1588:	B4E5 7104 CC9A 9E63	B2B5 7104 CC43 DE8E	AFF5 7104 CCDF 9E82	B7EB 1710 38E2 E79D
158C:	B2D8 F510 D5C3 A40C	B2DA F579 D783 BFFE	77FA F110 C2F3 FD4C	2CE7 F109 FCF3 8131
1590:	75B5 71B0 CC91 E40D	38EE B510 DA21 BC4D	2095 7179 CCD3 BFE0	3880 0910 0443 E40C
1594:	28B7 D110 EC73 E40C	2087 B31C A483 EC40	3895 6109 CC83 801C	38EE B509 D9C1 401D
1598:	2095 7179 CCD3 BFE0	1087 D110 EC73 E40D	2081 B31C A483 EC40	2C95 71B5 CC83 9663
159C:	3ADE 7179 CCF3 BFFE	38D7 30B5 CCF3 E6C6	22D5 7110 CDE3 A40C	3AB5 7104 CCF4 5EDA
15A0:	38EF 3510 D773 E57D	38E5 7104 CCF3 DF62	2096 F179 CCD3 BFE1	38E7 D109 FC72 C01E
15A4:	2097 B3A5 A883 AEA7	38E5 7110 CCF3 E40C	2095 7179 CCD3 BFE0	38E7 D110 FC73 E40C
15A8:	2091 B3BE A883 BE9B	75BA F510 DB11 E5EC	3AD5 7185 CCF3 A67F	19B1 3110 FCF3 AC0C
15AC:	38E5 5110 FC73 E40C	20E7 B310 E883 E45D	38E5 7113 FCF1 DD4C	2091 B3B5 E883 AABB
15B0:	3AD5 7105 CCF3 A67E	39B5 71E1 CCD2 E40C	39B5 7184 FC73 DF63	3ADE 35F5 DB53 AED7
15B4:	18E1 3104 FCF3 DF63	20E5 7313 84D2 FD3C	3898 A185 8CF3 9ED6	38E1 3104 CCF3 DF62
15B8:	38A5 74C5 C089 24F8	080F F8D9 C673 FFF2	38E6 7104 CD33 DB67	3876 30A4 CD53 DB67
15BC:	3806 7024 CD63 D14E	3827 3024 CD93 D14F	38E5 7104 CCF9 5DDF	3860 B9A4 EBFA DE22
15C0:	38E5 4104 0C76 DE3A	38E7 4109 3C93 8112	55B2 410B 4CDF C113	3522 611E 8CD3 E9D5
15C4:	3CA7 F11E CC93 D005	38E5 5025 FC73 DF22	3CE5 7110 CC83 E45C	38A2 7105 8CD2 2717
15C8:	B405 7104 CCF1 9E63	3401 0312 04F2 E44C	00EF F979 F87B 8200	20E8 B510 DCD3 E44D
15CC:	38E5 7426 DCB2 FF92	38EB 0713 1D43 E5FC	38B9 1425 DED4 6B97	9665 71A4 CC91 9B36
15D0:	3875 5024 FCFF D26E	38A7 E224 B6F3 D2C3	8CE5 71A4 CCF3 D31A	94A7 3104 CCF4 5387
15D4:	A4E7 7309 E0D3 801D	B5E7 F304 F6F6 D3DE	28BE B110 CCF3 E40C	28B5 742F DDA3 E407
15D8:	28B5 708C C183 E450	2FF5 7105 CD93 A772	28B5 728C F793 E453	2DB5 710B CD83 8113
15DC:	3AB7 70D4 CDE4 5EDB	38EF 3510 DBA3 E57C	38E5 7104 CCF3 D14E	3820 19A4 1474 152B
15E0:	A5B5 710B CCF3 8113	28A5 71A5 CAFA 2773	28E5 6104 8063 D59A	19B5 5107 81A3 9F73

ADDRESS				
15E4:	39B1 350B DDC3 8114	A285 731C E0D2 E400	B4E7 F304 F8F3 D3EA	28E5 7109 CCD3 803C
15E8:	3515 7104 CC82 D3DF	2CA5 7110 CD99 E40C	2CE5 7109 CD83 801D	18E5 7104 CCD2 D3DE
15EC:	2CE5 7105 CD73 E77F	9CE5 71A4 CC83 DB37	A285 7110 CCD2 E40C	3865 70D4 CD54 53EA
15F0:	28E5 7109 CCD3 803C	34E5 7110 CC84 7C0C	B517 F304 F8F2 D3DF	2CA5 7110 CD99 E40C
15F4:	2CE5 7109 CD83 803C	4485 7104 CCD2 D3DF	2CA5 7110 CD79 840C	2CA5 7105 CD89 A77E
15F8:	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C		

ADDRESS																
1600:	38A5	70C5	CD88	E4F9	5CE5	7109	CC47	5FFF	38E5	4104	42F9	5E02	1CE5	7104	CDB9	9E52
1604:	4007	8025	0CF3	E6FF	54E5	4184	8C89	DB56	380F	FA4B	ED73	C008	0067	F105	4CF4	2437
1608:	3860	0BA6	2534	2437	00BA	B709	70D4	0004	2C00	084B	04F3	C039	38A5	7105	CD64	2436
160C:	38B5	7109	CD24	0005	60E5	7109	CC4C	005E	4125	531E	1CF3	D804	3845	7044	CCF3	DE22
1610:	ADC5	7104	CCD6	DE82	54A4	41B0	8CFB	A40D	B0E7	F305	F8F3	E45A	00AF	F910	C643	E55C
1614:	6005	71E4	C3E4	159A	00EF	F9F4	DA7C	D52B	A407	704B	CCF3	C009	3A75	7105	CCD1	A873
1618:	3804	E84B	2063	C012	2CE7	B105	ECFC	E497	3959	1509	D40F	C008	3845	7445	DB33	E5D8
161C:	1865	731C	F6F3	E40D	28B5	731C	F6F3	E40C	28B5	731C	F8F3	E40C	28BE	B31C	F693	E40D
1620:	2D55	7110	CC93	CB1C	54E4	4110	8CF3	E40C	34E6	F105	CCFC	E583	2EE3	60B0	8CF3	E40F
1624:	38A5	7512	DBD3	E6DC	6265	71A5	C3E4	24AF	2E75	702F	CCF3	E5D6	28B5	708C	C184	6450
1628:	4285	702F	CCF3	E406	28B5	728C	F4F3	E45C	2910	0B05	1E93	ECC2	A285	702F	CCF3	E5D7
162C:	28B5	708C	C2F4	6451	9E75	702F	CCF3	E407	28B5	708C	C293	E451	2D15	7110	CD83	E40D
1630:	38E5	6195	CD94	64D6	34DB	3579	D353	BFFE	38E5	708E	CCF2	E458	28B5	708C	CCF3	D850
1634:	2CA5	7110	CCD4	640C	A0E5	7105	CC93	E583	38B5	7110	8CF3	E80D	44B5	711C	CC43	E800
1638:	38B4	E905	2091	A5CB	A4E5	7105	CC93	E582	38B5	7110	8C91	A80D	B8E7	F1C5	CEF3	E55A
163C:	3955	7111	CCF3	CB1D	44D5	711C	CCF3	D801	28B5	731C	E0F4	26D0	28E3	7105	CCF3	E6F7
1640:	95B5	71E6	CC91	A512	90E5	7109	CC93	8111	38E5	7023	FCF1	984C	2095	7105	CC83	A90B
1644:	34E5	7510	D804	640C	38B5	70C5	CCD1	A9C6	9B05	7105	CC93	A582	38B5	7110	8CF3	E80D
1648:	44B5	711C	CC44	2801	2CBB	2425	94C2	E533	38E0	1969	D0D2	E94B	3902	0905	0091	A5AB
164C:	2CD5	7110	CD19	240C	38E5	7110	8CF4	3C0C	9CE5	7105	CC97	E982	38B0	B905	EA43	E9B6
1650:	3BF5	7110	8CF3	CB1C	3A45	711C	CCF9	2801	3915	511C	8CF3	E801	3905	611C	8CF8	EC00
1654:	85F5	711C	CD18	A9C5	38E5	7105	CCD4	64D7	44D5	711C	CCF3	D801	2A45	711C	CCF1	AED1
1658:	3A35	711C	CCF4	6800	8623	711C	CC94	2801	2FF3	7110	CCF3	CB1D	44E3	71B5	CCF3	E6F6
165C:	4467	B11C	EC63	E6D0	5554	4111	8CF3	CB1D	3501	B11C	FC93	E6D1	2863	7510	DBD3	E6DC
1660:	38A8	A178	8CF3	E000	3B38	211D	C2FC	E803	38E5	6110	82F3	E801	2CD5	6108	82F3	DE01
1664:	2CB5	6110	82F3	E80D	38B5	6110	82F3	E40D	2CD5	6108	82F3	C1E1	2CB5	6110	82F3	E80D
1668:	38B5	6110	82F3	E40D	2CD8	2108	82F3	C01F	38E5	711E	CCF3	E408	28B5	528C	3AF3	D855
166C:	30C5	7000	CC84	640C	38E5	6179	CC91	8200	38E5	70EF	CCD2	E407	28B5	528C	3AF3	D855
1670:	28B5	7000	CC84	640D	38E4	E910	2091	240C	38E5	70EF	CCD2	E407	28B5	528C	3AF3	D855
1674:	28B5	7000	CC64	640D	8DB5	71E6	CC91	A5E7	88E5	7109	CC93	8111	38E5	7023	FCF1	984C
1678:	2095	7105	CC83	A9DF	34E5	7104	CCF4	5983	2CD5	60C4	8C83	D9C6	90E5	7104	CC93	D982
167C:	38B6	3110	8CF3	E80C	44B5	711C	CC44	2801	2CEB	2309	84F2	CBEA	38E8	7109	CCD3	803E
1680:	39DC	2984	0091	99AA	32E5	7110	CD03	E55F	44E5	7180	CD18	E40C	38E5	7110	8CF8	FC0C
1684:	94A5	7104	CC97	D983	2CE5	6104	8C83	D9C7	98E5	7104	CC93	D983	38B5	7110	8CF3	E80D
1688:	44B5	711C	CCF3	E801	2A35	711C	CCF1	AC00	85F7	B11C	CD11	2801	38E2	0969	0092	E94B
168C:	3B00	1904	D0D3	99AA	2CD5	7110	CD38	A40C	38EE	3110	8CF9	7C0C	90AC	3110	CC49	240D
1690:	80E5	7110	DC47	980C	84E5	7169	CC62	E94B	3BF0	1910	D0D3	CB1C	3B02	0904	0093	99AA
1694:	35E5	7110	CCF1	640D	8E4B	711C	CCF9	2800	14E5	7110	DC53	E80C	38B5	7110	8C53	E40D
1698:	14B5	711C	CCF8	FC00	4455	711C	CC56	E800	9CE5	7104	CC93	D982	2CE5	6104	8C83	D9C7
169C:	A0E5	7104	CC93	D982	38B5	7110	8CF3	E80D	44B5	711C	CC43	E800	38B4	E904	2091	99CA
16A0:	A835	7104	CC94	1982	38B5	7110	8CF3	E80D	44BB	711C	CC43	E801	18B5	7110	DC83	E40C
16A4:	126B	711C	CCF1	2C01	18A5	7110	DC83	E40D	38E5	7110	CCF3	ECOD	1A55	711C	CCF1	EC01
16A8:	4247	B11C	CD04	6AD1	38E5	7190	DCF4	280D	15B3	711C	CC46	E801	1075	7110	CC93	E6DC
16AC:	39B5	7110	CCF3	E40D	2CD3	7110	CCF3	E6DC	40E3	71F5	CCF3	E6F6	3955	7510	DB73	CB1C
16B0:	4443	7110	CCF3	E6DD	1DB3	7110	CCF3	E6DD	3503	7510	DBD3	E6DC	38E5	7310	CCF3	E40D
16B4:	3AD5	7110	CCF3	A40C	3BD5	7000	CCF4	240D	38E5	7110	CCF3	E40C	1CD7	B11C	EC74	2400
16B8:	4111	B11C	FCF4	6400	40A5	7110	CC94	240C	11B1	B11C	FC46	E401	1AB1	B11C	FC6A	E401
16BC:	54E0	0800	0EF3	E40D	3B35	70C0	CCF3	E40D	38E5	7110	CCF3	E79C	74E0	0B10	20F3	E4DC
16C0:	28EB	3110	CCF3	E57D	38E3	B105	CCF3	E41C	38A5	74C5	C083	E4F8	38A6	4110	8C92	E40D
16C4:	38E5	71EF	CCD3	E7C7	28B5	428C	7AF3	C055	28B1	3110	CC90	A40C	2CEB	2310	84F0	3D7C
16C8:	60D0	8C20	00B0	E40D	38E5	7105	8C13	E41C	3821	30CD	CCF3	E802	3805	7025	CCF3	E41D
16CC:	38E1	3105	DCF3	E41D	A287	B11C	ED84	2401	2E55	7110	CD93	E40D	2C61	B11C	FD61	A401
16D0:	2E35	7110	CD73	E40C	2E41	B11C	FD43	E400	2C05	70A9	CD53	FFF3	3905	7000	CCFA	240C
16D4:	8CE1	B000	FD33	E40C	60E4	0B10	00F3	E57D	38A5	7025	CCF3	E763	38E5	7595	C8E3	E5B1
16D8:	60E1	0D05	00B3	E6F6	60E4	0B10	00F3	E57D	38AF	3425	C6A3	E777	70E0	0F85	04F3	E5B8
16DC:	38A5	7425	C6D3	E5AD	60E1	0D04	00B3	D14E	3A15	70C8	CCF3	FFE1	38E5	7000	CCF3	E40C
16E0:	ABA5	7104	CCB0	9F3A	38E5	7110	CCF3	E40C	38E5	7110	CCF3	E40C				

ADDRESS												
1700:	38A5	74C5	C089	64F9	38E5	7104	CCF9	DDD2	38E5	7104	CCFA	1DDE	3865	51A4	CDBA	DE22
1704:	61E5	7104	CDAD	9E3B	3AB5	7110	CCFD	240C	B5F5	7110	CCFD	E40C	38A0	0904	40DD	5F72
1708:	1C75	7104	ACD1	1FBA	1047	D020	ECF3	E5DD	08E0	08C0	06F3	C80D	28A9	3710	BCE3	D1C2
170C:	2CA5	7110	C291	640C	B1EE	F110	CCD6	E40C	AEB5	7510	D613	E40D	38D5	5179	32FB	FFFE
1710:	38C5	80AB	83E3	C010	38C5	7179	CCFB	FFFC	ADC5	7104	CCD6	DE82	B047	F3C5	F664	245B
1714:	3BF6	F1A4	CDFF	959B	3865	70B4	CCF3	D5FA	B0E5	7109	CCDB	8186	98E5	7109	CCF3	8180
1718:	9EB5	711D	CCF2	2848	A0C5	711D	CCF2	2848	A4C7	711D	CCF2	2848	B0C6	711D	CCD2	2809
171C:	ACC5	7510	D614	240C	41C5	7104	CCD6	DE83	3BE5	71C5	CCF3	E4F7	B3F7	F311	F863	EF9C
1720:	38A6	F1A4	CDF4	159A	3865	70B4	CCF3	D5FA	3AF0	0911	2691	A40C	38EF	F909	FCD3	8112
1724:	38E7	D110	BC82	240D	2095	73A9	C883	8024	30D5	7107	CC83	AC93	39C0	1905	30F1	E59A
1728:	39C0	1905	4EF1	E59A	39C0	1905	86F1	E59A	3AE5	7105	DC93	A51A	38A5	7104	CCF4	159B
172C:	3AE5	7105	DC93	A51A	38A5	7104	CCF4	159B	3AE5	7105	DC93	A51A	38A5	7104	CCF4	159B
1730:	3AE5	7105	DC93	A51A	8C55	7024	CC53	DB7A	9F55	711C	CCF4	654F	A4E5	7110	CC69	A80D
1734:	8455	7110	CC78	A40D	386E	B510	D38A	240D	39B5	710B	CCF3	8113	3875	7104	CCF8	1572
1738:	45B5	730B	F5D3	8112	38A5	7110	CDE0	A40C	3B47	7104	CCF4	5EDA	38EF	3510	DCE3	E40C
173C:	38E5	7104	CCF3	DF62	3861	3110	CCF7	A40D	3B75	7110	CCF7	E40D	38E5	7110	CCF6	E40C
1740:	38E5	7109	CCF3	8113	3B50	091C	1064	640F	3895	7110	CCF8	240C	38E5	7110	CC82	245C
1744:	38E8	9513	9431	980C	3A05	75D5	D383	A69B	BAF5	7111	CCD6	E65C	348F	F825	FE94	2542
1748:	3880	1910	3674	255C	36F5	71E5	CCF4	655E	3900	1910	4E74	2553	3AF5	71E5	CCF4	655E
174C:	3900	1910	6674	2553	3AF5	71E5	CCF4	655E	3900	1910	7E74	2553	36F5	71E5	CCF4	655E
1750:	38A7	F109	BCD3	8187	890E	F510	D472	2453	90A5	7110	CCF8	666D	94A5	7110	CCF8	E40C
1754:	98A5	7110	CCF9	240D	A0A5	7110	CCF9	640D	08A5	5000	0CF9	E40C	34E6	7110	CDB3	9E5D
1758:	34EF	F913	FC92	E451	20D5	7109	CC63	8110	2075	7110	CCF1	FC5D	2115	6310	94F4	7D4C
175C:	20A1	B31C	6483	D840	3875	61B5	CC83	A563	38E6	7110	CCF3	E66D	38E5	7110	CCF3	E40C
1760:	ACE7	7000	CC93	E5BC	2CA5	7110	CDBA	E40D	09C5	7110	CC47	A40D	B5E5	5310	15C7	240C
1764:	39D5	7110	4C27	E40C	39F8	3110	8DD7	640D	38A0	0910	1693	A65C	1F77	F110	9C77	E45C
1768:	2095	7109	CC63	8016	20E5	7179	CCF2	FFE5	2867	A31C	A483	D840	3895	61A6	CC83	BDC6
176C:	38E0	0913	169F	6450	2095	7110	CC63	A80C	23D5	7110	CCF1	DD4C	2861	A31C	A483	D840
1770:	3895	61B6	CC83	BDB3	38D5	7110	CCF3	AE6D	38E6	F110	CC83	E44D	3B65	7000	CC87	A59D
1774:	4109	8465	9D33	C34E	3805	7024	CD23	DF63	3828	3024	CD33	DF63	3975	4110	C2F1	240D
1778:	54E5	6110	8C63	CD5D	48EB	61F0	ACF7	8953	18E9	9111	2CF7	AD5C	18E5	5110	4C68	E5FD
177C:	39E5	7110	CD47	A402	3865	7110	CD83	EC0C	48BF	531E	9CF3	D004	3978	0110	42F1	240C
1780:	54E5	6110	8C73	C55D	48EB	61F0	ACF7	0153	1CE9	9111	2CF7	2D5C	1CE5	5110	4C79	25FC
1784:	39C5	7110	CD57	2403	3875	7110	CD93	EC0C	4885	531E	DCF3	D005	38E5	7000	CCF3	E40C
1788:	38E5	4110	4262	CE53	392F	B510	D8D3	E40D	5559	A6D7	D8D3	FA36	38E5	70C0	CCF3	E66D
178C:	38E5	7000	CCF3	E40C	5555	5105	8DBA	FE2E	38E7	C309	B697	DFFE	38E1	E9B9	01D7	5FFE
1790:	34EA	C113	CCF7	E40C	B4E5	70B0	CFD3	CC0D	39FF	7110	CD77	E407	38E8	3110	CCF9	A40D
1794:	38E1	E8AC	01C3	C00E	2CEA	C113	4CF7	640C	38E0	08B0	1FC3	C40D	39D8	3110	CCF7	6407
1798:	B0EA	F0B0	CCF6	EA5D	38EC	1100	8274	4802	38E5	6111	9C74	6C0C	B1BA	F035	CCF3	E676
179C:	38E5	5110	DC73	E40D	3915	7110	CDA6	E662	3915	5042	C2F3	DC0C	38E8	3110	CCF6	E80C
17A0:	B1C5	7110	CC93	E65D	2CE0	7109	8C73	8113	29D7	B105	ACF4	289A	39E5	6110	CCD1	EE9C
17A4:	2DF7	A110	ACF4	265C	38EC	3509	D413	8133	39BF	F90B	E08B	4113	34E8	F113	CCF2	E40D
17A8:	38F5	511D	8A64	6805	1CB2	70C8	CCF3	C01E	3911	3110	DC84	840F	3CE5	5310	886F	E80D
17AC:	3915	52CB	B6F3	C01A	38A5	7110	CCFF	E79D	38EE	3513	9B21	DE1D	18E5	7310	E483	E85D
17B0:	1878	A712	BCF3	ED3D	3C65	7105	CC82	26BA	4095	5310	8883	BE6C	28A6	7310	E4F2	245C
17B4:	38B5	70C7	CCF2	2B3F	60E8	3110	CCF3	E40D	B9B6	7110	CCD3	E65C	45E2	7027	6CF1	AEF2
17B8:	38E5	70A5	CCF3	EEEB	389C	3510	DBC3	ACOD	20E5	7223	85E1	984D	38E5	6105	8CD3	E6EA
17BC:	21F5	7113	CC93	E59D	3DF5	631D	C891	E805	38D5	711D	CCF3	EC0F	B8EB	2310	C5EB	E40D
17C0:	39E7	3110	FCF2	E5BC	B4B7	F245	F6F1	AB16	36F3	531D	C4FB	DC42	38E5	7113	CC91	993D
17C4:	20E5	7185	CC73	E70B	391E	F1A0	CCF3	FE1C	1C7F	FB78	E071	E000	60E9	F109	C2F1	801B
17C8:	3AF0	08CD	186B	DC02	2CEE	B109	CCD1	8019	3CE6	71A0	C493	E66C	38E8	B110	8DF1	980E
17CC:	38D1	311D	CCF3	DDB3	3865	700D	CCF3	EED2	38E7	3105	CCF3	E752	70E0	0F10	0A63	E40C
17D0:	38A5	70CB	CCF3	C012	38A6	7110	CCF0	240D	3867	7424	DD43	DB56	38E7	7110	CCF3	E58D
17D4:	60E4	0B10	00F3	E57D	38AF	3425	C6A3	E75F	38E5	7595	C6E3	E5B1	60E1	0D10	00B3	E40C
17D8:	74E0	0B10	60F3	E57C	74A0	0A25	40F3	E41C	38A5	7020	CCF3	E4DD	38E5	7510	C073	E4DD
17DC:	ACE2	7030	CCF1	C25C	38E7	C311	76F1	EC0D	39D5	71B0	CCF3	E40D	3C76	E309	C891	C004
17E0:	39C5	6110	8CF3	E40C	38AB	24C5	DE53	E797	ACE2	7110	DDB3	E40D	1CEF	F979	F877	7FFF

ADDRESS				
17E4:	3DC5 6305 C897 2386	38D8 9028 CCF3 E001	38E5 7505 DEC3 E7B2	38D8 9028 FC73 DE01
17E8:	38E5 7505 DEC3 E7B2	38D8 9028 FC73 C1E1	38E5 7505 DEC3 E7B2	38E5 5110 FC73 E40C
17EC:	38EE F0C0 CD63 E66C	38E6 D000 CC23 E40D	3AB8 C110 8C73 E61D	39F5 6110 9C73 E65D
17F0:	38E6 A0B0 CCF3 CC0C	3C75 6309 8891 C004	39E5 6110 8CF3 E40D	2CEB 34C5 DE53 EB97
17F4:	3AB5 7111 CCFA E61C	1CEF F979 F877 FFFE	3DE5 6305 C897 A3CE	38E5 7110 CCF3 E40C
17F8:	38E5 7110 CCF3 E40C			

ADDRESS				
1800:	38E6 7110 CCF3 E40C	397D 3525 C082 F02D	38E5 7179 CCF3 FFC6	2CE9 8510 1123 C003
1804:	2CB7 011C 42F3 C54E	54E2 61F5 8C73 E423	48A9 A062 ACF3 FC0C	38E5 7504 D073 C10F
1808:	3CEE B109 CC93 C01F	3CB5 71AD CE91 E804	38CE 351D D0E1 E805	34E0 0D09 1E73 C01A
180C:	2CE8 9111 ECF3 E79C	38E1 3105 FCF3 E44A	34EF FB09 E093 C01A	2CE8 9113 ECF3 E79D
1810:	38E1 30C0 DCF3 E80D	34D0 0D0A 1A71 C019	390E B0B0 CCF3 E4CC	38E5 5110 CE73 E4CD
1814:	38EF 7510 C6E3 E7ED	38E3 5010 CCF3 E40C	38E5 7110 CCF3 E40C	38E5 7110 CCF3 E40C

READER COMMENT SHEET

Computer Systems Division

HP 3000 Series 64/68/70 Computer Systems Microcode Manual

30140-90045 Oct 1986

We welcome your evaluation of this manual. Your comments and suggestions help us to improve our publications. Please explain your answers under Comments, below, and use additional pages if necessary.

Is this manual technically accurate? ☐ Yes ☐ No

Are the concepts and wording easy to understand? ☐ Yes ☐ No

Is the format of this manual convenient in size, arrangement, and readability? ☐ Yes ☐ No

Comments:

This form requires no postage stamp if mailed in the U.S. For locations outside the U.S., your local HP representative will ensure that your comments are forwarded.

FROM: Date _____

Name _____

Company _____

Address _____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 718 CUPERTINO, CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Publications Manager
Hewlett-Packard Company
Computer Systems Division
19447 Pruneridge Avenue
Cupertino, California 95014

FOLD FOLD