User's Manual

STAND-ALONE OPERATING SYSTEM

093-000062-04

Ordering No. 093-000062

Printed in the United States of America
Rev. 04, July 1974

NOTICE

Original Release	-	October 1971
First Revision	-	July 1972
Second Revision	-	February 1973
Third Revision	-	June 1973
Fourth Revision	-	July 1974

This revision of the Stand-alone Operating System User's Manual, 093-000062-04, supersedes 093-000062-03 and constitutes a major revision to the manual. A chapter on CLI commands has been added and the system utility programs are described in greater detail in this revision.

PREFACE

Data General's Stand-alone Operating System (SOS) performs I/O for programs that operate in a non-disk environment. This environment may consist only of paper tape or may also include cassettes and/or magnetic tape. Included in the SOS Package are the following utility programs: Relocatable Loader, Extended Assembler, Symbolic Text Editor, SYSGEN, and Library File Editor. Systems with cassette or magnetic tape are supplied with a Command Line Interpreter (CLI) and a Core Image Loader/Writer. Optionally, a FORTRAN IV compiler is supplied with the system.

The user communicates with SOS through system command words built into his assembly language program and through the utility programs. For cassette or magnetic tape systems, the CLI utility may be used to perform file maintenance functions as well as to invoke other system utilities or user programs.

The current version of SOS has a number of new capabilities:

1. Support of a Real Time Clock.

   - The driver allows selection of 10Hz, 100Hz, 1000Hz, 60Hz, or 50Hz line frequency.

   - The clock may be accessed via system commands to set the time of day, get the time of day, set the date, get the date, get the current clock frequency, or suspend a program for a given period.

   - The clock may be accessed via CLI commands to set the time of day, set the date, or get both time of day and date.

2. Line Printer support with optional (80 or 132) column size.

3. Support of a second teletypewriter keyboard and teletypewriter printer.

4. New User Device Driver Links

   - System commands that identify a user interrupt device and remove it from the system.

   - Five external normals are declared in the SOS MAIN program and may optionally be resolved by the user as starting addresses of user DCT's. Two of the devices will occupy the highest SOS priorities in the interrupt search chain and the other three will occupy the lowest SOS priority. (The user may, of course, still elect to write his own code that will place devices at different priority levels.)

4. (Continued)

- SOS file/device names and SOS Channel Numbers are available for these user added device handlers.

5. The CLI has been substantially modified and many new features added in this version of SOS.

- Initializing and releasing magnetic tapes or cassettes.
- Copying files or tape reels and transferring files from one device/file to another.
- Producing dump format files and reloading dumped files.
- Making files in absolute or save file formats.
- Concatenating files.
- Comparing files.
- Starting programs in the SOS Debugger.
- Setting or getting the current date and time.
- The CLI BOOT command can be used to invoke either a system utility or a user file.

6. A number of less extensive changes were made in SOS or utilities under SOS to correct problems or to improve the system:

- System commands .EOPEN and .ROPEN, were added for RDOS compatibility but are treated as identical to .OPEN.
- The .SYSI initialization function is performed for the user if he starts his program at 377.
- When a CTRL A or CTRL C character is detected from the $TTI and the corresponding UST enable/disable location is disabled, the character is ignored instead of being passed on to the user program.
- The RDOS to SOS interface program is now named RDSI.
- The Relocatable Loader resolves all undefined external symbols to -1, compatible with the RDOS loader. Also, the insertion of a JMP @405 is is no longer made at location 377.

6. (Continued)

- The Core Image Loader/Writer may be installed directly on any magnetic tape or cassette reel, without first "MKSAVing" the file. It now starts or halts programs that it has loaded based on the TTY console command rather than the setting of DATA SWITCH ZERO on the Nova master console.

- In configuring a master reel of utilities, the configured Relocatable Loader now occupies File 1 and can then be used in the remainder of the configuring procedures.

TABLE OF CONTENTS

PREFACE

CHAPTER 1 - INTRODUCTION TO SOS

Introductory Concepts 1-1
RDOS to SOS Interface Program 1-2
Communicating with SOS 1-3
Generating SOS Systems 1-4
SOS Organization 1-5

CHAPTER 2 - SOS FILES AND DEVICES

Specifying SOS Devices 2-1
Loading SOS Routines and Device Drivers 2-2
Cassette and Magnetic Tape Files 2-4
Physical Characteristics of Cassette and Magnetic Tapes 2-4
Opening Magnetic Tape and Cassette Files 2-5

CHAPTER 3 - SOS UTILITY PROGRAMS

Introduction 3-1
Paper Tape Operation 3-1
Cassette/Magnetic Tape Operation 3-2
System Console Breaks 3-2
Core Image Loader/Writer 3-4
Installation Procedure 3-4
Bootstrap Procedure 3-4
Core Image Loader Operation 3-5
Core Image Writer Operation 3-6
Command Line Interpreter (CLI) 3-7
CLI Definition 3-7
R(eady) Message 3-7
CLI Activation 3-7
Symbols and Conventions Used in Command Line Syntax 3-8
Command Lines 3-9
Stacking Commands on a Command Line. 3-10
Long Command Lines 3-10
Switches 3-11

CHAPTER 3 - SOS UTILITY PROGRAMS (Continued)

Effect of Switches on a Command Line 3-12
Comma/Parenthesis Convention 3-13
Messages Concerning I/O 3-14
Error Messages 3-15
CLI Commands 3-17
APPEND 3-18
BOOT 3-19
BPUNCH 3-20
COPY 3-21
DEB 3-22
DUMP 3-23
FILCOM 3-25
GTOD 3-26
INIT 3-27
LOAD 3-28
MKABS 3-30
MKSAVE 3-31
PRINT 3-32
PUNCH 3-33
RELEASE 3-34
SDAY 3-35
STOD 3-36
TYPE 3-37
XFER 3-38
Relocatable Loader (RLDR) 3-39
Text Editor 3-41
Extended Assembler (ASM) 3-42
Library File Editor 3-45
SYSGEN (SYSG) 3-49
FORTRAN IV Compiler (FORT) 3-51

CHAPTER 4 - PROGRAM MODE OF SYSTEM COMMUNICATION

System Command Words 4-1
Command Word Format 4-1
Status on Return from System 4-2
Byte Pointer 4-4
Initialization of Communications 4-4
File Attribute Command (.GTATR) 4-5

CHAPTER 4 - PROGRAM MODE OF SYSTEM COMMUNICATION

Input/Output Commands 4-7
Open a File (.OPEN, .EOPEN, .ROPEN) 4-8
Get the Number of a Free Channel (.GCHN) . . 4-9
Close a File (.CLOSE) 4-10
Close all Files (.RESET) 4-10
Read a Line (.RDL) 4-10
Read Sequential (.RDS) 4-12
Use of the Card Reader ($CDR) in .RDL and .RDS Commands 4-12
Write a Line (.WRL) 4-15
Write Sequential (.WRS) 4-16
Console Commands 4-17
Get a Character (.GCHAR) 4-17
Put a Character (.PCHAR) 4-17
Memory Commands 4-18
Determine Available Memory (.MEM) 4-18
Change NMAX (.MEMI) 4-19
Clock/Calendar Commands 4-19
.GTOD 4-20
.STOD 4-20
.GDAY 4-20
.SDAY 4-21
.DELAY 4-21
.GHRZ 4-22
Servicing User Interrupts 4-22
.IDEF 4-22
.IRMV 4-23
Error Messages 4-24
Device Response to SOS Commands 4-26
User Status Table 4-30

CHAPTER 5 - CONFIGURING SOS UTILITY PROGRAMS

Supplied Tapes 5-1
Producing a Trigger 5-5
Procedures for Paper Tape Utilities 5-6
Configuring Utilities except the Assembler. . . 5-6
Configuring the Assembler 5-7
Producing a Master Reel 5-8

APPENDIX A - ADDING A USER-SUPPLIED DEVICE HANDLER TO SOS

Introduction A-1
SOS Device Handling Strategy A-2
SOS Links for Device Handlers A-5
SOS Device Control Table (DCT) A-10
SOS Interrupt Search List A-14
SOS Channel Number to Device Map A-17
SOS Filename Table A-18
SOS Interrupt Handling A-22
Device Start, Stop, and Dispatch Routines A-23
    Device Start Routine A-23
    Device Stop Routine A-23
    Device Dispatch Routines A-24
SOS Linkage A-24
Global SOS Subroutines A-27
Generalized SOS Subroutines A-27
    .OPN A-27
    .CLS A-28
    .WRSE A-28
    .WRLI A-29
    .RDSE A-29
    .RDLI A-30
    .RCHR A-30
    .ACHR A-31
    .IBUF A-31
    .OBUF A-32
    .STB A-32
    .LDB A-33
    .DISM A-33
    .IDCT A-33
    .SYSE A-34
Device Driver Example A-35

APPENDIX B - USER APPLICATION ROUTINES

Save-Restore Program (SAVRE) B-1
Command Table Builder (CTB) Program B-4

APPENDIX C - SYSTEM PARAMETERS

RDOS User Parameters (PARU.SR) C-1
SOS Parameters (PARA.SR) C-12
SOS User Application Parameters (PARU.SR) . . . C-15

LIST OF FIGURES

Figure 1-1 Loaded Program in Core 1-6

Figure 5-1 Sample $TTO Dialogue during Generation of a Master Reel 5-16

Figure A-1 SOS FNAME Program A-20

Figure A-2 Device Driver for $PTP A-36

LIST OF TABLES

Table 2-1 SOS Devices 2-1

Table 2-2 External Normal Symbols for Optional SOS Routines and Device Drivers 2-3

Table A-1 System Links for Adding User Device Handlers A-7

Table A-2 User Approaches for Adding Device Handlers A-8

CHAPTER 1

INTRODUCTION TO SOS

The Stand-Alone Operating System (SOS) Package consists of the SOS Libraries and the SOS Utility Programs. The SOS Libraries contain routines which perform I/O for all programs that execute in a non-disk environment. This I/O is performed on an interrupt driven basis, using core buffers unique to each declared device.

The SOS Utilities provide all the development facilities necessary to create, compile, assemble, load, and execute programs in the Stand-Alone environment. These programs perform their I/O through the SOS Library routines. The minimum core requirement for using the SOS Library routines is 4K; the minimum requirement for using the SOS Utiltities is 8K.

INTRODUCTORY CONCEPTS

The SOS Libraries are the following:

The SOS Cassette Tape Library (099-000041)

This library contains device driver routines which support from 1 to 8 cassette tape units; it is supplied to all users having cassettes.

The SOS Magnetic Tape Library (099-000042)

This library contains device driver routines which support from 1 to 8 magnetic tape transports; it is supplied to all users having magnetic tape units.

The SOS Library (099-000010)

This library is supplied to all SOS users. It contains device driver routines for all I/O devices except cassette and magnetic tape. It also contains the main control program, several user application routines, and the RDOS to SOS Interface. This routine makes the SOS program interface RDOS-compatible.

The arrangement of these programs as library files is convenient for SOS users; all desired routines may be selected from the libraries at (relocatable) load time

and incorporated into the load module. All undesired routines and device drivers are excluded from the load module. However, only three or less library files need to be passed through the Relocatable Loader rather than many separate relocatable binary files. The user must, of course, provide an external reference in a previously "loaded" program to cause the selection of any desired library programs for "loading" (see The Relocatable Loaders Manual, 093-000080, Loading of Library Tapes). The ordering of the programs within these libraries assures that the necessary external references are kept to a minimum. (See Chapter 2, Loading SOS Routines and Device Drivers.)

RDOS TO SOS INTERFACE PROGRAM

The RDOS to SOS Interface (RDSI) program makes SOS fully RDOS compatible. The primary difference between the two versions of SOS (with and without RDSI) is the following:

Programs that run with RDSI open files and devices by name; the "open" establishes a link between the file/device name and an RDOS Channel Number. Further accesses to the file in order to read, write, or close are made on the RDOS Channel Number. RDOS Channel Numbers from $0\text{-}76_8$ are available.

Programs that run without RDSI open files and devices on a physical SOS Channel Number. Further accesses to the file are made on the same SOS Channel Number. The SOS Channel Numbers range from $0\text{-}37_8$.* Table 2-1 lists the SOS file/device names and their associated channel numbers. (See also Appendix A.)

The former version of SOS is selected from the SOS Library by declaring .RDSI as an EXTERNAL NORMAL (.EXTN) symbol; the latter version is selected by declaring .SOS as an EXTERNAL NORMAL.(See Table 2-2, External Normal Symbols for Optional SOS Routines).

The only system commands that differ depending on the version of SOS being used are .OPEN and .GCHN. See Chapter 4, Program Mode of Communication, for a description of all System Commands.

Because of the occasional need to distinguish between these versions of SOS, the terminology below will be adopted for the remainder of this manual:

> RDOS - SOS shall refer to SOS configurations in which the RDSI program is present.

*However, 0-4 are unassigned; they are available for adding user device handlers. See Appendix A.

RDOS TO SOS INTERFACE PROGRAM (Continued)

Standard SOS shall refer to SOS configurations in which the RDSI program is absent.

Whenever the term SOS is used without one of the modifiers above, either SOS configuration is implied.

COMMUNICATING WITH SOS

There are three principal ways for a user to interface with SOS and to make the system work for him. These ways are:

- via programmed system calls
- through the console Command Line Interpreter commands (Only available for SOS configurations which include cassette or magnetic tape.)
- through the SOS Utility programs

System calls are issued as program instructions; they activate logic within the routines supplied in the SOS Libraries. Chapter 4 describes each system call recognized by SOS.

The Command Line Interpreter (CLI) is a system program that accepts command lines from the console and translates the input as commands to the operating system. In this manner, the CLI acts as an interface between the user at the console and SOS. The CLI outputs a ready message prompt, "R" when it is first loaded into core and whenever it completes the execution of a console command. The action of the CLI can be interrupted by depressing the reserved key combinations: CTRL and A or CTRL and C. The CLI may also be used to load the SOS utilities into core via the "BOOT" command. Chapter 3 describes the CLI in detail.

Each SOS Utility also acts as an interface between the user at the console and the operating system. These programs are more specific in purpose than the CLI however. For example, the SOS Editor produces ASCII and only ASCII output files for the console user; the SOS Assembler produces only absolute or relocatable binary files from assembler source files; and the SOS FORTRAN IV Compiler produces only assembler source files from FORTRAN source files.

COMMUNICATING WITH SOS (Continued)

All SOS users receive five SOS Utilities:

SOS Text Editor

SOS Extended Assembler

SOS Library File Editor

SOS SYSGEN Program

SOS FORTRAN IV Compiler (requires 16K of core minimum)

SOS Users having either cassette or magnetic tape drives receive three additional programs:

SOS Relocatable Loader*

SOS Command Line Interpreter

Core Image Loader/Writer

The operation of each SOS Utility is described in Chapter 3. The SOS Library and Utility tapes supplied for each system are listed in Chapter 5.

GENERATING SOS SYSTEMS

Because of the many combinations of devices at user installations, the facility to tailor utility programs to the hardware configuration is provided with the package. The procedure to produce a tailored system generally consists of performing a relocatable load on:

(1) the appropriate "trigger" relocatable binaries (.RBs) ;

(2) the SOS Libraries and,

(3) the respective utility .RBs.

The output from this procedure is a core image of the tailored SOS Utility; this image can then be saved as either a save file on cassette or magnetic tape or as an absolute binary file on paper tape. The save file may thereafter be loaded directly into core by the Core Image Loader; the absolute binary may be loaded directly by the Absolute Binary Loader. The "trigger" file, which causes the selection of desired routines from the SOS Libraries, may be produced by the SOS SYSGEN program. Relocatable binaries of each of the SOS Utilities are supplied in order for the user to perform these procedures. Preconfigured utilities are also

*SOS users without cassette or magnetic tape drives are supplied the Stand-Alone Extended Relocatable Loader, Absolute Binary: 091-000038; see Chapter 5.

supplied for systems with standard configurations; these programs are provided in either absolute binary or save file format. Chapter 5 gives detailed instructions for configuring SOS Utilities in magnetic tape, cassette tape, or paper tape environments.

SOS ORGANIZATION

All I/O drivers, control routines, tables, etc., included in the SOS Libraries are relocatable. They may occupy any block of core above location 440_8; this is the first location into which any of the DGC Relocatable Loaders deposit normally relocatable (.NREL) data. Locations $0-17_8$ in page zero are appropriated by the main control program in the SOS Library. Location 377_8 in page zero contains a "JMP" @2" instruction; whenever a program is started at location 377, a full initialization of SOS (equivalent to system command ".SYSI") is performed and control is then passed to the starting address specified in the user or utility program.

This address resides in the User Status Table or UST. Each user program has an associated UST. This table is set up by the Relocatable Loader; it contains information such as starting address, start and end of symbol table, highest address used, etc. It begins at location 400_8. See Chapter 4, User Status Table.

The highest 400_8 locations in memory are normally occupied by either the Core Image Loader/Writer or the Absolute Binary Loader. For this reason, the SOS memory commands (see Chapter 4, .MEM and .MEMI) recognize the highest physical memory location -400_8 as the highest memory available (HMA) to the user.

The memory that is available for user or utility programs and data is then essentially the following:

- All page zero locations (inclusive) between 20_8 and 376_8.
- All other memory between 440_8 and the last memory address -400_8, except that which is necessary for the selected combination of SOS routines, tables, and buffers.

Figure 1-1 gives a sample representation of all core after a user module has been loaded with SOS Libraries. For illustrative purposes, the module is assumed to be comprised of two user programs, USR1 and USR2, both of which contain normally and page zero relocatable data.

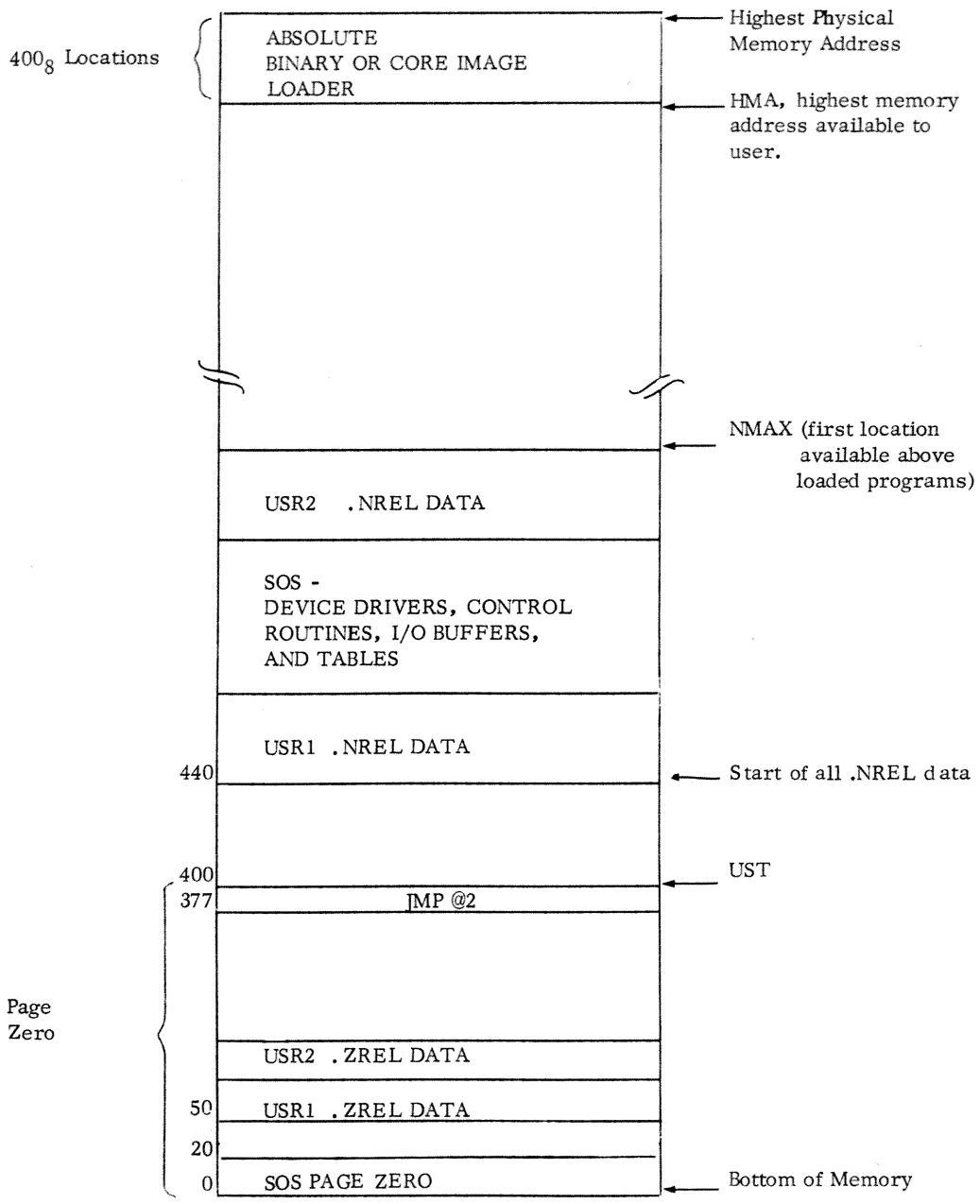


Figure 1-1. Loaded Program in Core

CHAPTER 2

SOS FILES AND DEVICES

SOS files may reside on any SOS device; these devices include magnetic tape units, high speed paper tape reader, high speed paper tape punch, teletype keyboard, etc. In this sense, the terms file and device are interchangeable in the SOS environment. Files are "read from" and "written to"; devices are also "read from" and "written to". In the case of magnetic and cassette tapes however, the term file is slightly more encompassing:

> The tape unit or device may contain several files. For example, cassette unit 1 may contain the files CT1:0, CT1:1, CT1:2 and CT1:3.

Cassette and magnetic tape units are the only devices upon which more than one file may reside at the same time. The expression file/device which occurs throughout the remainder of this manual denotes this distinction.

SPECIFYING SOS DEVICES

In a Standard SOS environment, a device is "opened" by passing its fixed SOS Channel Number to the operating system. Under RDOS-SOS, a device is "opened" by passing its filename and an RDOS Channel Number ($0\text{-}76_8$) to the operating system; the system then translates that name into the appropriate SOS Channel Number in order to complete the "open". The RDOS Channel Number that is passed is only assigned to that device for the duration of the "open". The SOS Channel Numbers are always assigned to the same devices however. The list of SOS devices by name and by SOS Channel Number is shown in Table 2-1.

Device Name	Fixed Channel No.	Device Name	Fixed Channel No.
TTI1	5	MT3	23
$PLT	6	MT4	24
TTO1	7	MT5	25
$TTP	10	MT6	26
$CDR	11	MT7	27
$TTO	12	CT0	30
$TTI	13	CT1	31
$LPT	14	CT2	32
$PTR	15	CT3	33
$PTP	16	CT4	34
$TTR	17	CT5	35
MT0	20	CT6	36
MT1	21	CT7	37
MT2	22		

Table 2-1. SOS Devices

SPECIFYING SOS DEVICES (Continued)

The device names are the same as those recognized by RDOS. TTI1 and TTO1 refer to a second teletype which uses device codes 50_8 and 51_8. The other names are self-explanatory. All SOS Utilities recognize devices by the names shown in this table.

LOADING SOS ROUTINES AND DEVICE DRIVERS

In order to select any program in the SOS Libraries for loading, an ENTRY symbol (.ENT) in the desired program must resolve an EXTERNAL (.EXTN or .EXTD) symbol declared in a previously loaded program. It is the user's responsibility to supply these EXTERNAL declarations to the Relocatable Loader before the SOS Libraries are provided as input to the loader. Each separately assembled program in the SOS Libraries has one or more ENTRY symbols associated with it. Many of these programs also have EXTERNAL declarations which select successive library programs for loading, once they themselves are loaded. For example, the SOS Magnetic Tape Library contains 8 separate programs: a control table and I/O buffer for units 1-7 plus a control table, I/O buffer and device driver (which controls from 1-8 units) for unit 0. In order to load the necessary programs to support 8 units, only one EXTERNAL declaration need be specified however. That program will then cause the loading of each subsequent program in the library.

The complete set of necessary EXTERNAL declarations in relocatable binary format is called a trigger. Triggers are produced in two ways:

(1) By assembling a source program of the form:

```
.TITLE    TRIGGER
.EXTN     A,B,C,...
.END
```

where A,B,C,... are the symbols that are ENTRY points in the desired routines in the SOS Libraries.

(2) By using the SYSGEN program to produce the relocatable binary file directly. In the above example, the SYSGEN command line shown below would produce an equivalent file on the high speed punch.

```
(SYSG) TRIGG/T $PTP/O A B C ... )
```

Table 2-2 lists the necessary EXTERNAL NORMAL declarations in order to select any SOS Library program for loading.

* Programmers using SOS or Stand-alone RLDR must use .EXTN.

CASSETTE SOS LIBRARY* 099-000041		
External Normal	Program Name	Purpose
.CTU7	CTU7	control table/core buffer to support cassette units 0-7.
.CTU6	CTU6	control table/core buffer to support cassette units 0-6.
⋮	⋮	⋮
.CTU1	CTU1	control table/core buffer to support cassette units 0 & 1.
.CTAD	CTADR	cassette driver w/control table/core buffer for unit 0.

MAGNETIC TAPE SOS LIBRARY* 099-000042		
External Normal	Program Name	Purpose
.MTU7	MTU7	control table/core buffer to support mag tape units 0-7.
.MTU6	MTU6	control table/core buffer to support mag tape units 0-6.
⋮	⋮	⋮
.MTU1	MTU1	control table/core buffer to support mag tape units 0 & 1.
.MTAD	MTADR	magnetic tape drive w/control table/core buffer for unit 0.

SOS LIBRARY 099-000010		
External Normal	Program Name	Purpose
.CTB	CTB	command table builder for SOS application programs
.SAVR	SAVRE	save/restore routine for SOS user application programs
.RDSI	RDSI	RDOS to SOS interface program (Will cause loading of .SOS)
.PLTD	PLTDR	plotter driver (will cause loading of .SOS)
.CDRD	CDRDR	card reader driver (will cause loading of .SOS)
.PTRD	PTRDR	paper tape reader driver (will cause loading of .SOS)
.PTPD	PTPDR	paper tape punch driver (will cause loading of .SOS)
.LPTD	LPTDR	80 column line printer driver (will cause loading of .SOS)
.L132	LP132	132 column line printer driver (will cause loading of .SOS)
.STTY	STTYDR	small teletype driver ($TTI, $TTO only) (will cause loading of .SOS)
.TTI1	TTY1	second $TTI and $TTO driver (will cause loading of .SOS)
.RTC5	RTCT5	50 Hz (line frequency) real time clock driver (will cause loading of .SOS)
.RTC4	RTCT4	60 Hz (line frequency) real time clock driver (will cause loading of .SOS)
.RTC3	RTCT3	1000 Hz real time clock driver (will cause loading of .SOS)
.RTC2	RTCT2	100 Hz real time clock driver (will cause loading of .SOS)
.RTC1	RTCT1	10 Hz real time clock driver (will cause loading of .SOS)
.SOS	SOS	main program of SOS

*When performing relocatable loads with the SOS Libraries, the Magnetic Tape and Cassette Libraries must always precede the SOS Library as input to the loader.

Table 2-2. External Normal Symbols for Optional SOS Routines and Device Drivers

CASSETTE AND MAGNETIC TAPE FILES

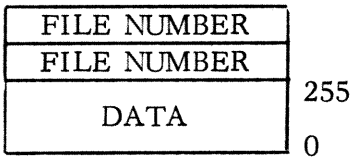
A single magnetic tape or cassette unit may contain up to 100 files, which are designated 0 through 99. Both the unit number and the file number must be specified in order to open these files.

Physical Characteristics of Cassette and Magnetic Tapes

Up to eight cassette drives and eight magnetic tape drives are permitted per system. Magnetic tape drives must be 9-track.

NOTE: No magnetic tape file may be written unless the write permit ring is inserted in the tape reel. No cassette file may be written unless the file protect tabs are in place on the cassette.

Data is read and written from cassette and magnetic tapes in the same format, fixed length blocks of 257_{10} words. The first 255 words of each block are data and the last two words contain the file number:



Files are variable in length, consisting of as many 257-word blocks as required. If the last block is not completely used, the remainder is padded with nulls.

After the first file and each succeeding file, an end-of-file (EOF) mark is written. Two EOF marks terminate the last file, so that all tapes have the following format:

Start of reel	
File 0	(n_0 blocks)
EOF	
File 1	(n_1 blocks)
EOF	
⋮	
File k	(n_k blocks)
EOF	
EOF	

Files must be placed on tape in numeric order. An attempt to place a fourth file (MT0:3) on a tape containing only MT0:0 and MT0:1 will cause an error indicating FILE NON-EXISTENT.

Physical Characteristics of Cassette and Magnetic Tapes (Continued)

Since consecutive EOF marks always indicate the end of tape to SOS, overwriting a file makes any previously succeeding files inaccessible. For example, if file 3 were overwritten on a tape containing 13 files (files 0-12), files 4 though 12 would become inaccessible and an attempt to reference any file greater than 3 would result in a FILE NON-EXISTENT error.

System commands described in later chapters are used to initialize a tape drive and release a tape drive. The CLI "INIT" command causes a tape on that drive to be rewound and initialized. Full initialization (INIT/F) causes the tape to be rewound and two EOF's to be written. An INIT/F should be performed on all blank (new) magnetic tapes and cassettes before using them. The RELEASE command causes the tape to be rewound and then released from the system.

If a parity error is detected during reading, the system attempts to reread data ten times before issuing a FILE DATA error. If an error is detected after writing, the system will attempt to backspace, erase and rewrite ten times. If the rewrite is unsuccessful after ten times, a FILE DATA error is returned.

Opening Magnetic Tape and Cassette Files

A cassette or magnetic tape file is opened using the .OPEN command (see page 4-8). When performing an .OPEN command under RDOS-SOS, AC0 must contain a byte pointer to the unit name/file number. The string for the pointer has the format:

MT*n*:*dd*

where: *n* is the unit number (0-7) of the magnetic tape unit.

dd is the file number (0-99).

Either a one-digit or two-digit number may be used to reference file numbers 0-9. File number 8 on unit 2 could be referenced as:

MT2:08 or MT2:8

CT2:08 or CT2:8

Opening Magnetic Tape and Cassette Files (Continued)

An example of this would be:

```
        LDA  0,FILE8            ;LINKS THE FILE
        .SYSTM
        .OPEN 3                 ;SPECIFY CHANNEL (0-76 OCTAL)
        error return
        normal return
FILE8:  .+1*2
        .TXT   *MT2:08*         ;FILE
```

Thus, channel number 3 is linked to the file (MT2:8) as specified by the byte pointer. Once the file has been opened, the file is then referenced by channel number 3.

Under Standard SOS, the fixed channel number is given in the .OPEN command (20-27 for magnetic tapes and 30-37 for cassette units). AC0 contains the file number, which in this case can be any positive number from $0\text{-}99_{10}$. For example:

```
        LDA  0, FILE8
        .SYSTM
        .OPEN 22                ;MT2
        error return
        normal return
FILE8:  8.                      ;FILE NUMBER
```

or

Opening Magnetic Tape and Cassette Files (Continued)

```
        LDA     0, FILE8
        LDA     2, CHNO
        .SYSTM
        .OPEN 77                ;77(8) IS DEVICE CODE OF CPU
        error return
        normal return
FILE8:  8.                      ;FILE NUMBER
CHNO:   22                      ;MT2
```

The second example illustrates that a channel number need not be specified in the command itself. The device code of the CPU is specified in the .OPEN command and the system uses the channel number passed in AC2. Note that FILE8 was a byte pointer in AC0 for RDOS-SOS but simply contains a file number for Standard SOS.

CHAPTER 3

SOS UTILITY PROGRAMS

INTRODUCTION

This chapter describes the operation of each of the SOS Utility Programs in detail. These utilities are supplied to users in three forms:

1. Relocatable Binary
2. Absolute Binary
3. Core Image or Save files

The absolute binary and save file versions are preconfigured with I/O support for all common SOS devices. The relocatable binary versions permit a user to tailor each utility exactly to his device configuration. Chapter 5 describes these procedures in detail.

The utility programs that are available for all SOS environments are the following:

SOS Text Editor
SOS Extended Assembler
SOS Library File Editor
SOS SYSGEN
SOS FORTRAN Compiler (requires minimum 16K core)

SOS environments that include at least one cassette or magnetic tape drive qualify for three additional utilities:

SOS Relocatable Loader
SOS Command Line Interpreter
Core Image Loader/Writer

PAPER TAPE OPERATION

SOS utility programs can be operated using the binary loader to load absolute binary paper tapes from the high-speed paper tape reader or the teletypwriter paper tape reader. The procedures for loading programs in this manner are described in Section 2.8 of "How to Use the Nova Computer."

CASSETTE/MAGNETIC TAPE OPERATION

Systems with one or more cassette or magnetic tape drives have two additional means of loading the SOS utilities:

1. Via the Core Image Loader
2. Via the CLI "BOOT" command

Files are specified to the Core Image Loader by unit number/file number (see the following explanation). Files are specified to the CLI "BOOT" command by either filename (CT0:4, MT2:0, etc.) or by utility mnemonic (ASM, EDIT, FORT, etc.). To load utilities by mnemonic, the CLI assumes the following order of these programs on unit 0:

File 0 : Core Image Loader/Writer
File 1 : Relocatable Loader
File 2 : Command Line Interpreter
File 3 : Text Editor
File 4 : Extended Assembler
File 5 : Library File Editor
File 6 : SYSGEN
File 7 : FORTRAN IV Compiler (optional)

A reel of magnetic or cassette tape with this arrangement of save files is called a master reel. Procedures for generating a master reel are discussed in Chapter 5, Configuring SOS Utility Programs.

SYSTEM CONSOLE BREAKS

There are two possible program breaks that can be generated at a system console: CTRL A and CTRL C breaks. The treatment of these breaks is generally consistent throughout the SOS utilities.

Pressing CTRL and A on a console keyboard causes the currently executing utility to stop, initialize, and re-issue its prompt message. For example if the CLI were performing a File Compare when this break was entered, the compare (and consequent listing) would terminate and the CLI prompt message (R) would be re-issued to the console. If this break was entered during an assembly, the assembly would terminate immediately and the assembler prompt message (ASM) would be re-issued. All SOS utilities except the Editor treat the CTRL A break in this manner.

SYSTEM CONSOLE BREAKS (Continued)

The SOS Editor treats the CTRL A break somewhat differently. If the break is entered while one of the following Editor commands is in progresss:

T, Y, N, E, or P

then the operation is terminated and the program restarts with I/O reset but with the input buffer intact. If this break is entered at some other time while the Editor is executing, it is ignored.

Pressing CTRL and C on a console keyboard causes an immediate transfer to the Core Image Loader program (and is consequently ignored by all non-cassette or magnetic tape utilities). This program issues its prompt message (#) and waits for the user to specify the next program to be loaded. It is very convenient when executing from a master reel (see Chapter 5) to use this feature to swap the utilities in and out of core. All program activity terminates immediately when this break is entered. All appropriately configured SOS utilities except the Editor treat the CTRL C break in this manner.

The SOS Editor ignores this break however. To return to the Core Image Loader from this utility, the "H" command must be issued.

The section on the User Status Table, page 4-30, describes the means available for employing these breaks by user programs.

CORE IMAGE LOADER/WRITER

The core image loader/writer is a utility program that performs two functions: it loads core image files from cassette or magnetic tape into core and produces core image or save files on cassette or magnetic tape from the contents of core. There are two versions of the core image loader/writer -- one for use with cassette drives and another for use with magnetic tape drives. The cassette version works only with cassettes and the magnetic tape version works only with magnetic tape. Both versions when loaded occupy the last 400_8 locations in core.

Installation Procedure

The absolute binary version of the Core Image Loader/Writer, when loaded into core, permits the user to install it on unit 0, file 0 of the appropriate tape. The program types the message

LOAD UNIT 0: STRIKE ANY KEY

at the system console. When a tape reel that is not write protected is mounted on unit 0 and the unit is ready, the user may depress any teletype key. The program then writes the appropriate subset of itself to file 0 of that unit. When this write operation is complete, the message

LOADER INSTALLED

is typed at the system console and the program HALTs. This tape reel is now usable for the Bootstrap Procedure described below. The Installation Procedure may be repeated; depress the CONTINUE switch on the master console and the LOAD UNIT 0: STRIKE ANY KEY message will be issued. A new tape reel may be mounted on unit 0, or the unit assignments changed and the procedure repeated.

Bootstrap Procedure

The core image loader/writer can be bootstrap-loaded from file 0 of the master cassette or magnetic tape reel. The master reel must be installed on cassette or magnetic tape unit 0 and the drive must be rewound manually. This can be done by pressing the REWIND button on the drive unit.

For machines without the Program Load option, deposit 060134 (for cassette units) or 060122 (for magnetic tape units) into location 376_8, and deposit 000377 into

Bootstrap Procedure (Continued)

location 377_8. Press the console switches for RESET and then START.

For machines with the Program Load option, set the data switches on the console to 100034 (for cassette units) or 10022 (for magnetic tape units), press RESET and then PROGRAM LOAD.

The core image loader/writer is read into page zero initially and then relocates itself to the high end of memory. At the end of the relocation process, the loader outputs a prompt (#) on the teletype. This prompt indicates that the core image loader is ready to accept a command. Whenever the core image loader/writer is resident in core, the core image loader may be restarted by setting the data switches to the address of the last location in memory, pressing RESET, and then pressing START. The core image writer can be started by setting the data switches to the address of the next to last location in core, pressing RESET and then pressing START.

Core Image Loader Operation

Having issued the # prompt on the teletype, the core image loader waits for an operater response of a device number (0-7) and a file number (0-99) separated by a colon. Device 0 need not be specified. For example:

```
#2:7)     (unit 2, file 7)
#4 )      (unit 0, file 4)
```

The indicated cassette or magnetic tape file is loaded from the specified device into memory, starting at the address specified in location 2. If the A key is struck instead of the carriage return after the unit and file entry, the loader will halt after loading is complete. For example:

```
#1:2A
```

causes file 2 of unit 1 to be read into core followed by a HALT.

The Core Image Loader always sets the UST CTRL C break location (USTBR) to point to itself after it loads a program. The loaded program may then ignore, disable, or change this location. (See User Status Table, page 4-20.)

If the core image loader encounters a non-recoverable error while trying to load a core image file, it will type

```
*ERR
```

and halt with the cassette or magnetic tape status word in AC0. The following list describes the error conditions assigned to each bit in the status word.

Core Image Loader Operation (Continued)

BIT	MEANING
1	Data late (perhaps due to a long indirect access chain or a faster device preempting the channel.
3	Illegal command
5	Lateral parity error in a word
6	Addressed tape is beyond the EOT marker
8	Addressed tape is at load point
10	Bad tape (e.g., data is found in an interrecord gap)
13	Unit is write locked
14	Odd number of bytes detected in a read or write attempt

If rewinding and substituting a fresh cassette or magnetic tape does not cure an error condition, a hardware malfunction is indicated; run the appropriate diagnostic program.

Core Image Writer Operation

The core image writer operates in a manner similar to that of the core image loader. When the core image writer is started (see page 3-5), it outputs a # prompt and waits for specification of a device number and a file number separated by a colon. Unit 0 need not be specified. After the file has been specified, the core image writer will request specification of the upper core address (NMAX) to be written onto tape. It does this by typing:

NMAX:

on the teletypewriter. The operator must then respond with the highest core address (in octal) whose contents he wants written into the core image cassette or magnetic tape file. Upon detection of a non-recoverable error, the core image writer proceeds in the same way as the core image loader. After completing a successful write, the program types OK and HALTS.

COMMAND LINE INTERPRETER (CLI)

CLI Definition

The Command Line Interpreter (CLI) is a system program that accepts command lines from a system console and translates them into commands to the operating system. The CLI acts as an interface between the user at the console and the system. In addtion, the CLI performs certain file housekeeping chores for the user and implements loading of other utility programs from a master cassette or reel.

To use the CLI, the core image loader must be resident in core and a master cassette or reel mounted on CT0 or MT0. A 0:2 command following the # prompt will load the CLI.

Ready Message

The CLI indicates to the user that the system is idle and that the CLI is ready to accept commands by typing a ready message on the console. The message consists of "R" followed by a carriage return.

CLI Activation

The user activates CLI responses to a command by typing a line and pressing the RETURN key or the CTRL L (form feed) keys. The CLI will not respond until RETURN or CTRL L is pressed. (RETURN and CTRL L are interchangeable; use of RETURN -- shown as) -- in the remainder of the text means either line terminator.)

Symbols and Conventions Used in Command Line Syntax

Symbol	Usage	Example
)	Represents pressing RETURN key, causing termination of the command line input and activation of the CLI.	PRINT $PTR)
↓	Represents pressing CTRL L keys (form feed) which acts the same way as the RETURN key.	TYPE $PTR ↓
\	Represents pressing SHIFT L keys, which causes deletion of the entire line. \ ) are echoed on the teletypewriter.	FILCOM \ )
←	Represents pressing RUBOUT key, which causes erasure of the previous character. ← is echoed on the teletypewriter.	FF←ILCD←OM
Δ (space)	Arguments are separated by commas or spaces. Extra commas and spaces have the same effect as a single symbol. (Note that only commas can be used as argument separators when used with parentheses.)	TYPEΔ$PTRΔCT0:0) TYPEΔ$PTR,ΔCT0:0) TYPEΔ$PTR,ΔΔCT0:0) TYPEΔ$PTR,Δ,CT0:0)
/	Right slash indicates that the character immediately following is to be interpreted as a switch.	XFER/A $PTR $PTP)
;	Command delimiter in a command line. Two or more commands may appear on a line separated by semicolons; none are executed until RETURN is pressed.	INIT MT0; INIT MT1)
↓ (SHIFT N)	The next RETURN is ignored as a command terminator. ↓ must appear as the character immediately before a carriage return.	PRINT $PTR/2 MT0:3↓) MT0:4 $CDR $PTR)
(,,,)	Repetition of commands or arguments.	PRINT MT0:(0,1,2,3))

Note: Use of these symbols and conventions is described in greater detail in sections following.

COMMAND LINES

A command line can consist of one or more commands followed by RETURN. A basic command line has one command.

The CLI command may be used to invoke a utility, invoke any file, or to perform a function. In the case of utility programs, the CLI presumes a sequence of save files on the master cassette or master reel and finds the appropriate program based on that assumption. If, for example, the user types the following command line:

```
BOOT ASM )
```

the CLI will load the assembler into core. The effect is the same as a 0:4 command to the Core Image Loader. The CLI is overwritten in core by the assembler and may only be restored to core by specifying the physical save file on which it resides (e.g. 0:2) to the Core Image Loader for loading.

Similarly,

```
BOOT CT1:2 )
```

will load the third file on unit one with the CLI being overwritten. It may be restored in the same manner as above. (Note: CT1:2 and CT1:02 are synonymous.)

However, for the command:

```
BOOT $PTR )
```

the Core Image Loader is replaced by the absolute binary loader for the purpose of loading absolute binary paper tape.

When performing a function, the CLI is not overwritten but merely executes the specified function and returns after the function is completed with the prompt message R. If, for example, the user types the command line:

```
FILCOM MT0:0 CT1:2 $LPT/L )
```

file MT0:0 is compared to file CT1:2 and dissimilar word pairs are printed in octal with their file displacement on the line printer. The CLI returns with the prompt R.

Stacking Commands on a Command Line

A command line is executed by the CLI when the user presses the RETURN key or the CTRL L keys on the console.

A number of commands may be stacked on a given line for execution. They are separated by semicolons. For example:

```
APPEND MT0:0 $PTR; GTOD; FILCOM MT0:0 MT1:1; BPUNCH MT0:0 MT1:3 )
```

The four commands are executed when the user presses RETURN. The CLI indicates execution of each command with the appropriate information, if any. At the completion of the entire command line, the CLI will prompt the user again with a ready message. For example, the previous command line will cause the response:

```
LOAD $PTR, STRIKE ANY KEY     ←  Prompt message for APPEND
05/22/74          18:48:40    ←  Response to GTOD
R                             ←  After MT1:3 is "BPUNCHed"
```

Long Command Lines

Each command line is limited to 132 characters in length*, although this may be extended by typing the symbol ↑ immediately before pressing the RETURN key. The up arrow causes the carriage return to be ignored. For example:

```
INIT CT2; PRINT CT0:2; PUNCH CT0:2; DUMP MT0:0 ↑ )
CT0:3/S FORT. LB )
```

is executed as if the following had been typed:

```
INIT CT2; PRINT CT0:2; PUNCH CT0:2; DUMP MT0:0 CT0:3/S FORT. LB )
```

In the previous example, the second line starts a new argument. Note that when a RETURN is ignored, there is no delimiter between the last character on one line and the first character on the next line. Therefore, in the example the blank argument delimiter has been inserted before the up arrow.
The user can, of course, break an argument or command word into two lines:

```
INIT CT2; PRINT CT0:2; PUNCH CT0:2; DU ↑ )
MP MT0:0 CT0:3/S FORT. LB )
```

is equivalent to:

```
INIT CT2; PRINT CT0:2; PUNCH CT0:2; DUMP MT0:0 CT0:3/S FORT. LB )
```

---

* The maximum length of all commands input to the CLI at any one moment is 300 characters (not counting ↑ up arrows).

Switches

Commands and their arguments may be modified by a series of switches pertaining to the command or argument. A switch is indicated by a right slash (/) followed immediately by either a letter or a decimal digit.

Numeric switches specify the number of times the previous argument is to be repeated in the command line. For example:

```
PRINT $PTR/6 )
```

indicates that six ASCII tapes from the paper tape reader are to be printed by the line printer.

Numeric switches are cumulative. The following commands are equivalent:

```
PRINT $PTR/1/3/2 )

PRINT $PTR/6 )
```

The digit 1 in a numeric switch is the same as no switch. The following commands are equivalent:

```
PRINT $PTR )

PRINT $PTR/1)
```

The digit 0 has no effect upon the number of times a file name is repeated if it appears in a list of numeric options. For example, the following commands are equivalent:

```
PRINT $PTR/6 )

PRINT $PTR/1/0/2/3 )

PRINT $PTR /1/2/3/0/0 )
```

However, when used alone, the 0 switch has the same effect as 1. For example, the following are equivalent commands:

```
PRINT $PTR/1 )

PRINT $PTR/0 )

PRINT $PTR )
```

Switches (Continued)

Letter switches have distinct meanings that depend upon the command or argument with which they are associated. The detailed descriptions of each CLI command indicate the meanings of each letter switch that can be used in the command.

A letter switch that follows a command word is a global switch and applies to all arguments of the command line. A switch that follows an argument is a local switch and applies only to the particular argument. For example, the Make Absolute Binary command (MKABS) has both a local and global switch, S , to specify the address that is placed in the start block of the absolute binary file that is produced. The command:

```
MKABS  MT0:1  $PTP )
```

causes the save file on MT0:1 to be punched in absolute binary format on the high speed punch. The start block on the tape will have an address of zero which causes the binary loader to HALT after the tape is loaded.

The global switch, S, causes the address contained in the UST location USTSA to be placed in the start block. This address represents the relocated address of the last relocatable binary start block that passed through the loader when the save file was generated. Hence the command:

```
MKABS/S MT0:1 $PTP )
```

causes such a block to be produced in the absolute binary tape. When the binary loader completes the loading of this tape, the program will begin execution at its starting address.

The local switch, S, causes the octal address that it modifies to be placed in the start block. This feature is useful for cases where a starting address is missing from the save file or an address other than the user specified address at USTSA is desired. An example of such a command is:

```
MKABS  MT0:1 $PTP  377/S )
```

Effect of Switches on Command Lines

A switch affects a command line as if the switch were a comma or space. For example, the following commands are equivalent:

```
MKABS/S  MT0:1   $PTP )
```

Effect of Switches on Command Lines (Continued)

```
MKABS /S MT0:1 $PTP)

MKABS/SMT0:1 $PTP )
```

Thus the switch delimits the command word MKABS from the argument MT0:1.

If a character other than a number or a letter follows the right slash, the slash acts merely as a delimiter. For example,

```
MKABS/ S MT0:1 $PTP)
```

The slash is ignored because it is followed by a space. Since there is no SOS file or device named S, an error message will result.

Comma/Parenthesis Convention

CLI commands can be repeated with each of several arguments or argument strings by first separating the arguments or strings with commas and then by enclosing all the arguments within single parentheses. Moreover, a series of commands can be made to use a common argument by placing the commands within single parentheses. The general form of the convention is:

```
S1 (S2, S3,...,Sn) Sn+1 ... )
```

where: S1...Sn+1 are strings within the command line sequence.

Use of the above generalized form results in a series of commands which are executed serially by the CLI:

```
S1 S2 Sn+1...)

S1 S3 Sn+1... )
    .
    .
    .
S1 Sn Sn+1 ... )
```

Only one set of parentheses is permitted per command line. Neither command terminators (carriage return, form feed, semicolons, additional left parentheses) nor nulls can be present within the parentheses. Use of this convention to compare a series of tape files is illustrated in the single command:

Comma/Parenthesis Convention (Continued)

```
FILCOM (MT0:1 MT1:2, MT0:0 CT0:0) $LPT/L )
```

the CLI expands this command into the following pair of CLI commands:

```
FILCOM MT0:1  MT1:2  $LPT/L )

FILCOM MT0:0  CT0:0  $LPT/L )
```

Use of comma without argument strings within parentheses is permitted under limited conditions. If the parentheses with commas are preceded by a single command, the command will be repeated n+1 times for n commas.

```
GTOD (,,,) )
```

is equivalent to:

```
GTOD )
GTOD )
GTOD )
GTOD )
```

Messages Concerning I/O

Some commands require manual operation of an I/O device. If the user issues such a command, he will receive a message prompting or instructing him as to the proper action. For example, if the user issues the command:

```
XFER/A $PTR MT0:0 )
```

which requests that a source file be transferred from the paper tape reader to a magnetic tape file named MT0:0 the system replies:

```
LOAD $PTR, STRIKE ANY KEY.
```

The user can then mount the paper tape in the reader and strike any key on the console. The key struck to start the device is not echoed on the console.

When a series of files are to be transferred, or loaded from a device requiring manual intervention for each file, the message will be issued the appropriate number of times. For example, if the user issues the command:

```
APPEND MT0:0 $PTR/2 )
```

which requests that the file MT0:0 be created from two files input from the paper tape reader, the following messages will occur:

```
LOAD $PTR, STRIKE ANY KEY.

LOAD $PTR, STRIKE ANY KEY.
```

The second message is typed out after the first file has been transferred.

Error Messages

When the user issues a command that contains an error, an appropriate error message will be typed out.

When a user gives a command that is legal for some arguments and illegal for others, an error message is issued for each of the illegal arguments. The correct portions of the command are executed. For example,

```
R
APPEND CT1:3 CT2:4 MT1:5 )      concatenate two files to CT1:3
R
XFER $PTR CT0:2 )               transfer file from $PTR to cassette
LOAD $PTR, STRIKE ANY KEY.
R
FILCOM CT1:3 CT0:2 CT2:4 )
TOO MANY ARGUMENTS.
R
FILCOM CT1:3 CT0:2  $TTI/L )
FILE WRITE PROTECTED: FILE $TTI.
R
FILCOM CT1:3 )
NOT ENOUGH ARGUMENTS            no action occurs
R
FILCOM CT1:3 CT0:2 )            correct arguments; listing to $TTO
```

Error Messages (Continued)

In general, error messages are quite explicit, giving the user sufficient information to correct his error easily. A few examples are shown:

```
R
MT0:4 )
NOT A COMMAND: MT0:4          correct command is BOOT MT0:4
R
XFER MT0:3 $PTR )
FILE WRITE PROTECTED: FILE $PTR
R
INIT $PTR )
NOT A DIRECTORY DEVICE: $PTR
R
MKSAVE $PTR MT3:74 )
OUT OF SPACE: FILE MT3:74
R
XFER MT4:5 MT2:7 )
FILE DOES NOT EXIST: FILE MT2:7
R
XFER $TTO $PTP )
FILE READ PROTECTED: FILE $TTO
R
INIT CT8 )
NOT A DIRECTORY DEVICE :CT8
R
BOTO CT1:2 )
NOT A COMMAND: BOTO
R
```

CLI Commands

This section contains definitions and descriptions of each of the CLI commands. The commands are listed in alphabetical order at the bottom of the page and described in that order on pages following.

The following conventions are used to define individual CLI command formats:

All upper case letters represent valid command line elements.

Items in a command line printed in lower case indicate either command information or file names which must be supplied in the command line.

Elements enclosed in modified brackets, ⌈ ⌉ , are optional. Stacked items indicate alternate choices.

The ellipsis (...) is used to indicate that preceding file types or bracketed material may be repeated if desired.

The comma (,), and right slash (/) are significant and necessary parts of any command line definition in which they are found.

APPEND	-	Concatenate two or more files.
BOOT	-	Invoke a utility program or load any file.
BPUNCH	-	Copy a binary file on $PTP.
COPY	-	Copy a cassette or magnetic tape reel.
DEB	-	Start a program about to be executed in the Debugger.
DUMP	-	Produce a dump format file.
FILCOM	-	Compare two files.
GTOD	-	Get time and date.
INIT	-	Initialize cassette or magnetic tape.
LOAD	-	Reload dumped files.
MKABS	-	Make an absolute binary file from a core image or save file.
MKSAVE	-	Make a save file from an absolute binary file.
PRINT	-	Print an ASCII file on the line printer.
PUNCH	-	Copy an ASCII file on the $PTP.
RELEASE	-	Release cassette or magnetic tape.
SDAY	-	Set today's date.
STOD	-	Set the time.
TYPE	-	Output the contents of an ASCII file on the system console.
XFER	-	Copy a file to another file.

Name: APPEND

Format: APPEND newfilename oldfilename$_1$[... oldfilename$_n$]

Purpose: To create a new file, consisting of a concatenation of one or more old files in the order in which their names are listed as arguments. The old files are not changed by the command.

Switches: None.

Examples: APPEND MT0:0 CT1:1 CT1:2 CT1:3 CT1:4)

causes creation of the file MT0:0 containing the contents of files CT1:1 CT1:2 CT1:3 CT1:4 in that order.

APPEND CT0:0 CT1:2 MT1:0 CT1:1 $PTR)

causes creation of the file CT0:0 containing the files CT1:2, MT1:0, CT1:1 and the paper tape reader.

The same device cannot be used for both input and output files.

CLI Commands (Continued)

Name: BOOT

Format: BOOT filename

Purpose: To invoke a utility program or any executable file. To boot a utility program the following are used:

Mnemonics:

ASM	-	load the Assembler
EDIT	-	load the Text Editor
FORT	-	load the FORTRAN IV compiler
LFE	-	load the Library File Editor
RLDR	-	load the Relocatable Loader
SYSG	-	load SYSGEN
CLI	-	load Command Line Interpreter

Switches:

Local: /A - modifies the filename and causes the Core Image Loader to HALT instead of starting the program afer it is loaded.

Examples: BOOT EDIT)

loads and starts the Editor from a master reel.

BOOT RLDR/A)

loads the Relocatable Loader from a master reel and HALTs.

BOOT $TTR)

LOAD $TTR, STRIKE ANY KEY

loads an absolute binary tape from the teletypewriter reader.

BOOT CT1: 2/A)

causes file 2 on cassette unit 1 to be loaded and HALTs.

Name: BPUNCH

Format: BPUNCH filename$_1$ [filename$_2$ ...]

Purpose: To punch a given file or files in binary on the high speed punch. The command is the equivalent of a series of XFER commands:

XFER filename$_1$ $PTP;... ;XFER filename$_n$ $PTP)

The files may come from any input device.

Switches: None.

Examples: BPUNCH MT0:0 MT0:1 CT0:3 $PTR)

causes files MT0:0,MT0:1, CT0:3, and $PTR to be punched on the high speed punch.

BPUNCH $PTR)

causes a duplicate of the paper tape in the high speed reader to be punched.

CLI Commands (Continued)

Name: COPY

Format: COPY sourcefile destinationfile

where: sourcefile and destinationfile are cassette or magnetic tape files

Purpose: To copy a cassette or magnetic tape reel to another cassette or magnetic tape reel. The specified sourcefile denotes the beginning of the input reel. The specified destinationfile denotes the beginning of the output reel. Each subsequent file on the input reel is copied to a subsequent file on the output reel, until the end of reel (or double EOF mark) is detected on the input reel. The double EOFs are also copied to the output reel.

Switches:
Global: /V - Verify each file that is copied on the system console.

Examples: COPY CT0:0 CT1:0)

copies the entire reel on cassette unit 0 to the reel on cassette unit 1.

COPY/V MT1:3 MT0:5)

copies all files on magnetic tape unit 1, beginning at file 3, to magnetic tape unit 0, beginning at file 5. Each file that is copied is verified on the system console. The copy continues until a double EOF is encountered on unit 1.

Name: DEB

Format: DEB filename

Purpose: To debug a program which is about to be loaded and executed. The SOS Debugger, SDEB, or the RTOS/RDOS Debugger, IDEB, must have been included in the load module produced by the RLDR. The program save file must reside on magnetic or cassette tape. This command is identical to "BOOT" except that control is given to the address at the UST location USTDA instead of USTSA.

When debugging, memory can be examined, break points set, the program run, etc. After making any necessary changes in the program, the user can preserve the core image of the program by issuing the $V Debug command (SDEB only) and writing out the save file using the Core Image Writer.

Switches:

Global: None

Local: /A - HALT after loading the program; do not start the Debugger.

Example:

```
DEB CT1:5                 ← Debug CT1:5
1004/ ADD 0 2  ADD 2 0    ← Change program
$V                        ← $V break issued
#1:5                      ← Old save file is specified for the modified
                            program
NMAX: 16200               ← NMAX is specified
OK                        ← File has been rewritten to CT1:5
                          ← CONTINUE key causes return to Debugger
$R                        ← Program is started at the beginning address
```

Name: DUMP

Format: DUMP outputfilename filename$_1$ {filename$_2$ . . .},)

{oldfilename/S newfilename} . . .)

Purpose: To produce a dump format file (see DGC 093-000075, Appendix F) of a given file or files. This facility is desirable for SOS users who

- wish to back up cassette or magnetic tape files on paper tape.
- transfer SOS files to an RDOS system. The checksum information in dump files assures the exactness of the transfer.

When the local switch, S, is not used to dump a file, the name that is placed in the name block of the corresponding dump file record is the SOS filename, such as CT0:2, MT1:0, $PTR, etc.

Switches:

Global:

/L - list the dumped files on the $LPT (overrides /V).

/S - dump a file to paper tape ($TTP or $PTP) in segments. The file is punched in segments of up to 16K bytes each. An attempt to dump to a file other than $TTP or $PTP causes the error printout: ILLEGAL FILE-NAME

/V - verify dump with a list of names of dumped files on the console.

Local:

/S - assign the following name to the record output in the dump file.

Notes: The /S global switch is used to segment backup paper tapes that would otherwise be difficult or impossible to handle as a single tape, for example the FORTRAN IV compiler. Each segment of tape is punched with a unique segment number enabling the system to verify that tapes are later reLOADed in proper sequence. Appropriate leader and trailer and continuation blocks mark the segmentation of

Notes: (Continued)

the paper tape. It is the user's responsibility to separate the tapes after punching and mark them for his use. Failure to load the proper tape in the proper sequence causes an error message to be printed. The message is shown under Notes in the LOAD command.

Examples:

```
DUMP/L MT0:0 CT0:0/S FORT.LB CT1:0/S CILWCT.SV CT1:1/S PARU.SR)
```

causes CT0:0 to be dumped with the name FORT.LB, CT1:0 to be dumped with the name CILWCT.SV, and CT1:1 to be dumped with the name PARU.SR. All are dumped as the first file on magnetic tape unit 0. The dumped files will be listed on the line printer. They would be specified as FORT.LB, CILWCT.SV, and PARU.SR when reloaded by the LOAD command. (See LOAD command, page 3-28.)

```
DUMP MT0:0 CT1:23 CT2:4 )
```

causes the 24th file on cassette unit 1 and the 5th file on cassette unit 2 to be dumped onto the first file on magnetic tape unit 0. The names CT1:23 and CT2:4 will be retained on the dump file and when reloaded by the LOAD command would be specified as such on the console.

Name: FILCOM

Format: FILCOM filename$_1$ filename$_2$ {listing file/L}

Purpose: To compare two files, word by word, and print dissimilar word pairs. Dissimilar word pairs at the same displacements in the two files are printed on the console if the /L switch is not used. Words are printed in octal and their displacements in the file are also printed. File organizations of the two files may differ.

Switches:
Local: /L - listing file

Example: FILCOM MT0:0 MT1:3 $LPT/L)

causes a word-by-word comparison of the contents of the files MT0:0 and MT1:3. Any dissimilar word pairs will be printed in octal on the line printer, along with their respective word displacements in the files:

```
000025/   044516   042530
000141/   000014   020044
000142/   000000   046120
000143/   000000   052057
000144/   000000   046015
000145/   000000   000014
```

If MT1:3 were a null file, the entire contents of file MT0:0 would be printed. Dashes would be printed for file MT1:3.

If the files to be compared wraparound after the first 64K words compared, the beginning of the second 64K words compared (and any subsequent 64K words compared will be indicated by the following printout preceding the word pair comparisons:

PLANE nn

where: nn is two digits indicating which 64K words are being compared as follows:

01 - second 64K words compared
02 - third 64K words compared
:
:

Name: GTOD

Format: GTOD

Purpose: To get the time of day and the current date.

Switches: None.

Example:
```
GTOD )
2/17/73  21:24:20
R
```

The message indicates that the time is 20 seconds after 9:24 p.m., and the date is February 17, 1973.

Name: INIT

Format: INIT devicename

where devicename is one of CT0, CT1, ... CT7 or MT0, MT1, ... MT7

Purpose: Initialize cassette or magnetic tape. The cassette or magnetic tape unit specified is rewound. If there is no such unit in the system, the error message NOT A DIRECTORY DEVICE will be printed on the teletype.

Switches:

Global: /F - This switch causes two EOF records to be written to the appropriate cassette or magnetic tape at the beginning of the tape.

Local: None.

Examples: INIT CT1)

initializes cassette unit 1.

INIT/F MT0)

initializes magnetic tape unit 0 and writes two EOF records.

Name: LOAD

Format: LOAD inputfilename

Purpose: To restore a previously dumped file or files. This feature is desirable for SOS users who wish to:

- restore backed up (or dumped) SOS files.
- transfer dumped RDOS files to Stand-alone Systems for use there.

As the CLI encounters the name block of each record in the dump file, it types this name on the system console. The user then specifies the file to which this record will be restored in the following manner:

-) - use the filename typed on the console, as CT0:4

name) - use the (user) specified name, such as CT1:5 instead of CT0:4 (from above).

.) - use the previous output filename, incremented by one. For example, if CT1:5 were specified previously, this file would be restored to CT1:6. This response is obviously illegal for the first record encountered in the dump file.

) - skip this record and proceed to the next one.

As the dump file is processed by the CLI, only the desired records are selected for output. When all desired records have been restored, a CTRL A break may be entered to terminate the command.

If new names have been assigned to the files during the DUMP - such as FORT. LB (see DUMP command) - each new name will be typed out on the console as each record is read. In such cases, an appropriate output file must be specified.

Switches:

Global: /N - Only list the files on the console; do not load.

Local: None.

Notes: When reloading a paper tape that was dumped in segments (/S global switch to the DUMP command), a failure to load the appropriate tape in the appropriate order will cause the following printout:

YOU LOADED filename TAPE# nn)
I WANT filename TAPE# nn..... TRY AGAIN)

where filename is the name of the file, e.g., FORT.LB.
nn is the number of the segment, e.g., 01, 02, 03, ...

In the case where a second tape is loaded before the first tape, the filename will appear only in the first line of the message:

YOU LOADED filename TAPE# 02)
I WANT TAPE# 01..... TRY AGAIN)

Examples:

```
LOAD $PTR )

(FORT.LB) CT2:0 )        ;restore file to CT2:0
(SOSCT.LB) . )           ;restore file to CT2:1
(SOS.LB) CT1:2 )         ;restore file to CT1:2
R                        ;end of dump file

LOAD MT0:0 )

(MT0:00) MT1:0 )         ;restore file to MT1:0
($PTR) )                 ;skip this file
(PARU.SR) $PTP )         ;restore file to high speed punch
(MT1:1) - )              ;restore file to MT1:1
R                        ;end of dump file

LOAD/N CT1:2 )
ABC.SR                   ;each dumped file is printed
DEF.SR                   ;on the system console
CT2:3
JKL.SR
R                        ;end of dump file
```

Name: MKABS

Format: MKABS save filename absolute binary filename

Purpose: To make an absolute binary file from core image (save) file.

Switches:

Global: /S - starting address switch. The starting address of the save file as specified in USTSA of the file will be used as the address for an absolute binary start block. Default is a null start block which causes the binary loader to halt when loading is complete.

Local: /F - first address (relative to location 0 of the save file) from which the absolute binary file is to be created.

/S - starting address switch. An absolute binary start block will be output with the starting address specified by the preceding octal argument.

/T - last address (relative to location 0 of the save file) which is to become part of the absolute binary file.

Examples: MKABS MT0:0 $PTP)

punches an absolute binary file to the paper tape punch from file MT0:0

MKABS MT0:0 $PTP 1000/S)

punches an absolute binary file with a start block specifying 1000 as the starting address.

MKABS/S CT2:1 $TTP 0/F 10000/T)

punches an absolute binary file with a start block which contains the address specified in USTSA. The first 10001 octal locations of the save file are included in the absolute binary file that is generated at the teletypewriter punch.

CLI Commands (Continued)

Name: MKSAVE

Format: MKSAVE absolute-binary-filename save-filename

Purpose: To create a core image (save) file from an absolute binary file.

Switches: None.

Example: MKSAVE $PTR CT0:2)

causes creation of a core image file on CT0:2 from the absolute binary file loaded in the paper tape reader.

Name: PRINT

Format: PRINT filename$_1$ [filename$_2$ ...]

Purpose: To print the given file or files on the line printer. The command is the equivalent of a series of XFER commands.

XFER/A filename$_1$ $LPT; . . . ;XFER/A filename$_n$ $LPT)

The source files may come from any input device. If a parity error is detected, a left slash (\) is printed in place of the bad character, and the message "PARITY ERROR" is output on the console; printing then continues.

Switches: None.

Example: PRINT MT0:2 MT1:0 CT0:3 $PTR)

causes source files MT0:2 MT1:0 CT0:3 and the paper tape mounted in the high speed reader to be printed on the line printer.

Name: PUNCH

Format: PUNCH filename$_1$ [filename$_2$ . . .]

Purpose: To copy a given file or files on the high speed punch. The command is the equivalent of a series of XFER commands:

XFER/A filename$_1$ $PTP; . . .;XFER/A filename$_n$ $PTP)

The source files may come from any input device. If a parity error is detected, a left slash (\) is punched in place of the bad character, and the message "PARITY ERROR" is output; punching continues.

Switches: None.

Examples: PUNCH CT0:0 $PTR $TTR)

causes files CT0:0, $PTR, and the paper tape mounted in the teletypewriter reader to be punched on the high speed punch.

Name: RELEASE

Format: RELEASE devicename

where: devicename is one of CT0, CT1, ... CT7, or MT0, MT1... MT7

Purpose: To release a cassette or magnetic tape. The cassette or magnetic tape unit specified is rewound. If there is no such unit in the system, the error message NOT A DIRECTORY DEVICE will be printed on the teletypewriter.

Switches: None.

Examples: RELEASE CT7)

rewinds cassette unit 7.

RELEASE MT4)

rewinds magnetic tape unit 4.

CLI Commands (Continued)

Name: SDAY

Format: SDAY month day year

Purpose: To set the system calendar. Only spaces or commas can be used to separate the date. This date is never changed by the system.

Switches: None.

Example: SDAY 4 17 74)

sets the system calendar to April 17, 1974.

Name: STOD

Format: STOD hour minute second

Purpose: To set the system clock. (The system clock is a 24 hour clock.) Only spaces or commas can be used to separate the time arguments.

Switches: None.

Example: STOD 21 24 0)

causes the system clock to be set to 9:24:00 p. m.

Name: TYPE

Format: TYPE filename$_1$ [filename$_2$ . . .]

Purpose: To copy a given file or files on the program console. The command is the equivalent of a series of XFER commands:

XFER/A filename$_1$ $TTO;. . . ;XFER/A filename$_n$ $TTO)

The source files may come from any device. When a character with bad parity is detected, a left slash (\ ) is typed and the message "PARITY ERROR." is output; typing of the remainder of the file continues.

Switches: None.

Example: TYPE MT0:1 MT0:2 $PTR CT0:3)

This command causes the following files to be typed or displayed on the program console: tape files MT0:1 and MT0:2, a file mounted in the high speed reader, and cassette file CT0:3.

Name: XFER

Format: XFER sourcefile destinationfile

Purpose: To copy a file. The system will detect parity errors in ASCII files, and it will detect parity errors in binary files on magnetic or cassette tape. When a parity error is detected in the input file, the message PARITY ERROR, FILE: xxx will be output to the console. If one or more parity errors are detected in a paper tape file, a backslash will be substituted for each bad character. If a parity error is detected in a magnetic tape or cassette file, the transfer is terminated.

Switches:

Global : /A - ASCII transfer. Transfer the file line by line taking appropriate read/write action, such as inserting line feeds after each carriage return when transfer is from cassette to line printer.

/I - Ignore input parity on ASCII transfer. The switch is only meaningful when used with the /A switch. Normally, parity is checked as described under Purpose.

Local: None.

Examples: XFER $PTR MT0:3)

causes the file in the paper tape reader to be transferred to the tape file MT0:3.

XFER/A $PTR $LPT)

causes $PTR to be printed on the line printer.

XFER $PTR $PTP)

causes another tape to be punched, identical to the one read from the paper tape reader.

XFER CT0:3 CT3:4)

transfers file 3 on cassette unit 0 to file 4 on cassette unit 3.

OPERATION OF SOS UTILITY PROGRAMS

The Relocatable Loader (RLDR)

Format: (RLDR) filename$_1$/S filename$_2$ [filename$_3$...]

Note that the name of the utility appears above in parentheses. This means that the user does not type RLDR as part of the command line; RLDR is a prompt given by the Relocatable Loader indicating that the system utility is ready to accept a command line, consisting of names of files to be loaded.

Purpose: To produce an executable core-resident file and a core-image (or save) file from relocatable binary files. Both the core-resident and core-image file start at address zero. The symbol table is included in the files only if SOS Debug has been loaded and then the table always follows the program immediately.

The debugger provided for cassette and magnetic tape systems is SOS Debug, found on relocatable binary tape 089-000167.

The loading of user ZREL begins at location 50_8, and user NREL loading starts at location 440_8. Locations 400-437 contain the User Status Table. Location 405_8 contains the starting address of the loaded program.

The maximum size of each loaded program cannot exceed the maximum core address less 1325_8. The relocatable loader will type symbol tables for programs whose size exceeds that maximum value, but the loading process will not be completed for such programs.

Upon completion of each successful load, the message "OK" is output and the system halts with the loaded program in core.

Switches:
(Local)

/A causes a listing of the symbol table with symbols ordered alphabetically. (The local switch /L must also be given to define an output listing device.)

/L causes a listing of the symbol table on the output file or device whose name precedes the switch. Symbols in the table will be ordered numerically by symbol value.

Switches: (Local) Continued

/N set current NMAX (the starting load address) for the file name following the switch to be forced to the absolute address immediately preceding this switch.

/P pause before opening this file.

/S output the save file to the file (either cassette or magnetic tape unit) whose name precedes this switch. (Must always be present.)

/U load user symbols appearing within the file whose name precedes this switch.

/n load the preceding file n times, where n is a digit from 2 to 9.

Errors:

NO INPUT FILE SPECIFIED.

NO SAVE FILE SPECIFIED.

No core image output device has been specified with a /S switch.

SAVE FILE IS READ/WRITE PROTECTED

The save file must permit both reading and writing; only cassette and magnetic tape units are permitted as save file devices.

I/O ERROR nn

where: nn is one of the following error codes:

1	-	ILLEGAL FILE NAME
7	-	ATTEMPT TO READ A READ-PROTECTED FILE
10	-	WRITE-PROTECTED FILE
12	-	NON-EXISTENT FILE

Example:

```
$TTO/L/A CT2:0/S $PTR CT1:6 16500/N CT1:0 )
```

causes the relocatable loader to load programs into core from the paper tape reader ($PTR), file 6 on cassette drive 1(CT1:6), and file 0 on cassette drive 1 (CT1:0). The program from CT1:0 is loaded starting at address 16500_8 (16500/N). A core image file containing the loaded programs is written to file 0 of cassette unit 2 (CT2:0/S). An alphabetically ordered symbol table is output on the teletype printer ($TTO/L/A).

Text Editor

Format: Not applicable.

Purpose: To create or modify ASCII files. The prompt given by the Text Editor when it is ready to accept editing commands is: *

The text editing commands are described in the Text Editor User's Manual, 093-000018. Note, as described there, that all output files must be closed with the GC command. Execution of P or E commands does not insure that the final input page will be written onto the output file.

Switches: Not applicable.

Errors: See errors given for the Core Image Loader command and those given in the Text Editor Manual, Chapter 5.

Example: After loading and activating the Text Editor, input and output files must be assigned, for example:

```
*
GRCT2:3$CWCT1:3$$
```

causes a source file to be read from file 2 of cassette unit 3 and an output file to be written as file 3 of unit 1.

Extended Assembler

Format: (ASM) 0 filename$_1$. . .

(ASM) 1 filename$_1$. . .

(ASM) 2 filename$_1$. . .

The parenthesized utility file name (ASM) indicates that ASM is a prompt issued by the utility when ready to accept a user command line and is not typed by the user.

Purpose: To assemble one or more ASCII source files. Output may be an absolute or relocatable binary file, with an optional listing file. Input files are assembled in the order specified in the command line. A cassette or magnetic tape unit may not be used for both input and output, nor may it be used for more than one output file. A cassette or magnetic tape unit may be used for more than one input file.

In the above formats, 0, 1, and 2 are keys describing the number of passes required. Global switches listed below modify the key in a given command line. Action taken by the assembler depends upon the key specified in the command line:

0 - Perform pass one on the specified input source file(s), then halt with the highest symbol table address (SST) in AC0. Normally the source file is a Command Definition tape. The user may then invoke the core image writer to preserve a copy of this assembler on any cassette file.

1 - Perform pass one and pass two on the specified input files, producing the specified binary and listing files. At the completion of pass two, the assembler outputs a new prompt, ASM, and awaits a new command line.

2 - Perform pass two only on the specified input files producing the specified binary and listing files. The symbol table used for this pass is that produced by the most recently executed pass one assembly. At the completion of this pass, the assembler outputs a new prompt, ASM, and awaits a new command line.

Switches:

Global:

/E suppress assembly error messages normally output to the $TTO.

/T suppress the listing of the symbol table.

/U include local (i.e., user) symbols in the binary output file.

Local:

/B relocatable or absolute binary file is output on the given device.

/L any output device to which the listing is directed.

/N any input file which is not to be listed on pass 2.

/P pause before accepting a file from a device. The message:

PAUSE - NEXT FILE, filename

is output by the assembler which waits until any key is struck on the teletypewriter console.

/S skip this source file during pass two.

/n repeat the given input source file n times, where n is a digit from 2 to 9.

Errors:

NO .END

No .END statement in any source file.

I/O ERROR nn

where: nn is one of the following error codes:

1	-	ILLEGAL FILE NAME
7	-	ATTEMPT TO READ A READ-PROTECTED FILE
10	-	WRITE-PROTECTED FILE
12	-	NON-EXISTENT FILE

Examples: 1 CT1:0/B $LPT/L CT0:0/S CT0:1 CT0:2/N)

causes a two-pass assembly to be executed using cassette files 0,1, and 2 on unit 0 as input files. A binary file is produced on unit 1, file 0, and a listing file is printed on the line printer. On pass 2, input file CT0:0 is skipped, and input file CT0:2 is not listed.

2/U CT0:13 CT0:14 CT0:18 CT0:8 CT1:1/B CT2:0/L)

causes the second pass of an assembly to be executed on input files 13, 14, 18 and 8 (in that order) on cassette unit 0. The binary, containing user symbols, is produced on file 1 of cassette unit 1, and the listing is produced on file 0 of cassette unit 2.

1/E CT0:16 CT0:17 CT1:0 CT1:1 $LPT/L)

causes a two-pass assembly to be executed on input files CT0:16, CT0:17, CT1:0, and CT1:1 with a listing printed on the line printer. Error messages normally output to the $TTO are suppressed, and no binary file is produced.

1 CT0:0/S CT0:0/P CT1:1/B)

causes a two-pass assembly to be performed on two files, both of which will be read from unit 0. The first file is a parameter list which will be read during the first pass only. After this parameter list is read, the pause message is output, and a new file is placed in cassette unit 0. The first file of this new reel is scanned for both pass 1 and pass 2 to complete the assembly, File 1 of unit 1 receives the binary output; no listing is produced.

Library File Editor

Format:

(LFE) A {listingname/L} inputname{/B}{/#}... {recordname{/R}})

(LFE) D inputname/I outputname/O recordname {recordname...})

(LFE) I {inputname/I} outputname/O {insertname{/#}...}+)
{recordname/A insertname{/#}...}...)
{recordname/B insertname{/#}...}...)

(LFE) M outputname/O inputname{/B}{/#}... {inputname{/B}{/#}}...)

(LFE) R inputname/I outputname/O recordname updatename)
{recordname updatename}...)

(LFE) T {listingname/L} inputname{/B}{/#}... {inputname{/B}{/#}}...)

(LFE) X inputname/I outputname/O recordname)

where: inputname is the name of the input file which contains the relocatable binaries or libraries;

recordname is the name of the particular logical record to be analyzed, replaced, extracted, or before or after which insertions will occur;

listingname is the name of the file used to output the analysis;

outputname is the name of the file which will contain the new library;

insertname is the name of the file containing relocatable binaries which are to be inserted;

updatename is the name of the file which will replace the record.

The parenthesized utility file name (LFE) indicates that LFE is a prompt issued by the utility when ready to accept a user command line and is not typed by the user.

Purpose: To analyze the contents of a library file, to list titles in a library file, to merge libraries, to update libraries, to extract logical records from a library file, and to create new libraries.

A library file (or library) is comprised of a set of relocatable binary programs, called logical records. This collection of logical records

Purpose: (Continued)

is denoted by a special beginning and ending block. Selection of any program for loading is triggered by the occurrence of a global entry within the program that resolves an external declaration within a previously loaded program. (See the Library File Editor Manual, 093-000074, for a complete description of this utility.)

The key letters that may follow the prompt, LFE, and their meanings are:

Key Letter	Command
A	Analyze a set of library files and/or binaries, or analyze selected records in a library.
D	Delete logical records from a library.
I	Insert binaries into either a new or an existing library.
M	Merge a library and binaries to form a new library.
R	Replace logical records in a library with new binaries.
T	List titles in a set of libraries or binaries.
X	Extract specific logical records from a library.

Record Names:

Record names are those assigned to logical records by the .TITL pseudo-op. Only the first five characters are meaningful.

Switches:

Local:

/A - After switch naming the logical recordname after which insertions will occur. (Insert command.)

/B - Before switch naming the logical recordname before which insertions will occur. (Insert command.)

or

Binary record switch, indicating that the modified name represents a relocatable binary file rather than a library file. (Analyze, merge, and titles commands.)

/I - Input library file switch, specifying the file which contains the existing library. (Delete, insert, replace and extract commands.)

/L - Listing switch, indicating the file will be used to output the listing. (Analyze and titles commands.)

Switches:

Local: (Continued)

/O - Output library file switch naming the new library file. (Delete, insert, merge, replace, and extract commands.)

/R - Record switch. The logical recordname preceding the switch will be analyzed. (Analyze command.)

/# - Number switch, indicating how many binary records or libraries will be read. (Modifies $PTR or $TTR files only in analyze, insert, merge, and titles commands.)

Examples: A $LPT/L $PTR ABC/R CDE/R

causes logical records ABC and CDE, in a library loaded on the high speed reader, to be analyzed. The analysis will be output on the line printer.

D $PTR/I $PTP/O ABC DEF

causes logical records ABC and DEF to be deleted from the input library file, with the resulting library output on the high speed punch.

I $PTP/O $PTR/3

causes three relocatable binary records mounted on the high speed reader to be punched as a library file by the high speed punch.

I $PTR/I $PTP/O M/A $PTR

causes a library to be updated with binaries. If the input library consists of logical records L, M, and N and the update record is Q, Q is inserted after record M. The result is an updated library consisting of records L, M, Q, and N. The new library is punched by the $PTP and the high speed reader is the single input device.

M CT1:23/O $PTR/3 $PTR/B/2

causes three libraries to be input, then two binaries, forming a new library output as file number 23 on cassette drive 1.

R $PTR/I $PTP/O REC1 $PTR

causes a library file, input on the high speed reader, to have one of its logical records (REC1) replaced by an update input via the high speed reader. The new library will be punched on the high speed punch

Examples: (Continued)

T $PTR

causes the titles of all logical records in the library file mounted in the high speed reader to be printed on the teletypewriter printer by default.

X $PTR/I $PTP/O ABC

causes logical record ABC to be extracted from its library file, which is input on the high speed reader. Record ABC is a binary punched on the high speed punch.

SYSGEN

Format: (SYSG) driver$_1$...driver$_n$ [,RDSI] [,CTB] outputfile/O trigger/T

The parenthesized utility name (SYSG) is a prompt issued by SYSGEN when the utility is ready to accept a command line and is not part of the command line typed by the user.

Purpose: SYSGEN generates triggers for use in configuring SOS utility programs. Each driver in the command line is an entry symbol for a program in one of the SOS libraries. (See Table 2-2.) For each of these entry symbols included in the command line, an external normal reference to the program is included in the trigger. The trigger is entirely made up of these external normal references.

Switches:

Local:

/O specifies the output file.

/T specifies the preceding name as the title of the trigger. If this switch is omitted, the title "SGTRG" is given to the relocatable binary file produced.

Errors: NOT ENOUGH ARGUMENTS

At least two arguments are needed.

OUTPUT FILE WRITE PROTECTED, FILE: filename

NO OUTPUTFILE SPECIFIED

ILLEGAL SYMBOL NAME: symbolname

Invalid character in the command line.

FILE DOES NOT EXIST, FILE: filename

UNEXPECTED SYSTEM ERROR

Computer halts with the system error code in AC2.

SYSGEN (Continued)

Example: .PTRD .PTPD .MTU4 .STTY .RDSI MT1:3/O TRIG/T

would cause the SYSGEN utility to output a trigger named TRIG containing external normal references to the driver routines for the high-speed paper tape reader, high-speed paper tape punch, magnetic tape unit 4, teletypewriter keyboard and printer, and to the routine for the RDOS to SOS interface (.RDSI).

The FORTRAN IV Compiler (FORT)

Format: (FORT) filename$_1$ ...

The parenthesized utility file name (FORT) indicates that FORT is a prompt issued by the utility when ready to accept a user command line and is not typed by the user.

Purpose: To compile one or more FORTRAN source files, producing an output file suitable for input to the Extended Assembler. An output file and a listing file may optionally be specified in the command line using local switches described below. The command line must contain at least one input file name.

If more than one input file name is specified, the message:

TO CONTINUE, STRIKE ANY KEY

is typed on the teletypewriter whenever one of the intermediate files has passed through the compiler. The next input file must be ready for opening when the user strikes the key. No other prompt messages are output for intermediate input files.

Input files are compiled in the order in which they are specified in the command line. At the completion of each compilation, the prompt:

FORT

is again typed at the console. The prompt is reissued if no input file name is found in the command line.

If the last specified input file does not contain an END statement, the message:

END OF FILE

is typed at the console and the compiler must be restarted at location 377.

Switches:

Local:

/L causes listing to be directed to the file named.

/O causes output to be directed to the file named.

Switches:

Local:

/S causes FORTRAN IV variables and statement numbers to be equivalenced to symbols acceptable to the assembler. (This switch must modify the output file name.)

/X causes compilation of statements with an X appearing in column 1 of the source line. (This switch must modify the output file name.)

/n compile the preceding input file n times.

Errors: FATAL I/O ERROR xx

This message is typed at the console if an unexpected system error occurs. xx is one of the two-digit error codes defined in the SOS User Parameter Tape, PARU.SR.

Examples: $PTR/2 $PTP/O $LPT/L)

causes FORTRAN IV to compile two source programs from the paper tape reader, outputting the assembler source code to the paper tape punch and a listing to the line printer.

MT0:2 MT0:3 MT1:0/O)

causes FORTRAN IV to compile source programs on files 2 and 3 of magnetic tape unit 0, outputting the assembler source code to file 0 of magnetic tape unit 1.

CHAPTER FOUR

PROGRAM MODE OF SYSTEM COMMUNICATION

SYSTEM COMMAND WORDS

Users communicate with the Stand-alone Operating System using system command words assembled into their programs. System command words and the mnemonic .SYSTM that must precede each command word are recognized as legal mnemonics by the SOS and RDOS Assemblers.* Appearance of the mnemonic .SYSTM in a program results in the assembling of a JSR @ 17 instruction which allows system communication through the main system entry address stored in page zero. For this reason, all assembly langugage programs issuing system calls must be assembled using one of the standard DGC Assemblers in order to run under SOS. The system command word must be assembled as the word following .SYSTM.

Once system action is complete, normal return is made to the second instruction after the command word. If an exceptional condition is detected, return is made to the first instruction following the system command word.

The general form of a system call in a program is:

```
.SYSTM
command
exceptional return      ;STATUS IN AC2
normal return           ;AC's & carry preserved or information
                        ;returned as specified for the particular
                        ;command.
```

COMMAND WORD FORMAT

There are two basic command word formats:

command n and command

In the first format, n is a digit (from 0 to 76) representing an I/O channel number or n specifies the CPU (77) as described in the next paragraph. The channel number n (0-76) indicates a logical link to an opened file. When no I/O channel is needed in command execution, the command word appears alone in the instruction. If the command requires arguments, these are passed in the accumulators.

* DGC Extended Assembler Manual, 093-000040

DGC Macro Assembler Manual, 093-000081

COMMAND WORD FORMAT (Continued)

Any system command requiring a channel number n need not specify this number in the command itself. By specifying octal 77 (the device code of the CPU) as the channel number in the instruction, the system will use instead the number passed in AC2. For example, the following instructions specify a write to channel 3:

```
        LDA  2, C3
        .SYSTM
        .WRS  77
        JSR  ERR
          .
          .
          .
C3:  3
```

(Standard SOS recognizes only SOS channels 0-37 unless the SOS Channel Number to Device Map has been expanded; see Appendix A).

STATUS ON RETURN FROM SYSTEM

Status of the accumulators upon return from the system is as follows:

If the system returns no information as a result of the call, the carry and all accumulators except AC3 will be preserved. AC2 is used when an exceptional return is made to return a numeric error code. Error codes are listed by number at the end of this chapter and the applicable codes are listed for each command.

AC3 is destroyed by a .SYSTM call (as it is by the use of JSR). On return from the system however, AC3 is loaded with the contents of memory location 16. This location is defined as a permanent symbol by the SOS Assembler and has the name USP (User Stack Pointer). A convenient method of saving AC3 is to store it in USP before issuing the .SYSTM call. The system will load AC3 with the contents of USP upon return from the call.

The SOS commands are shown in Table 4-1. They represent a subset of the RDOS system commands. All other RDOS system commands, including .RTN, result in an error return with error code:

AC2	Mnemonic	Meaning
2	ERICM	Illegal system command

.CLOSE	Close a file.
.DELAY	Delay the execution of a task.
.EOPEN	Open a file for reading and writing.
.GCHAR	Get a character from the console.
.GCHN	Get the number of a free channel.
.GDAY	Get today's date.
.GHRZ	Examine the real time clock.
.GTATR	Get the file attribute.
.GTOD	Get the time of day.
.IDEF	Identify a user device.
.IRMV	Remove a user device.
.MEM	Determine available memory.
.MEMI	Allocate an increment of memory using NMAX.
.OPEN	Open a file for reading and writing.
.PCHAR	Output a character from the console.
.RDL	Read a line.
.RDS	Read sequential bytes.
.RESET	Close all files.
.ROPEN	Open a file for reading and writing.
.SDAY	Set today's date.
.STOD	Set the time of day.
.SYSI	Initialize SOS devices.
.WRL	Write a line.
.WRS	Write sequential bytes.

Table 4-1. SOS Commands

BYTE POINTER

One argument commonly passed in an accumulator is a byte pointer. A byte pointer contains the word address, in bits 0-14, which contains or will receive the byte. Bit 15 specifies which half (0 left; 1 right). Note that this is the reverse of the byte pointer as specified in "How to Use the Nova Computers." To use the subroutine shown on page 2-21 of the manual, make the following change:

```
Replace:    MOV  0  0  SZC

With:       MOV  0  0  SNC
```

The ".OPEN" command requires file names to be specified by means of a byte pointer to the name. The file name is stored as a character string. The string must consist of characters packed left to right (.TXTM 1) with the high order bit of each byte equal to zero. The string must have a terminating byte containing one of the following characters: null (000), form feed (014), carriage return (015), or space (040).

For example, the word at location BPTR contains a byte pointer to a properly specified file name, $CDR:

```
BPTR:       2*NAME
            .TXTM 1
NAME:       .TXT  *$CDR*
```

INITIALIZATION OF COMMUNICATIONS (.SYSI)

The .SYSI command must be issued before any other SOS commands are used. It initializes all tables, clears each SOS device, restores NMAX to its value at load time, and enables interrupts from all devices. Additional .SYSI commands may be issued if the user wishes to clear devices on restart.

The format of the command is:

```
.SYSTM
.SYSI           ;INITIALIZE SOS
error return    ;NEVER TAKEN
normal return   ;AC's AND CARRY PRESERVED
```

FILE ATTRIBUTE COMMAND (GTATR)

The file attribute command allows the user to determine the current attributes of a file or device. The bit settings of AC0 and AC1 determine the file attributes and device characteristics respectively.

This command obtains the attributes of a file and device characteristics. To obtain attributes, the file must be opened (see .OPEN). The number of the channel is given in the system command. The format of this command is:

```
.SYSTM
.GTATR n
error return
normal return
```

Upon return, AC0 contains the file attributes. The bit/attribute correspondence used in interpreting the bit configuration returned in AC0 is shown below:

Bit	Mnemonic	Meaning
1B0	ATRP	Device is read-protected.
1B1	ATCHA	Device is attribute-protected; has unsuppressable device characteristics (e.g., the $TTI is attribute-protected; therefore it always has the DCKEY characteristic while it is open.).
1B14	ATPER	Permanent file. All devices are permanent files.
1B15	ATWP	Write-protected file.

FILE ATTRIBUTE COMMAND (.GTATR) (Continued)

AC1 contains the device characteristics of the file. These do reflect the characteristic inhibit mask supplied when the file was opened. The bit/characteristics correspondence used in interpreting the bit configuration returned in AC1 as shown below:

Bit	Mnemonic	Meaning
1B0	DCDIR	Directory device -i.e., a data channel block transfer device.
1B1	DCC80	80-column device.
1B3	DCFFO	Device requiring form feed on opening.
1B4	DCFWD	Full word device (reads or writes more than a byte.)
1B6	DCLAC	Output device requiring line feeds after carriage returns.
1B7	DCPCK	Input device requiring a parity check; output device requiring parity to be computed.
1B8	DCRAT	Output device requiring a rubout after every tab.
1B9	DCNAF	Output device requiring nulls after every form feed.
1B10	DCKEY	A keyboard input device.
1B11	DCTO	A teletype output device.
1B12	DCCNF	Output device without form feed hardware.
1B13	DCIDI	Device requiring operator intervention.
1B14	DCCGN	Output device without tabbing hardware.
1B15	DCCPO	Output device requiring leader/trailer.

Possible errors resulting from a .GTATR command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
15	ERFOP	Attempt to get attributes of an unopened file.

INPUT/OUTPUT COMMANDS

All I/O is handled by system I/O commands. These commands require a channel number (0-76)* to be given in the second field of the command word. If the channel number is 77, then AC2 must contain the desired channel number. A channel is linked to a particular file or device by means of a .OPEN, .EOPEN, or .ROPEN command. The association between a file and a channel number can be broken by using the .CLOSE command. All currently open files can be closed by using the .RESET command.

The system provides two basic modes of reading and writing files. The first mode is line mode, where data read or written is assumed to consist of ASCII character strings terminated by either carriage returns, form feed characters, or nulls. Position within a file is implicit from the last call. That is, file data is processed line by line in sequence from the beginning of the file to its end.

In line mode, the system handles all device dependent editing at the device driver level. For example, line feeds are ignored on paper tape input devices and are supplied after carriage returns to all paper tape output devices. Furthermore, reading and writing never require byte counts, since reading continues until a terminator is read and writing proceeds until a terminator is written. The line mode commands are .RDL and .WRL.

The second mode is unedited sequential mode. In this mode, data is transmitted exactly as read from the file or device. No assumption is made by the system as to the nature of this information. Thus this mode would always be used for processing binary files. This mode requires the user program to specify byte counts necessary to satisfy a particular read or write request. As with line mode, position within a file is determined by the position at the end of the last sequential mode call, and the first read or write occurred at the beginning of the file after it was first opened. The sequential mode commands are .RDS and .WRS.

*Standard SOS recognizes only SOS channels 0-37 unless the SOS Channel Number to Device Map has been expanded; see Appendix A.

Open a File (.OPEN, .EOPEN, or .ROPEN)

Before other I/O commands can be used, a file must be linked to a SOS Channel Number. .OPEN links a file with a channel number and makes the file available to any user for both reading and writing.* .EOPEN and .ROPEN will achieve exactly the same results and have been included for the sake of RDOS compatibility.

When running with RDOS-SOS, AC0 must contain a byte pointer to the file name. When running with STANDARD SOS, AC0 never contains an argument - unless the file being opened is a cassette or magnetic tape unit. In that case AC0 contains the file number.

A characteristic mask must be passed in AC1. (If AC1 contains 0, no characteristics are inhibited. Also, if the device is attribute protected, the mask is ignored.) For every bit set in this word, the corresponding device characteristic (as defined previously in the discussion of the .GTATR command) is inhibited. The characteristics will be inhibited for the duration of the .OPEN. For example, if the user has an ASCII tape without parity to be read from the paper tape reader, he may inhibit parity checking by the following (when running under RDOS-SOS):

```
        LDA     0, READR
        LDA     1, MASK
        .SYSTM
        .OPEN   3

READR:  .+1*2
        .TXT    *$PTR*
MASK:   DCPCK                   ;PARITY CHARACTERISTIC
```

In general, the user will wish to preserve all device characteristics as defined by the system. This can be accomplished by a SUB 1, 1 instruction before the system call instruction.

The .OPEN command results in the initialization of the control table for the device, the output of leader on paper tape output devices, or a prompt message for input devices requiring user intervention: "LOAD filename, STRIKE ANY KEY".

*When running under Standard SOS, this link is already established; the user need only pass the physical SOS Channel Number according to the procedures given here.

Open a File (.OPEN , .EOPEN , or .ROPEN) (Continued)

The format of the .OPEN command is:

```
.SYSTM
.OPEN n                 ; OPEN CHANNEL n
error return
normal return
```

Possible errors resulting from .OPEN commands are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
1	ERFNM	Illegal file name.
12	ERDLE	File does not exist.
21	ERUFT	Attempt to use channel already in use.
31	ERSEL	Unit improperly selected.

Get the Number of a Free Channel (.GCHN)

This call enables the user to obtain the number of a channel that is currently unused, if any, so that a file may be opened on this channel via one of the file open calls. .GCHN does not open a file on a free channel; it merely indicates a channel that is free at the moment. RDOS-SOS returns the lowest available channel number in the range 0-76. This command is illegal when running under Standard SOS.

The format of this call is:

```
.SYSTM
.GCHN
error return
normal return
```

Upon a normal return, the information is returned in AC2:

AC2 - Free channel number

One possible error return may occur.

AC2	Mnemonic	Meaning
21	ERUFT	No channels are free.

Close a File (.CLOSE)

After use, files may be closed to insure an orderly ending sequence. The channel number is then available for other I/O. The format of the .CLOSE command is:

```
.SYSTM
.CLOSE n                ;CLOSE CHANNEL n
error return
normal return
```

If the file closed requires trailer (such as the high speed punch) it will be output on the .CLOSE.

Possible errors resulting from a .CLOSE command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal number.
15	ERFOP	Attempt to reference a channel not in use.

Close all Files (.RESET)

This command causes all currently open files to be closed. The command also insures that any partially filled buffers are written before the file is closed. The format of the .RESET command is:

```
.SYSTM
.RESET
error return
normal return
```

The error return from this command is never taken.

Read a Line (.RDL)

This command causes an ASCII line, written with even parity, to be read. AC0 must contain a byte pointer to the starting byte address within the user area into which the line will be read. This area should be 133 bytes long.

Reading will terminate normally after transmitting either a carriage return, form feed, or null to the user. Reading will terminate abnormally after transmission of 132 (decimal) characters without detecting a carriage return, form feed, or null as the 133 rd character, upon detection of a parity error, or upon an end-of-file (CTRL Z).

Read a Line (.RDL) (Continued)

In all cases, the read byte count including the carriage return, form feed, or null will be returned in AC1. If the read is terminated because of a parity error, the character having the incorrect parity will be stored with the parity bit cleared as the last character read. The byte pointer to the character in error can always be computed as:

C(AC0)*+C(AC1)-1

The format of the .RDL command is:

```
.SYSTM
.RDL n                    ;READ FROM CHANNEL n
error return
normal return
```

Possible errors resulting from a .RDL command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
6	EREOF	End of file.
7	ERRPR	Attempt to read a read-protected file.
15	ERFOP	Attempt to reference a file not opened.
22	ERLLI	Line limit (132 characters) exceeded.
24	ERPAR	Parity error.
30	ERFIL	File read error.

* $C(AC_n)$ means "contents of AC_n"

Read Sequential (.RDS)

Sequential mode transmits data exactly as read from the file. AC0 must contain a byte pointer to the starting byte address within the user area into which the data will be read, and AC1 must contain the number of bytes to be read. The format of the .RDS command is:

```
.SYSTM
.RDS n                    ;READ FROM CHANNEL n
error return
normal return
```

Possible errors resulting from a .RDS command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
6	EREOF	End of file.
7	ERRPR	Attempt to read a read-protected file.
15	ERFOP	Attempt to reference a file not opened.
30	ERFIL	File read error.

Upon detection of an end-of-file, the partial read count will be returned in AC1.

Use of the Card Reader ($CDR) in .RDL and .RDS Commands

When using $CDR (card reader) as an input device, the end of file condition of a .RDL will occur only if a special end of file code is detected in column 1 of a card. This code is all rows punched. It can be punched on a 029 keypunch by multipunching "+", "-", and "0" through "9". Note that a Hollerith to ASCII translation only occurs if a .RDL has been requested. The translation assumes 029 keypunch codes. A .RDL is terminated upon the first trailing blank (which is translated as a carriage return). If all 80 columns are data, a carriage return is appended as the eighty-first character. If an illegal character is detected, a back slash is substituted and the line is terminated. A table of Hollerith - ASCII translation is given on the following two pages.

If a .RDS is given, the card is read in image binary. Each two bytes will be used to store a single column.

CHAR.	CARD CODE	ASCII CODE
NUL	12-0-9-8-1	000
SOH	12-9-1	001
STX	12-9-2	002
ETX	12-9-3	003
EOT	9-7	004
ENQ	0-9-8-5	005
ACK	0-9-8-6	006
BEL	0-9-8-7	007
BS	11-9-6	010
HT	12-9-5	011
LF	0-9-5	012
VT	12-9-8-3	013
FF	12-9-8-4	014
CR	12-9-8-5	015
SO	12-9-8-6	016
SI	12-9-8-7	017
DLE	12-11-9-8-1	020
DC1	11-9-1	021
DC2	11-9-2	022
DC3	11-9-3	023
DC4	4-8-9	024
NAK	9-8-5	025
SYN	9-2	026
ETB	0-9-6	027
CAN	11-9-8	030
EM	11-9-8-1	031
SUB	9-8-7	032
ESC	0-9-7	033
FS	11-9-8-4	034
GS	11-9-8-5	035
RS	11-9-8-6	036
US	11-9-8-7	037

CHAR.	CARD CODE	ASCII CODE
SPACE	NO PUNCHES	040
!	12-8-7	041
"	8-7	042
#	8-3	043
$	11-8-3	044
%	0-8-4	045
&	12	046
'	8-5	047
(	12-8-5	050
)	11-8-5	051
*	11-8-4	052
+	12-8-6	053
,	0-8-3	054
-	11	055
.	12-8-3	056
/	0-1	057
0	0	060
1	1	061
2	2	062
3	3	063
4	4	064
5	5	065
6	6	066
7	7	067
8	8	070
9	9	071
:	8-2	072
;	11-8-6	073
<	12-8-4	074
=	8-6	075
>	0-8-6	076
?	0-8-7	077

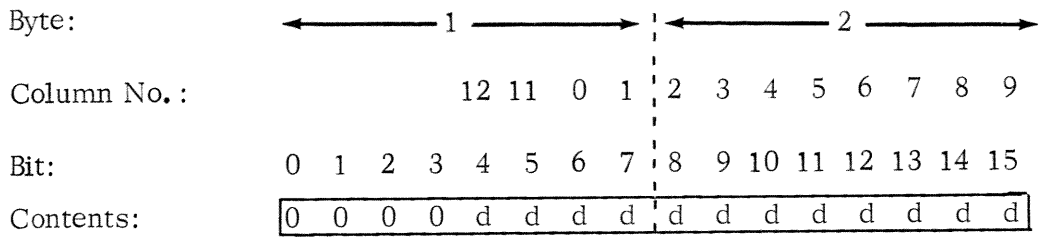
Hollerith-ASCII Translation Table

CHAR.	CARD CODE	ASCII CODE
@	8-4	100
A	12-1	101
B	12-2	102
C	12-3	103
D	12-4	104
E	12-5	105
F	12-6	106
G	12-7	107
H	12-8	110
I	12-9	111
J	11-1	112
K	11-2	113
L	11-3	114
M	11-4	115
N	11-5	116
O	11-6	117
P	11-7	120
Q	11-8	121
R	11-9	122
S	0-2	123
T	0-3	124
U	0-4	125
V	0-5	126
W	0-6	127
X	0-7	130
Y	0-8	131
Z	0-9	132
[	12-8-2	133
\	0-8-2	134
]	11-8-2	135
¬ or ↑	11-8-7	136
— or ←	0-8-5	137

CHAR.	CARD CODE	ASCII CODE
`	8-1	140
a	12-0-1	141
b	12-0-2	142
c	12-0-3	143
d	12-0-4	144
e	12-0-5	145
f	12-0-6	146
g	12-0-7	147
h	12-0-8	150
i	12-0-9	151
j	12-11-1	152
k	12-11-2	153
l	12-11-3	154
m	12-11-4	155
n	12-11-5	156
o	12-11-6	157
p	12-11-7	160
q	12-11-8	161
r	12-11-9	162
s	11-0-2	163
t	11-0-3	164
u	11-0-4	165
v	11-0-5	166
w	11-0-6	167
x	11-0-7	170
y	11-0-8	171
z	11-0-9	172
{	12-0	173
¦	12-11	174
}	11-0	175
~	11-0-1	176
DEL	12-9-7	177

Hollerith - ASCII Translation Table (Continued)

The packing is done as follows:



The "d's" will be 1 for every column punched. End-of-card in a read sequential is signified by a byte pair containing the word 100000. The specified byte count must be an even number.

Write a Line (.WRL)

This command presumes an ASCII file. AC0 must contain a byte pointer to the starting byte address within the user area from which characters will be written. Writing will terminate normally upon writing a null, a carriage return, or a form feed, and abnormally after transmission of 132 (decimal) characters without detection of a carriage return, a null, or a form feed as the 133rd character. In all cases, AC1 will contain, upon termination, the number of bytes written from the user area to complete the request. The termination of a write line upon a null permits formatting output without forcing a carriage return.

The format of the .WRL command is:

```
.SYSTM
.WRL n                    :WRITE TO CHANNEL n
error return
normal return
```

Write a Line (.WRL) (Continued)

Possible errors resulting from the .WRL command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
10	ERWPR	Attempt to write a write-protected file.
15	ERFOP	Attempt to reference a file not opened.
22	ERLLI	Line limit (132 characters) exceeded.

Write Sequential (.WRS)

This command writes data exactly as it is in the user area. AC0 must contain a byte pointer to the starting address of the data within the user area, and AC1 must contain the number of bytes to be written. The format of the .WRS command is:

```
.SYSTM
.WRS  n          ;WRITE TO CHANNEL n
error return
normal return
```

Possible errors resulting from a .WRS command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
10	ERWPR	Attempt to write a write-protected file.
15	ERFOP	Attempt to reference a file not opened.

CONSOLE COMMANDS

Buffered transfer of single characters between a program console and AC0 is handled by the commands. GCHAR and . PCHAR. These commands reference $TTI/$TTO. No channel number is required for these commands, and the console is always available to them without requiring an open command.

Get a Character (. GCHAR)

This command returns a character typed from the console in AC0. The character is right-adjusted in AC0 with bits 0-8 cleared. No channel is required: the $TTI is always used as input for this command. The format of the . GCHAR command is:

```
. SYSTM
. GCHAR
error return
normal return
```

No error return is possible from this command: if no character is currently in the TTI input buffer, the caller waits.

Put a Character (. PCHAR)

This command transmits a character in AC0, bits 9-15, to the console. No channel is required: the $TTO is always used as output for this command. The format of the . PCHAR command is :

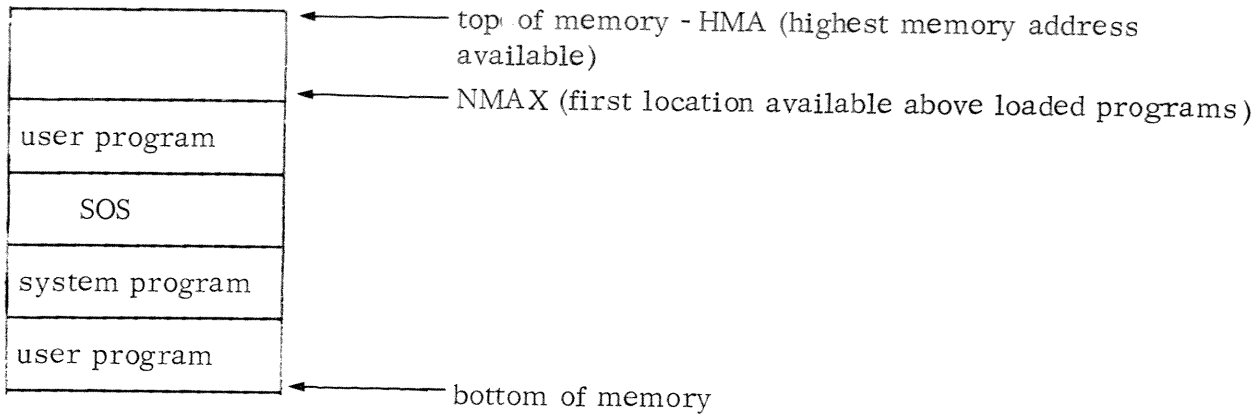
```
. SYSTM
. PCHAR
error return
normal return
```

No error return is possible from this command.

MEMORY COMMANDS

Upon completion of a relocatable load, the Stand-alone Operating System resides in lower memory, among various user or system programs comprising the load module.

Memory then looks essentially like this:



The highest memory address available (HMA) is usually the first word below the Binary or Core Image Loader.

If a user symbol table has been loaded at the high end of user memory, HMA will be the first word below the user symbol table. (The loader, by default, moves the symbol table down so that the bottom of the table coincides with the first location not loaded into by the program.)

Determine Available Memory (.MEM)

This command returns the current value of NMAX in AC1 and the value of HMA in AC0. HMA represents the location immediately below either the bottom of the Binary or Core Image Loader or the end of the user symbol table. A SUB 1, 0 instruction determines the amount of additional memory available to the user program. The format of the .MEM command is:

```
.SYSTM
.MEM
error return
normal return
```

There are no error returns from this command.

Change NMAX (.MEMI)

This command allows the user to increase or decrease the value of NMAX. The increment or decrement (in two's complement) is passed in AC0. The command causes the value of NMAX to be updated in the UST (in USTNM) and the new NMAX to be returned in AC1. The format of this call is:

```
.SYSTM
.MEMI
error return
normal return
```

NMAX will not be changed if the new value of NMAX would be equal to or higher than HMA. No check is made as to whether or not the user decreases NMAX below its original value (as determined at relocatable load time) nor, if his symbol table resides in upper memory, whether he increases NMAX above the bottom of his symbol table.

Whenever a user program requires memory space above the loaded program, .MEMI should be invoked first to allocate the number of words needed. The allocated memory space may be used by programs for buffers, user stacks, temporary storage, etc.

There is one error resulting from a .MEMI command:

AC2	Mnemonic	Meaning
26	ERMEM	Attempt to allocate more memory than available.

CLOCK/CALENDAR COMMANDS

Four commands are provided to permit the system to keep track of the time of day and the current date. Dates are referenced as month/day/year. The time of day is given using a 24-hour clock. In addition, the .GHRZ command provides the frequency of the Real Time Clock, and the .DELAY suspends the execution of a program.

NOTE: All Clock/Calendar commands except .GHRZ are treated as illegal commands if a Real Time Clock Driver is absent from the load module.

Get the Time of Day (.GTOD)

This command requests the system to pass the current time in hours, minutes, and seconds to the caller. (To be meaningful, this time must have been initialized by the ".STOD" command; see below.) The time will be returned as follows:

AC0 - Seconds: AC1 - Minutes: AC2 - Hours (using a 24-hour clock)

The format of this command is:

```
.SYSTM
.GTOD
error return
normal return
```

No error return is possible.

Set the Time of Day (.STOD)

This command permits the setting of the system clock to a specific hour, minute, and second. The user passes the initial values as follows:

AC0 - Seconds: AC1 - Minutes: AC2 - Hours (using a 24-hour clock)

The format of this command is:

```
.SYSTM
.STOD
error return
normal return
```

A possible error message is:

AC2	Mnemonic	Meaning
41	ERTIM	Illegal time of day.

Get Today's Date (.GDAY)

This command requests the system to return the number of the current month, day, and year. (To be meaningful, this date must have been initialized by the ".SDAY" command; see below.) The month is returned in AC1, the day in AC0, and the current year (less 1968) is returned in AC2. The format of the .GDAY command is:

Get Today's Date (.GDAY) (Continued)

```
.SYSTM
.GDAY
error return
normal return
```

No error return is possible.

Set Today's Date (.SDAY)

This command permits the setting of the system calendar to a specific date. The user passes the number of the month in AC1 (January is month number 1), the number of the day within the month in AC0 and the number of the current year--less 1968--in AC2. This is the date that is unconditionally returned to the .GDAY command. It is not incremented when the time of day clock overflows. The format of the .SDAY command is:

```
.SYSTM
.SDAY
error return
normal return
```

One possible error message is:

AC2	Mnemonic	Meaning
41	ERTIM	Illegal day, month, or year.

Delay the Execution of a Task (.DELAY)

The calling program is suspended for the number of real-time clock pulses indicated by AC1. The real-time clock frequency was specified at load time (see .GHRZ).

The format of this command is:

AC1 - Number of RTC pulses

```
.SYSTM
.DELAY
error return
normal return
```

The error return is never taken. A "JMP ." instruction within the system is executed until this period elapses.

Examine the System Real Time Clock (.GHRZ)

This system call permits the user to examine the Real Time Clock frequency. The frequency is returned in AC0, in the following manner:

AC0	Meaning
0	There is no Real Time Clock in the system.
1	Frequency is 10 HZ.
2	Frequency is 100 HZ.
3	Frequency is 1000 HZ.
4	Frequency is 60 HZ (line frequency)
5	Frequency is 50 HZ (line frequency)

The format of this call is:

```
.SYSTM
.GHRZ
error return
normal return
```

The error return is never taken.

SERVICING USER INTERRUPTS

There are several considerations which must be made by any user wishing to service device interrupt requests. See Appendix A - Adding User Supplied Device Handlers.

Identify a User Interrupt (.IDEF)

In order to introduce to the system those devices (not identified at LOAD time) whose interrupts the system is to recognize, the system call. IDEF must be issued. This adds an entry to the SOS Internal Search List (See Appendix A). AC0 contains the device code of the new device. AC1 contains the address of the new device's DCT. If the device code that is passed is 77_8, then AC1 contains the address to which the system passes control whenever it detects a power-fail interrupt. The format of the command is:

```
.SYSTM
.IDEF
error return
normal return
```

Identify a User Interrupt (Continued)

Possible error messages are:

AC2	Mnemonic	Meaning
36	ERDNM	Illegal device code(>77_8). Device code 77_8 is reserved for power monitor/auto restart option.
45	ERIBS	Interrupt device code in use.

Remove User Interrupt Servicing Program (.IRMV)

To prevent the system's recognition of user interrupts which have been previously identified by the .IDEF command, the .IRMV command must be issued. AC0 contains the user device code which is to be removed from the system's recognition. The format of the .IRMV command is:

```
.SYSTM
.IRMV
error return
normal return
```

One possible error message may be given:

AC2	Mnemonic	Meaning
36	ERDNM	Illegal device code (>77_8).

ERROR MESSAGES

Code	Mnemonic	Meaning	Applicable Commands
0	ERFNO	Illegal channel number.	.OPEN .EOPEN .ROPEN .GTATR .RDL .RDS .WRL .WRS
1	ERFNM	Illegal file Name.	.OPEN .EOPEN .ROPEN
2	ERICM	Illegal system command.	- - -
3	ERICD	Illegal command for device.	.RDL .RDS .WRL .WRS
6	EREOF	End of file.	.RDL .RDS
7	ERRPR	Attempt to read a read-protected file.	.RDL .RDS
10	ERWPR	Attempt to write a write-protected file.	.WRL .WRS
12	ERDLE	Attempt to reference a non-existent file	.OPEN .EOPEN .ROPEN
15	ERFOP	Attempt to reference a file not opened	.GTATR .CLOSE .RDL .RDS .WRL .WRS

ERROR MESSAGES (Continued)

Code	Mnemonic	Meaning	Applicable Commands
21	ERUFT	Attempt to use channel already in use.	.OPEN .EOPEN .ROPEN .GCHN
22	ERLLI	Line limit exceeded on read or write line.	.RDL .WRL
24	ERPAR	Parity error on read line.	.RDL
26	ERMEM	Attempt to allocate more memory than available.	.MEMI
30	ERFIL	File read error.	.RDL .RDS
31	ERSEL	Unit not properly selected.	.OPEN .EOPEN .ROPEN
36	ERDNM	Illegal device code ($>77_8$)	.IRMV .IDEF
41	ERTIM	Illegal time or date.	.SDAY .STOD
45	ERIBS	Interrupt device code in use.	.IDEF

DEVICE RESPONSE TO SOS COMMANDS

This section describes the function performed by the SOS I/O commands, as applied to each of the devices supported by DGC.

TTI1

.OPEN	Device is initialized.
.CLOSE	Device is reinitialized.
.RDS	The specified bytes are read into the user area from the device unedited.
.RDL	The ASCII string is read into the user area from the device. The input stream is echoed on the TTO1. A rubout character deletes the previous input character and causes a back arrow to be echoed. The shift L character causes the entire input string to be deleted. CTRL Z causes an END OF FILE return. Line feeds are ignored.

$PLT

.OPEN	Device is initialized.
.CLOSE	Device is reinitialized after outstanding I/O is complete.
.WRS	The specified bytes are output to the device, unedited.
.WRL	Illegal command to this device.

TTO1

.OPEN	Device is initialized.
.CLOSE	Device is reinitialized after outstanding I/O is complete.
.WRS	The specified bytes are output to the device unedited.
.WRL	The ASCII string is output to the device with simulated tabbing, a line feed inserted after carriage return, and nulls inserted after form feeds.

$TTP

.OPEN	Device is initialized: leader is punched.
.CLOSE	Device is reinitialized after outstanding I/O is complete; trailer is punched.
.WRS	The specified bytes are output to the device, unedited.
.WRL	The ASCII string is output to the device with rubout characters inserted after tabs, a line feed inserted after carriage returns, and nulls inserted after form feeds.

DEVICE RESPONSE TO SOS COMMANDS (Continued)

$CDR

.OPEN	Device is initialized; a prompt message is written and a response is necessary for the program to continue.
.CLOSE	Device is reinitialized.
.RDS	The specified bytes are read into the user area from the device, unedited. Each pair of bytes read represents one full column of the card. Bits 4-7 of the first byte represent card rows 12, 11, 0, and 1 respectively. Bits 0-7 of the second byte represent card rows 2-9 respectively. A one in bit 0 of the first byte of a pair indicates end of card. No other meaningful data is included in this byte pair.
.RDL	The 80-character ASCII string is read into the user area from the device. If the characteristic DCC80 was suppressed on the .OPEN, then 72 columns are read. The translation from Hollerith is performed in the card reader driver. A 12-11-0-1-2-3-4-5-6-7-8-9 punch causes end of file. The byte count returned to the user reflects the last non-blank character on the card. See page 4-12.

$TTO

.OPEN	Device is initialized.
.CLOSE	Device is reinitialized after outstanding I/O is complete.
.WRS	The specified bytes are output to the device unedited.
.WRL	The ASCII string is output to the device with simulated tabbing , a line feed inserted after carriage return, and nulls inserted after form feeds.

$TTI

.OPEN	Device is initialized.
.CLOSE	Device is reinitialized.
.RDS	The specified bytes are read into the user area from the device, unedited.
.RDL	The ASCII string is read into the user area from the device. The input stream is echoed on the $TTO. A rubout character deletes the previous input character and causes a back arrow to be echoed. The shift L character causes the entire input string to be deleted. CTRL Z causes an END OF FILE return. Line feeds are ignored.

DEVICE RESPONSE TO SOS COMMANDS (Continued)

$TTR

.OPEN	Device is initialized: a prompt message is written and a response is necessary for the program to continue.
.CLOSE	Device is reinitialized.
.RDS	The specified bytes are read into the user area from the device, unedited.
.RDL	The ASCII string is read into the user area from the device. Rubouts and line feeds are ignored.

CTA (See also Chapter 2, SOS Cassette and Magnetic Tape Files.)

.OPEN	The specified file is located on the tape reel, and the read/write head positioned to the file mark preceding.
.CLOSE	Device is reinitialized. Following write operations, the last partial block is output to the file(padded with nulls if necessary) and two end of file marks are written.
.RDS	The specified bytes are read into the user area from the file, unedited.
.RDL	The ASCII string is read into the user area from the file. Rubouts and line feeds are ignored.
.WRS	The specified bytes are output to the file, unedited.
.WRL	The ASCII string is output to the file.

$LPT

.OPEN	Device is initialized.
.CLOSE	Device is reinitialized; a form feed character is output.
.WRS	The specified bytes are output to the device, unedited.
.WRL	The ASCII string is output to the device with simulated tabbing, and line feeds are inserted after carriage returns.

NOTE: If the external symbol .LPTD triggered the loading of the LINE PRINTER Driver, then all characters after the 80th are ignored on .WRL commands. If the symbol .L132 was the trigger, then all 132 characters will be printed.

DEVICE RESPONSE TO SOS COMMANDS (Continued)

$PTR

.OPEN	Device is initialized; a prompt message is written and a response is necessary for the program to continue.
.CLOSE	Device is reinitialized.
.RDS	The specified bytes are read into the user area from the device, unedited.
.RDL	The ASCII string is read into the user area from the device. Rubouts and line feeds are ignored.

$PTP

.OPEN	Device is initialized; leader is punched.
.CLOSE	Device is reinitialized after outstanding I/O is complete; trailer is punched.
.WRS	The specified bytes are output to the device, unedited.
.WRL	The ASCII string is output to the device with rubouts inserted after tabs, a line feed after a carriage return, and nulls after a form feed.

MTA (See Chapter 2, SOS Cassette and Magnetic Tape Files.)

.OPEN	The specified file is located on the tape reel, and the read/write head positioned to the file mark preceding.
.CLOSE	Device is reinitialized. Following write operations, the last partial block is output to the file(padded with nulls if necessary) and two end of files marks are written.
.RDS	The specified bytes are read into the user area from the file, unedited.
.RDL	The ASCII string is read into the user area from the file. Rubouts and line feeds are ignored.
.WRS	The specified bytes are output to the file, unedited.
.WRL	The ASCII string is output to the file.

USER STATUS TABLE

The User Status Table (UST) is a 22 octal word table which records all information pertinent to the execution of an entire program level. This table is located at addresses 0400 through 0421* inclusive and has the following structure:

address	label	contents
400	USTPC	Program counter
401	USTZM	ZMAX
402	USTSS	Start of Symbol Table (SST)
403	USTES	End of Symbol Table (EST)
404	USTNM	NMAX after runtime .MEMIs
405	USTSA	Starting address of program
406	USTDA	Debugger address; -1 if not loaded
407	USTHU	Highest load address
410	USTCS	FORTRAN common area size
411	USTIT	Interrupt address (Control A keyboard character)
412	USTBR	Break address (Control C keyboard character)
413	USTCH	Number of channels and TCBs (unused by SOS)
414	USTCT	Current TCB pointer (unused by SOS)
415	USTAC	Start of active TCB chain (unused by SOS)
416	USTFC	Start of free TCB chain (unused by SOS)
417	USTIN	Initial Start of NREL code (INMAX)
420	USTOD	Overlay directory address (unused by SOS)
421	USTSV	FORTRAN State Variable Save routine pointer (unused by SOS)

Location 400 - USTPC is the program counter.

Location 401 - USTZM points to the first available location in page zero for page zero relocatable code.

Location 402 and 403 - USTSS and USTES point to the start and end of the symbol table respectively. The loader sets 402 and 403 to 0 if the debugger is not loaded.

Location 404 - USTNM contains NMAX. The loader sets the pointer to the first free location for further loading or for allocation of temporary storage at run time.

*Location 12, USTP always points to the start of the UST.

USER STATUS TABLE (Continued)

Location 405 - USTSA points to the program starting address, specified by the .END statement. If no starting address is specified by any loaded program, -1 is stored in 405. If several programs specify starting addresses, USTSA contains the address specified in the last program loaded. (Location 377 contains a JMP @2, which transfers control to a routine in SOS which performs a .SYSI and then branches to the program starting address. Therefore, the user can conveniently restart his program at 377, assuming that he has specified a starting address.)

Location 406 - USTDA points to the starting address of the debugger, or if the debugger is not loaded, 406 contains -1.

Location 407 - USTHU is set to the value of NMAX at the termination of loading. This word is never changed by the operating system during program execution. It is used to reset USTNM whenever a .SYSI is executed.

Location 410 - USTCS contains the size of the FORTRAN unlabeled common area.

Location 411 and 412 - USTIT and USTBR are the interrupt address and break address respectively. Both are initialized to -1. Whenever the location contains 0 or -1, the corresponding interrupt is ignored by the system. To receive control after CTRL A interrupts, USTIT must be changed by the user program to the desired transfer address. The Core Image Loader sets USTBR to the beginning address of the Core Image Loader whenever it loads a save file into core. The user program may then modify USTBR to either point to its own CTRL C break address or to disable CTRL C interrupts. When control is passed to the USTIT address, machine interrupts are enabled; when control is passed to the USTBR address, machine interrupts are disabled.

Location 413-416 - Locations that are used by RDOS and RTOS.

Location 417 - USTIN contains the address of the start of normally relocatable code (440_8).

Location 420 - Location used by RDOS.

Location 421 - USTSV points to the address of the FORTRAN state variable save routine.

CHAPTER 5

CONFIGURING SOS UTILITY PROGRAMS

The process of configuring a utility program generally consists of the following:

1. Producing a trigger which specifies the desired I/O support.
2. Performing a relocatable load of the trigger, the appropriate SOS Libraries, and the relocatable binary (RB) version of the utility program.

These procedures apply to all SOS utilities except the assembler. The assembler program requires an execution pass on the DGC command definitions in order to expand its permanent symbol table appropriately after it has been loaded.

SUPPLIED TAPES

The paper tapes listed below comprise the entire SOS package. These tape lists are separated according to those that are supplied to all users and those that are supplied to paper tape, cassette tape, and magnetic tape users respectively.

The absolute binary (AB) versions of the programs listed below are preconfigured with conventional paper tape support that is, high speed paper tape reader and punch, full teletype (keyboard, printer, reader, and punch) and 80 column line printer. In addition to these devices, the absolute binary programs supplied to cassette users include three cassette units (0, 1, and 2); absolute binaries supplied to magnetic tape users include three magnetic tape units (0, 1, and 2).

The paper tapes that are supplied to all SOS users are the following:

Name	Tape Number
SOS Library	099-000010
SOS Text Editor (RB) - utility	089-000104
SOS Extended Assembler (RB) - utility	089-000106
SOS Library File Editor (RB) - utility	089-000081
SOS SYSGEN (RB) - utility	089-000122
RDOS User Parameters (PARU)	090-000883
SOS System Parameters (PARA)	090-000498
SOS User Application Parameters (PARUA)	090-000889
SOS Debug	089-000167

SUPPLIED TAPES (Continued)

Name	Tape Number
Extended Assembler Command Definitions	
DGC Basic Instructions	090-001482
Floating Point Interpreter	090-001483
Operating System Instructions	090-001484

The additional paper tapes that are supplied to users without cassette or magnetic support are the following:

Name	Tape Number
SOS Extended Assembler (AB) - utility	091-000069
SOS Library File Editor (AB) - utility	091-000057
SOS SYSGEN (AB) - utility	091-000070
Stand-alone Extended Relocatable Loader (AB)	091-000038
Relocatable Binary Punch Program (RB)	089-000080

The additional paper tapes that are supplied to users with cassette drives are the following:

Name	Tape Number
SOS Cassette Library	099-000041
Cassette Core Image Loader/Writer (AB) - utility	091-000067
SOS Relocatable Loader (RB) - utility	089-000120
SOS Command Line Interpreter (RB) - utility	089-000121
SOS SYSGEN with cassette support (AB) - utility	091-000071
SOS CLI with cassette support (AB) - utility	091-000072
SOS Relocatable Loader with cassette support (AB) - utility	091-000073

The additional paper tapes that are supplied to users with magnetic tape drives are the following:

Name	Tape Number
SOS Magnetic Tape Library	099-000042
Magnetic Tape Core Image Loader/Writer (AB) - utility	091-000068
SOS Relocatable Loader (RB) - utility	089-000120
SOS Command Line Interpreter (RB) - utility	089-000121
SOS SYSGEN with mag tape support (AB) - utility	091-000074
SOS CLI with mag tape support (AB) - utility	091-000075
SOS Relocatable Loader with mag tape support (AB) - utility	091-000076

SUPPLIED TAPES (Continued)

Available to cassette users at extra cost are cassette reels 070-000002 and 070-000008. Available to magnetic tape users at extra cost is magnetic tape reel 071-000004. The programs in save (SV) file format on these tapes are preconfigured in the same manner as the corresponding absolute binaries listed above. In the subsequent procedures to produce a tailored master reel, the library and relocatable binary files on these tapes may be substituted for the corresponding paper tapes as input to the SOS Relocatable Loader.

The contents of these reels are as follows:

CASSETTES (Model #3235)

070-000002:

Core Image Loader/Writer (SV)	File 0
Relocatable Loader (SV)	File 1
Command Line Interpreter (SV)	File 2
Text Editor (SV)	File 3
Assembler (SV)	File 4
Library File Editor (SV)	File 5
SYSGEN (SV)	File 6

071-000003:

Cassette Library	File 0
SOS Library	File 1
Command Line Interpreter (RB)	File 2
Text Editor (RB)	File 3
Assembler (RB)	File 4
Relocatable Loader (RB)	File 5
Library File Editor (RB)	File 6
SYSGEN (RB)	File 7
Extended Assembler Command Definitions:	
Nova Basic Instructions	File 8
Floating Point Interpreter	File 9
Operating Systems	File 10
RDOS User Parameters	File 11
SOS Stand-alone Parameters	File 12
SOS User Application Parameters	File 13

SUPPLIED TAPES (Continued)

MAGNETIC TAPE (Model #3236)

071-000004:

Core Image Loader / Writer (SV)	File 0
Relocatable Loader (SV)	File 1
Command Line Interpreter (SV)	File 2
Text Editor (SV)	File 3
Assembler (SV)	File 4
Library File Editor (SV)	File 5
SYSGEN (SV)	File 6
SOS Magnetic Tape Library	File 7
SOS Library	File 8
Command Line Interpreter (RB)	File 9
Text Editor (RB)	File 10
Assembler (RB)	File 11
Relocatable Loader (RB)	File 12
Library File Editor (RB)	File 13
SYSGEN (RB)	File 14
Extended Assembler Command Defininitions:	
Nova Basic Instructions	File 15
Floating Point Interpreter	File 16
Operating Systems	File 17
RDOS User Parameters	File 18
SOS Stand-alone Parameters	File 19
SOS User Application Parameters	File 20

The relocatable binary versions of the SOS Utility programs can be used to produce executable versions of these programs configured with arbitrary I/O support. This facility permits users to eliminate all undesired device drivers and to reclaim the core space taken up by them.

The source files (090 series tapes) contained in the package are used to:

1. Add permanent symbols to user generated SOS Assemblers.

2. Define equivalences in certain user programs; the various parameter tapes provide these facilities.

The remainder of this chapter describes in detail the procedures for generating SOS Utility Programs that are tailored to the user's I/O configuration.

PRODUCING A TRIGGER

Triggers are produced by the SYSG program. This program accepts a command line, which contains device driver ENTRY symbols, from the console device. It outputs a relocatable binary file (the trigger) which is comprised of EXTERNAL NORMAL symbols corresponding to the named device drivers. These EXTERNAL NORMALs cause the selection or "triggering" of the desired routines for loading when the trigger precedes the SOS Libraries as input to the relocatable loader.

The first step to produce a trigger is to load and start the SYSG program. This can be done by using the binary loader to load an absolute binary SYSG paper tape (091-000070, 091-000071, or 091-000074). SYSG can be loaded from the cassette or magnetic tape using the Core Image Loader/Writer.

When the SYSGEN program is started, it outputs the prompt message:

```
SYSG
```

and waits for the user to type a command line. This command line has the following format:

(SYSG) driver$_1$ ... driver { triggername/T }^{n} .RDSI { .CTB } { .RTC1 } output-file /O ↑)

where:

driver: is an entry symbol in the desired device driver routine. Table 2-2 lists all possible symbols

.CTB is the optional Command Table Builder. This symbol must be specified for triggers to be used in configuring the SOS Assembler, the SOS Relocatable Loader, the SYSGEN program, and the SOS FORTRAN IV Compiler.

.RTC1 causes the loading of a 10 Hz Real Time Clock Driver from the SOS Library. This symbol must be specified for the CLI.

output-file is the name of the file/device to which the user wishes the trigger to be output. This name must be followed by the /O switch.

trigger-name is the optional title (.TITL) of the trigger. If this name is omitted the trigger will be named SGTRG, by default. If the name is present, it must be followed by the /T switch.

PRODUCING A TRIGGER (Continued)

An example of the use of the SYSG program is shown below. This command line causes a trigger to be generated at the high speed paper tape punch to provide support for a small teletype, high speed reader and punch, Command Table Builder, and cassette units 0 and 1:

```
(SYSG) .PTRD .PTPD .STTY $PTP/O .RDSI .CTB .CTU1 )
```

The trigger produced will have the title SGTRG since no trigger title was specified.

After the trigger has been output, the SYSGEN program will again type its prompt (SYSG) and wait for another command. If more than one utility program is to be configured, requiring different triggers, it is most convenient to generate all the necessary triggers before overwriting the SYSGEN program in core.

A discussion of the SYSGEN program, including its associated error messages, is included in Chapter 3.

PROCEDURES FOR PAPER TAPE UTILITIES

Configuring Utilities except the Assembler

The following is a step by step description for configuring all SOS Paper Tape Utilities except the assembler. The end result is an absolute binary paper tape of the utility. Before starting this procedure, the trigger to be used should be generated on paper tape. Each typed command in this procedure must be terminated by a carriage return.

1. Using the binary loader, load the Extended Relocatable Loader (tape 091-000038).

2. Mount the trigger in the teletype reader and type 1, or in the high-speed paper tape reader and type 2.

3. Mount the SOS Library (099-000010) in the teletype reader and type 1, or in the high-speed paper tape reader and type 2. If the trigger specifies support for cassette or magnetic tape drives, perform this process first for the SOS Cassette Library (099-000041) or the SOS Magnetic Tape Library (099-000042), and then the SOS Library.

4. Mount the relocatable binary version of the program to be configured in the teletype reader and type 1, or in the high-speed paper tape reader and type 2.

5. Type 5 and note the value of NMAX output by the relocatable loader on the teletype; this number will be used in Step 12.

PROCEDURES FOR PAPER TAPE UTILITIES (Continued)

Configuring Utilities except the Assembler (Continued)

6. Mount the relocatable binary punch program (089-000080) on the teletype reader and type 1, or on the high-speed paper tape reader and type 2.

7. Type 6 and note the value of RBFP output by the relocatable loader on the teletype; this number will be used in Step 9.

8. Type 8 to terminate the loading process.

9. Enter RBFP (from Step 7) into the data switches on the computer console, press RESET and then press START.

10. Type 0H for output on the teletype punch or 1H for output on the high-speed paper tape punch.

11. Type 1, nmaxP where nmax is the value of NMAX noted in Step 5.

12. Type 377E, to specify a starting address for the program.

Configuring the Assembler

The following is a step by step description for configuring the SOS Assembler. It is similar to the preceding procedures; however before an absolute binary tape is punched, the DGC command definitions are added to the assembler permanent symbol table.

1. Using the binary loader, load the Extended Relocatable Loader (tape 091-000038).

2. Mount the trigger in the teletype reader and type 1, or in the high-speed paper tape reader and type 2.

3. Mount the SOS Library(099-000010) in the teletype reader and type 1, or in the high-speed paper tape reader and type 2. If the trigger specifies support for cassette or magnetic tape drives, perform this process first for the SOS Cassette Library (099-000041) or the SOS Magnetic Tape Library (099-000042), respectively, and then mount the SOS Library.

4. Mount the relocatable binary version of the SOS Extended Assembler (089-000106) in the teletype reader and type 1, or in the high-speed paper tape reader and type 2.

PROCEDURES FOR PAPER TAPE UTILITIES (Continued)

Configuring the Assembler (Continued)

5. Enter 016500_8 in the data switches on the computer console and type 3.

6. Mount the relocatable binary punch program (089-000080) on the teletype reader and type 1, or on the high-speed paper tape reader and type 2.

7. Type 6 and note the value of RBFP output by the relocatable loader on the teletype; this number will be used in Step 13.

8. Type 8 to terminate the loading process.

9. Press CONTINUE on the computer console to start the assembler.

10. Mount the extended assembler command definitions tape (090-001482) in the teletype reader and type 0 $TTR/3, or in the high-speed paper tape reader and type 0 $PTR/3.

11. Mount the remaining tapes (090-001483 and 090-001484) when they are requested by the prompt message and strike any console key as requested.

12. When the assembler halts, examine AC0 and note its contents (NMAX); this value will be used in step 15.

13. Enter RBFP (from step 7) into the data switches on the computer console, press RESET, and then press START.

14. Type 0H for output on the teletype punch or 1H for output on the high-speed paper tape punch.

15. Type 1, nmaxP where nmax is the value of NMAX noted in step 11.

16. Type 377E to specify a starting address for the program.

PRODUCING A MASTER REEL

The following procedure details the necessary steps to configure SOS utility programs and at the same time produce a master cassette or magnetic tape reel. The assumption is made that only one cassette or magnetic tape drive is available and that all programs input to the Relocatable Loader are on paper tape. The high speed paper tape reader is assumed to be the input device for these tapes.

PRODUCING A MASTER REEL (Continued)

Before starting this procedure, the triggers should have been generated on paper tapes. When the cassette or magnetic tape reel that is required for this procedure has been mounted, the cassette should have its file protect tab in place and the magnetic tape should contain a write permit ring, so that files may be written on them. Each typed command in this procedure must be terminated by a carriage return. The master reel generated on unit 0 need never be removed from that unit during the following procedures.

1. CILW:

   Using the binary loader, load and start the absolute binary version of the Core Image Loader/Writer (091-000067 for cassette; 091-000068 for magnetic tape). When started this program will output the following message to the console device:

   LOAD UNIT 0: STRIKE ANY KEY

   Place the cassette or magnetic tape reel that is to become the master on unit 0 and depress any key on the console keyboard. This program will write a Core Image Loader/Writer (CILW) to file 0 of unit 0. When the loader has been successfully written the message,

   LOADER INSTALLED

   is printed at the console and the tape is rewound. (This program may be used to install additional loaders at this point. After the program HALTs, the user may depress the CONTINUE switch on the master console. The above message is again printed and the user may then change cassette or mag tape reels or unit assignments and continue the above procedures).

2. RLDR:

   The binary loader remains in high core. Using this loader, load the absolute binary version of the SOS Relocatable Loader (091-000073 for cassette; 091-000076 for magnetic tape). This program outputs the prompt message (RLDR) to the console device; respond with the command line:

   CT0:1/S $PTR/4 (for cassette)
   MT0:1/S $PTR/4 (for magnetic tape)

2. RLDR: (Continued)

If the teletype reader is used, substitute $TTR for $PTR. As the relocatable loader requests them, mount the following paper tapes in order:

1. The trigger to be used for the Relocatable Loader.
2. The SOS Cassette Library (099-000041) for cassette or the SOS Magnetic Tape Library (099-000042) for magnetic tape.
3. The SOS Library (099-000010)
4. The relocatable binary version of the SOS Relocatable Loader (089-000120)

The loader will produce a core image of the tailored Relocatable Loader on file 1 of the master reel, type OK, and HALT when finished. This version of the loader is now usable for the remaining procedures.

It will typically need to be re-loaded via the CILW, but in this one case it is already present in core. Hence to restart the loader, merely depress the CONTINUE switch on the master console.

3. CLI:

After the Relocatable Loader outputs the prompt message (RLDR) to the console, respond with the command line:

```
CT0:2/S  $PTR/4  (for cassette)
MT0:2/S  $PTR/4  (for magnetic tape)
```

If the teletype reader is used, substitute $TTR for $PTR. As the relocatable loader requests them, mount the following paper tapes in order:

1. The trigger to be used for the Command Line Interpreter.
2. The SOS Cassette Library (099-000041) for cassette or the SOS Magnetic Tape Library (099-000042) for magnetic tape.
3. The SOS Library (099-000010)
4. The relocatable binary version of the SOS Command Line Interpreter (089-000121).

The loader will produce a core image or save file version of this utility program on the designated file of the master reel. It will type OK and HALT when finished.

PRODUCING A MASTER REEL (Continued)

4. Restore RLDR:

The Relocatable Loader must be restored to core using the CILW. Restart the CILW by setting the address of the last location in core into the data switches on the master console. Depress RESET, and then START. (An alternate procedure to start this program is to manually rewind unit 0, make the unit ready, depress RESET on the master console, and then depress PROGRAM LOAD switch. Data switches 0, 11, and 14 must be set to ones before depressing PROGRAM LOAD. This procedure must be used after step 3 because the CILW has not yet overwritten the absolute binary loader in high core).

The CILW is invoked and types its prompt message:

#

to the console device. Respond to this prompt with the command line:

0:1

The user-configured version of the SOS Relocatable Loader will be returned to core.

5. EDIT:

Repeat step 3 making the following substitutions:

1. CT0:3/S or MT0:3/S instead of CT0:2 or MT0:2 respectively in the Relocatable Loader command line.

2. The trigger for the SOS Text Editor instead of the trigger for the CLI.

3. The relocatable binary version of the SOS Text Editor (089-000104) instead of that of the CLI.

6. Restore RLDR:

Repeat step 4 to restart the Relocatable Loader.

7. ASM:

Repeat step 3 making the following substitutions:

1. CT0:4/S or MT0:4/S instead of CT0: 2/S or MT0: 2/S respectively in the Relocatable Loader command line.

2. The trigger for the SOS Assembler instead of the trigger for the CLI.

3. The relocatable binary version of the SOS Assembler (089-000106) instead of that of the CLI.

8. ASM:

Depress the CONTINUE switch on the master console to start the assembler.

9. ASM:

Respond to the assembler prompt message (ASM) with the following command line:

0 $PTR/3 (or 0 $TTR if the teletype reader is used)

If the Floating Point Instruction definitions are not desired, respond with:

0 $PTR/2 or 0 $TTR/2 .

10. ASM:

Mount each command definition tape as it is requested (by the LOAD $TTR / $PTR , STRIKE ANY KEY message) and depress any console key. These tapes are (in order):

Nova Basic Instructions	(090-001482)
*Floating Point Interpreter	(090-001483)
Operating System Instructions	(090-001484)

*only use if these command definitions are desired for permanent symbols.

ASM: (Continued)

The assembler will perform a one pass assembly on these source files and then HALT.

11. ASM:

EXAMINE (using the master console switches) the contents of AC0 and note for use in step 12.

12. CILW:

Start the Core Image Writer by setting the address of the next to last location in core into the data switches on the master console, pressing RESET, and then START. Respond to this program's prompt message (#) with:

0:4

Respond to the next prompt message (NMAX:) with the octal value noted in step 11. The Core Image Writer will rewrite a core image of the assembler to file 4 of the master reel. This copy contains the assemblers permanent symbols in the symbol table. It will type OK and HALT when finished.

13. Restore RLDR:

Repeat step 4 to restart the Relocatable Loader.

14. LFE:

Repeat step 3 making the following substitutions:

1. CT0: 5/S or MT0: 5/S instead of CT0:2/S or MT0: 2/S respectively in the Relocatable Loader command line.

2. The trigger for the SOS Library File Editor (LFE) instead of the trigger for the CLI.

3. The relocatable binary version of the SOS LFE (089-000081) instead of that of the CLI.

PRODUCING A MASTER REEL (Continued)

15. Restore RLDR:

Repeat step 4 to restart the Relocatable Loader.

16. SYSG:

Repeat step 3 making the following substitutions:

1. CT0:5/S or MT0:6/S instead of CT0:2/S or MT0:2/S respectively in the Relocatable Loader command line.

2. The trigger for the SOS SYSGEN program instead of the trigger for the CLI.

3. The relocatable binary version of SOS SYSGEN (089-000122) instead of that of the CLI.

This completes the generation of a master reel.

CONFIGURING A SOS FORTRAN IV COMPILER

The procedures for configuring a FORTRAN IV compiler are identical to the previous procedures. The additional tapes that are required are the relocatable binary version of the FORTRAN IV Compiler (089-000161) and the SOS FORTRAN Interface (089-000041). The input order of the tapes to the appropriate relocatable loader is the following:

1. FORTRAN Compiler trigger (see Producing a Trigger).

2. SOS Cassette Library, SOSCT. LB (099-000041) for cassette systems.

3. SOS Magnetic Tape Library, SOSMT. LB (099-000042) for magnetic tape systems.

4. SOS Library, SOS. LB (099-000010).

5. SOS FORTRAN Interface, SOSFI. RB (089-000041).

6. FORTRAN IV Compiler, FORT. RB (089-000161).

For magnetic tape and cassette systems, the core image (save) version of this program may be placed onto the master reel as file 7. If it occupies this position on a master reel, then it may be read into core via the CLI command:

```
BOOT FORT
```

See Chapter 3, BOOT command for a further description.

Example

Figure 5-1 shows the $TTO output as a result of the procedures to generate a master reel. The triggers were generated on paper tape. The trigger titles which are printed reflect the three versions of triggers necessary to configure all SOS utilities:

TRIG1 - used for the RLDR, ASM, SYSGEN, and FORTRAN IV Compiler; Contains .RDSI, .CTB, and all desired device driver symbols.

TRIG2 - used for the CLI; Contains .RDSI, .RTCI, and all desired device driver symbols.

TRIG3 - used for the Text Editor and LFE; Contains .RDSI and all desired device driver symbols.

The other device driver symbols used for this example are:

- .CDRD - card reader
- .STTY - small teletype
- .TTI1 - second teletype
- .L132 - 132 column line printer
- .PTRD - high speed paper tape reader
- .PTPD - high speed paper tape punch
- .MTU1 - two magnetic tape units

In this example, a master magnetic tape reel is created on unit 0.

```
LOAD UNIT Ø: STRIKE ANY KEY      ← Step 1

LOADER INSTALLED

RLDR MTØ:1/S $PTR/4              ← Step 2
LOAD $PTR, STRIKE ANY KEY        ← RLDR Trigger
 TRIG1
LOAD $PTR, STRIKE ANY KEY        ← SOSMT.LB
LOAD $PTR, STRIKE ANY KEY        ← SOS.LB
LOAD $PTR, STRIKE ANY KEY        ← RLDR.RB
 RLDR
XN .GDTE ØØ5617
XN .SDTE ØØ5615
XN .UØ4D ØØ5651
XN .UØ3D ØØ5650
XN .UØ2D ØØ5647
XN .UØ1D ØØ5646
XN .UØØD ØØ5645
XN .CTU7 ØØ5704
XN .CTU6 ØØ5703
XN .CTU5 ØØ5702
XN .CTU4 ØØ5701
XN .CTU3 ØØ5700
XN .CTU2 ØØ5677
XN .CTU1 ØØ5676
XN .CTUØ ØØ5675
XN .MTU7 ØØ5674
XN .MTU6 ØØ5673
XN .MTU5 ØØ5672
XN .MTU4 ØØ5671
XN .MTU3 ØØ5670
XN .MTU2 ØØ5667
XN .CTAD ØØ7504
XN .OPPP ØØ7502
XN .OPTP ØØ5644
XN .PLTD ØØ5653
XN .RTCD ØØ6750
XN .DLY  ØØ5613
XN .STDD ØØ5607
XN .GTDD ØØ5611
    NMAX ØØ14661
    ZMAX ØØØ3Ø2
    CSZE
     EST
     SST

OK
```

List of unresolved EXTERNAL NORMALs put out by the SOS Relocatable Loader. These are routines and drivers in the SOS Library that are unused and hence not included in the load module.

Figure 5-1. Sample $TTO Dialogue during Generation of a Master Reel

```
RLDR MTØ:2/S $PTR/4                          ← Step 3
LOAD $PTR, STRIKE ANY KEY
 TRIG2                                       ← CLI Trigger
LOAD $PTR, STRIKE ANY KEY                    ← SOSMT.LB
LOAD $PTR, STRIKE ANY KEY                    ← SOS.LB
LOAD $PTR, STRIKE ANY KEY                    ← CLI.RB
 SOSCL
XN .UØ4D ØØ547Ø
XN .UØ3D ØØ5467
XN .UØ2D ØØ5466
XN .UØ1D ØØ5465
XN .UØØD ØØ5464
XN .CTU7 ØØ5523
XN .CTU6 ØØ5522
XN .CTU5 ØØ5521
XN .CTU4 ØØ552Ø
XN .CTU3 ØØ5517
XN .CTU2 ØØ5516
XN .CTU1 ØØ5515
XN .CTUØ ØØ5514
XN .MTU7 ØØ5513
XN .MTU6 ØØ5512
XN .MTU5 ØØ5511
XN .MTU4 ØØ551Ø
XN .MTU3 ØØ55Ø7
XN .MTU2 ØØ55Ø6
XN .CTAD ØØ7323
XN .OPPP ØØ7321
XN .OPTP ØØ5463
XN .PLTD ØØ5472
    NMAX Ø17ØØØ
    ZMAX ØØØ223
    CSZE
     EST
     SST

OK
```

Figure 5-1. Sample $TTO Dialogue during Generation of a Master Reel (Continued)

```
# Ø:1
RLDR MTØ:3/S $PTR/4                    ← Step 4
LOAD $PTR, STRIKE ANY KEY              ← Step 5
 TRIG3                                 ← Editor Trigger
LOAD $PTR, STRIKE ANY KEY              ← SOSMT.LB
LOAD $PTR, STRIKE ANY KEY              ← SOS.LB
LOAD $PTR, STRIKE ANY KEY              ← EDIT.RB
 EDIT
XN .GDTE ØØ5236
XN .SDTE ØØ5234
XN .UØ4D ØØ527Ø
XN .UØ3D ØØ5267
XN .UØ2D ØØ5266
XN .UØ1D ØØ5265
XN .UØØD ØØ5264
XN .CTU7 ØØ5323
XN .CTU6 ØØ5322
XN .CTU5 ØØ5321
XN .CTU4 ØØ532Ø
XN .CTU3 ØØ5317
XN .CTU2 ØØ5316
XN .CTU1 ØØ5315
XN .CTUØ ØØ5314
XN .MTU7 ØØ5313
XN .MTU6 ØØ5312
XN .MTU5 ØØ5311
XN .MTU4 ØØ531Ø
XN .MTU3 ØØ53Ø7
XN .MTU2 ØØ53Ø6
XN .CTAD ØØ7123
XN .OPPP ØØ7121
XN .OPTP ØØ5263
XN .PLTD ØØ5272
XN .RTCD ØØ6367
XN .DLY  ØØ5232
XN .STDD ØØ5226
XN .GTDD ØØ523Ø
    NMAX Ø12555
    ZMAX ØØØ235
    CSZE
     EST
     SST

OK
```

Figure 5-1. Sample $TTO Dialogue during Generation of a Master Reel (Continued)

```
# Ø:1                                    <-- Step 6
RLDR MTØ:4/S $PTR/4                      <-- Step 7
LOAD $PTR, STRIKE ANY KEY                <-- ASM Trigger
 TRIG1
LOAD $PTR, STRIKE ANY KEY                <-- SOSMT.LB
LOAD $PTR, STRIKE ANY KEY                <-- SOS. LB
LOAD $PTR, STRIKE ANY KEY                <-- ASM. RB
 ASM
XN .GDTE ØØ5617
XN .SDTE ØØ5615
XN .UØ4D ØØ5651
XN .UØ3D ØØ5650
XN .UØ2D ØØ5647
XN .UØ1D ØØ5646
XN .UØØD ØØ5645
XN .CTU7 ØØ57Ø4
XN .CTU6 ØØ57Ø3
XN .CTU5 ØØ57Ø2
XN .CTU4 ØØ57Ø1
XN .CTU3 ØØ57ØØ
XN .CTU2 ØØ5677
XN .CTU1 ØØ5676
XN .CTUØ ØØ5675
XN .MTU7 ØØ5674
XN .MTU6 ØØ5673
XN .MTU5 ØØ5672
XN .MTU4 ØØ5671
XN .MTU3 ØØ567Ø
XN .MTU2 ØØ5667
XN .CTAD ØØ75Ø4
XN .OPPP ØØ75Ø2
XN .OPTP ØØ5644
XN .PLTD ØØ5653
XN .RTCD ØØ675Ø
XN .DLY  ØØ5613
XN .STDD ØØ56Ø7
XN .GTDD ØØ5611
    NMAX Ø16601
    ZMAX ØØØ364
    CSZE
     EST
     SST

OK
```

Figure 5-1. Sample $TTO Dialogue during Generation of a Master Reel (Continued)

```
ASM Ø $PTR/2                          <-- Step 8
LOAD $PTR, STRIKE ANY KEY             <-- Step 9 (No floating point instructions.)
LOAD $PTR, STRIKE ANY KEY             <-- Step 10
                                      <-- Step 11 (EXAMINE AC0.)
# Ø:4                                 <-- Step 12
NMAX: 17703                           <-- Value recorded in Step 11.
OK
# Ø:1                                 <-- Step 13
PLDR MTØ:5/S $PTR/4                   <-- Step 14
LOAD $PTR, STRIKE ANY KEY             <-- LFE Trigger
 TRIG3
LOAD $PTR, STRIKE ANY KEY             <-- SOSMT.LB
LOAD $PTR, STRIKE ANY KEY             <-- SOS.LB
LOAD $PTR, STRIKE ANY KEY             <-- LFE.RB
 STLFE
XN .GDTE ØØ5236
XN .SDTE ØØ5234
XN .UØ4D ØØ527Ø
XN .UØ3D ØØ5267
XN .UØ2D ØØ5266
XN .UØ1D ØØ5265
XN .UØØD ØØ5264
XN .CTU7 ØØ5323
XN .CTU6 ØØ5322
XN .CTU5 ØØ5321
XN .CTU4 ØØ532Ø
XN .CTU3 ØØ5317
XN .CTU2 ØØ5316
XN .CTU1 ØØ5315
XN .CTUØ ØØ5314
XN .MTU7 ØØ5313
XN .MTU6 ØØ5312
XN .MTU5 ØØ5311
XN .MTU4 ØØ531Ø
XN .MTU3 ØØ53Ø7
XN .MTU2 ØØ53Ø6
XN .CTAD ØØ7123
XN .OPPP ØØ7121
XN .OPTP ØØ5263
XN .PLTD ØØ5272
XN .PTCD ØØ6367
XN .DLY  ØØ5232
XN .STDD ØØ5226
XN .GTDD ØØ523Ø
    NMAX Ø14716
    ZMAX ØØØ26Ø
    CSZE
     EST
     SST

OK
```

Figure 5-1. Sample $TTO Dialogue during Generation of a Master Reel (Continued)

```
# 0:1                                    <---Step 15
RLDR MT0:6/S $PTR/4                      <---Step 16
LOAD $PTR, STRIKE ANY KEY                <---SYSG Trigger
 TRIG1
LOAD $PTR, STRIKE ANY KEY                <--- SOSMT.LB
LOAD $PTR, STRIKE ANY KEY                <--- SOS.LB
LOAD $PTR, STRIKE ANY KEY                <--- SYSG.RB
 SYSG
XN .GDTE 005617
XN .SDTE 005615
XN .U04D 005651
XN .U03D 005650
XN .U02D 005647
XN .U01D 005646
XN .U00D 005645
XN .CTU7 005704
XN .CTU6 005703
XN .CTU5 005702
XN .CTU4 005701
XN .CTU3 005700
XN .CTU2 005677
XN .CTU1 005676
XN .CTU0 005675
XN .MTU7 005674
XN .MTU6 005673
XN .MTU5 005672
XN .MTU4 005671
XN .MTU3 005670
XN .MTU2 005667
XN .CTAD 007504
XN .OPPP 007502
XN .OPTP 005644
XN .PLTD 005653
XN .RTCD 006750
XN .DLY  005613
XN .STDD 005607
XN .GTDD 005611
    NMAX 011041
    ZMAX 000135
    CSZE
     EST
     SST


OK
```

Figure 5-1. Sample $TTO Dialogue during Generation of a Master Reel (Continued)

APPENDIX A

ADDING A USER-SUPPLIED DEVICE HANDLER TO SOS

INTRODUCTION

This appendix is intended to ease the task of adding a special device handler to the SOS Library. There are effectively two types of devices that may be incorporated into SOS.

1. A type of device that requires interrupt dispatching provided by the system but which can otherwise be controlled outside of the system. By regulating the dispatching of interrupts, the system selectively controls the interruptibility of that device's interrupt service routine. Thus, certain devices may interrupt other devices under system control. Devices which have the property of requiring this interrupt service only from the system are hereafter referred to as Level One devices.

2. A type of device that requires the interrupt service described above, as well as base (non-interrupt) level control by the system. These devices are the most common; all SOS devices that perform I/O through system commands (.WRL, .RDL, .WRS, .RDS) fall into this category. These devices are hereafter referred to as Level Two devices.

Devices of both levels are currently supported by drivers in the SOS Library. The Real Time Clock is a Level One device; the high speed punch, $PTP, is a Level Two device. The clock requires interrupt service only from the system. The punch requires interrupt service as well as base level control in order to perform line and sequential output. Furthermore the interrupt service is necessary to synchronize the base level control. (It should be noted that an I/O device, as well as an interrupt only device, may be added to the system with Level One Support. Level One merely precludes the control of the device through system I/O commands; however, the user may wish to control the starting and stopping of a device through his own subroutines while relying on the system only to dispatch interrupts to him appropriately.)

The remainder of this appendix fully details the considerations to be made for adding device handlers to SOS. There are a number of existing links in SOS that facilitate these procedures (see page A-7). No source code within the SOS Library need ever be modified to add devices. However, several tables within the SOS Library may require changes. The modified tables need only be introduced to SOS at load time. Thus the procedures are in all cases reduced to the addition of new relocatable binary programs to a load module which includes the SOS Libraries.

SOS DEVICE HANDLING STRATEGY

In order to appreciate the procedures for incorporating device handlers into SOS, it is useful to understand the strategy that the system employs to control devices. This strategy can be analyzed in terms of its applicability to Level One and then Level Two devices.

Consider first Level One devices which must be serviced at the interrupt level only. (As previously mentioned, all system devices have this property.)

When an interrupt is detected by the system, the interrupted machine state must be saved. Next, the interrupting device must be identified in order to determine where to dispatch control to service this interrupt. Before control passes to the interrupt service routine however, the system must determine which devices may interrupt this routine and then appropriately enable that set of devices. The interrupt service routine may then perform any number of functions, ranging from several machine instructions to several hundred. (See SOS Interrupt Handling, page A-22.) The one common function of all interrupt service routines is to clear the interrupting device. When the interrupt service routine completes its functions, it returns control to the system interrupt dispatcher. This routine now must restore the previously saved machine state and return control to the program counter at the time of the interrupt.

This restored machine state, to which control is returned, may be either a base level program or another interrupt service routine that was interrupted. In the latter case, the above procedures beginning at the interrupted point would be repeated and the machine state finally restored would be that at the time that the former device interrupted. In this manner, interrupts are stacked and selectively processed by the system.

There are two variables which determine the relative priority with which individual interrupting devices will be serviced by the system:

1. The SOS Interrupt Search List which determines the order in which the interrupt dispatcher looks for interrupt service routines once an interrupting device is identified.

2. The interrupt mask which each individual device specifies when its interrupt service routine gets CPU control. This mask simply determines which devices may interrupt this device's interrupt service routine. (See DCT Mask Word DCTMS, page A-10.)

In general, these variables should be consistent among devices; that is, the first device in the Interrupt Search List should not be interruptible by any other devices, the second device in the list should only be interruptible by the first, and the last one should be interruptible by all other devices. This scheme cannot be rigidly adhered to because many devices share a priority bit in the interrupt mask. It is recommended that as nearly as possible device handlers observe these conventions. There is actually no interdependence between these variables however; if a user does not follow these conventions in adding his device handler, he will merely introduce slight inefficiencies into the processing of interrupts. The procedures for user specification of these device handler variables are discussed later in this appendix.

These considerations broadly summarize the information that must be provided to and used by the system to control Level One devices. To properly support Level Two devices, several additional pieces of information are required by the system:

1. The system must be able to associate the text string identifying the file/device with a physical SOS Channel Number whenever the device is opened. (This applies only to systems which use the RDOS to SOS Interface Program. When that program is not used, then the SOS Channel Number is specified in the ".OPEN" command).

2. The system must be able to map from the SOS Channel Number to a body of information which enables it to recognize the unique properties of the device, start it, stop it, and buffer I/O for it. This body of information is contained in the Device Control Table, DCT.

The DCT also contains much of the information that is used by the system interrupt dispatcher to service both Level One and Level Two devices at interrupt time. A DCT or some subset of it must therefore be provided for every unique device being incorporated into SOS, regardless of the device support level. The word by word layout of the DCT is described starting on page A-10. There are various means available to the user to link the DCT that he supplies into the system. These options are itemized and elaborated upon in the following section.

SOS DEVICE HANDLING STRATEGY (Continued)

The critical system requirements for each device level are summarized below:

Level One Devices:

1. A DCT
2. A position in the SOS Interrupt Search List
3. An interrupt service routine which performs all necessary interrupt level device functions and returns control appropriately to the system interrupt dispatcher.
4. A device clear routine which is called on all system initializations (.SYSI) and resets (.RESE).
5. All other necessary device control routines. These routines are never entered through the operating system however.

Level Two Devices:

1. A DCT
2. A position in the SOS Interrupt Search List.
3. An interrupt service routine (see above).
4. A device clear routine (see above).
5. A (physical) SOS Channel Number associated with the DCT. (Inherent in this requirement is a position in the SOS Channel Number Map.)
6. A file/device name associated with the SOS Channel Number. This only applies to devices which are running in an RDOS-SOS environment.
7. The routines to perform the appropriate subset of system commands (.OPEN , .CLOSE , .RDL , .WRL , .RDS and .WRS) meaningful to the device. These routines make partial or full use of the SOS global subroutines.

SOS LINKS FOR DEVICE HANDLERS

Once the user has determined what things are required by the system to add a device handler, it becomes necessary to know how to supply them. The answer requires a discussion of the links that are currently enbedded within the system for incorporating additional devices. The summary of the existing provisions within the system is as follows:

(1) The SOS Interrupt Search List has positions (in the form of unresolved EXTERNAL NORMAL symbols) for up to five addtional devices. These list positions are limited to two of high priority - after the Real Time Clock and before all other devices - and three of low priority - after all other system devices. If more positions are necessary, or if positions at other priorities are necessary, then the user may supply his own Interrupt Search List as described later.

(2) There are currently five unused SOS Channel Numbers available in the SOS Channel Number to Device Map. These Channel Numbers are 0-4, and there exists an unresolved EXTERNAL NORMAL symbol within the map for each of these numbers. Whenever one of these symbols is resolved, SOS expects the resolution of the symbol to be the start of a user supplied DCT. If these five available channels are inadequate, then the user may supply an additional Channel Number to Device Map for SOS Channel Numbers higher than 37_8. He must then supply the appropriate resolutions for the symbols in his high Channel Number Map. See SOS Channel Number to Device Map, page A-17.

(3) There are currently five unused file/device names corresponding to SOS Channels 0-4 in the SOS Filename Table:

UD00
UD01
UD02
UD03
UD04

If these names are unsatisfactory, they may be modified within the table by the user. If more names are required (corresponding to added SOS Channel Numbers) then the table may be expanded by the user. This table is easily accessible because it represents a separate logical record in the SOS library. By using the LFE, it can be replaced within the library. The one constraint placed on all device/file names is that they contain exactly four characters. See SOS Filename Table, page A-18.

The details for resolving these system links are given in the following sections. Table A-1 lists the existing system links and their limitations. Users adding device handlers are encouraged to adopt the simplest possible approach in incorporating these handlers. Table A-2 lists the general procedures that a user may follow to add a device handler according to the complexity of his needs.

Global Symbol	Meaning when Resolved	SOS Interrupt Search List Position	SOS Physical Channel Number	SOS Associated File/Device Name
.U00D	Start of a user supplied DCT.	After Real Time Clock; before all other devices.	0	UD00
.U01D	Start of a user supplied DCT	After UD00; before all other devices.	1	UD01
.U02D	Start of a user supplied DCT	After all other devices.	2	UD02
.U03D	Start of a user supplied DCT	After UD02	3	UD03
.U04D	Start of a user supplied DCT	After UD03	4	UD04
.OPPP	Points to a user supplied Table to be used during initialization to establish an alternative Interrupt Search List. See page A-14.	N/A	N/A	N/A
.OPTP	Points to a user supplied Table which supplements the SOS Channel Number Map. This table contains DCT pointers corresponding to high SOS Channel Numbers (channels greater than 37_8). See page A-17.	N/A	N/A	N/A
.FNAM	Points to the start of the File Name Table. This table may require modification or expansion to add user devices. See page A-18.	N/A	N/A	N/A

Table A-1 System Links for Adding User Device Handlers

Interrupt Service Requirements	SOS Channel Number Requirements	File/Device Name Requirements	Simplest User Approach
lowest priority	none (implies a Level One device).	none	Add the supplied DCT to the system at runtime through the system command ".IDEF".
any priority	none	none	Same as above.
highest priority	none	none	Supply a DCT beginning at the ENTRY point ".U00D".
lowest priority	any one number (implies a Level Two device).	none	Supply a DCT beginning at the ENTRY point ".U02D".; use SOS Channel 2 to access the device.
lowest priority	any two numbers	none	Supply DCTs at ".U02D" and ".U03D"; use SOS Channels 2 and 3 to access the devices.
lowest priority	any three numbers	none	Supply DCTs at ".U02D", ".U03D", and ".U04D"; use SOS Channels 2, 3, and 4 to access the devices.
highest priority	any one number	none	Supply a DCT at ".U00D"; use SOS Channel 0 to access the device.
highest priority	any two numbers	none	Supply DCTs at ".U00D" and ".U01D"; use SOS Channels 0 and 1 to access the devices.
any priority	any five or less numbers	none	Supply DCTs at ".U00D" through ".U04D" as required; use SOS Channels 0-4 to access the devices.
any priority	any five or less numbers	Any names (implies running in RDOS to SOS Interface environment).	Supply DCTs at ".U00D" through ".U04D" as required. Use the names "UD00", "UD01",..., "UD04" to open the device(s) on RDOS Channels.
specific priority other than lowest or highest	none	none	Supply the table beginning at the ENTRY point ".OPPP" which establishes the SOS Interrupt Search List; Supply the appropriate DCT.

Table A-2 User Approaches for Adding Device Handlers

Interrupt Service Requirements	SOS Channel Number Requirements	File/Device Name Requirements	Simplest User Approach
specific priority other than lowest or highest	any five or less	none	Supply the ".OPPP" table; supply DCTs ".U00D"-".U04D" as required. Use SOS Channels 0-4 as required.
specific priority other than lowest or highest	any five or less	any names	Supply the ".OPPP" table; supply DCTs ".U00D" - ".U004" as required; use the names "UD00" - "UD04" to open the devices on RDOS Channels.
any priority	any five or less	Names other than "UD00" - "UD04"	Supply DCTs at ".U00D" through ".U04D" as required. Replace the FNAME table in the SOS Library with a modified table containing the desired four letter names in the first 20_{10} character positions. Use these names to open the devices on RDOS Channels.
specific priority	any five or less	Names other than "UD00" - "UD04"	Supply the ".OPPP" table; then repeat the procedures above.
any priority including specific priorities	more than five	none	Supply the ".OPPP" table; Supply the Table beginning at ENTRY point ".OPTP" to map the added SOS Channel Numbers to DCTs. Supply the DCTs at ".U00D" - ".U04D" and also the additional DCTs as required. Use SOS Channels 0-4 and 40, 41, 42,... as required to access the devices.
any priority including specific priorities	more than five	Names other than and in addition to "UD00" - "UD04"	Supply the ".OPPP" table. Supply the ".OPTP" table. Supply the DCTs at ".U00D" - ".U04D" and also the additional DCTs as required. Replace the FNAME table with a modified, expanded table. Use these new names to open the devices on RDOS Channels.

Table A-2 User Approaches for Adding Device Handlers (Continued)

SOS DEVICE CONTROL TABLE (DCT)

Each SOS device requires a control table. Although some elements of the table may not be used by an added driver, the table must be defined exactly as in the following description, so that the critical elements reside at the correct displacements. This table requires 33 octal locations (displacements 0-32 from the DCT layout description in the SOS Parameter Tape). These displacements and their meanings are as follows:

Equivalence	Displacement	Meaning
0	DCTCD	The octal device code. Must be assembled into the table.
1	DCTMS	The mask of all lower priority devices, including this device. This mask is used to disable interrupts from all lower priority devices while processing an interrupt from this device. This mask should reflect the priorities established by the SOS Interrupt Search (see next section). The mask bits are defined in the SOS parameter tapes, and must be assembled into the table. See Figure A-2.
2	DCTCH	The active device characteristics from the RDOS User Parameter Tape. This word is derived by masking the complement of the user's AC1 on a ".OPEN" command with the device's fixed characteristics (see DCTFC). The device's fixed characteristics must be assembled into the table; they are not referenced for Level 1 devices.
3	DCTLK	The link to the next priority device, a pointer to its control table. This word is initialized by a .SYSI. The priorities are established by SOS Interrupt Search List.
4	DCTIS	The address of the interrupt service routine. The address must be assembled into the table. (See SOS Interrupt Handling, page A-22.

SOS DEVICE CONTROL TABLE (DCT) (Continued)

Equivalence	Displacement	Meaning
5	DCTIL	The interrupt frame links. This points to the DCT of the last interrupted device. This word is maintained by the SOS interrupt service dispatcher.
6	DCTDT	The Command Dispatch Table address for this device. The Command Dispatch Table must be ordered in the following manner: 0 - open routine address 1 - close routine address 2 - read/write sequential routine address 3 - read/write line routine address Any of the before mentioned functions that are illegal for a device should contain a -1 in their location. The address must be assembled into the table. (See Device Start, Stop, and Dispatch Routines, page A-23.
7	DCTST	The address of the device start routine. The address must be assembled into the table. See Device Start, Stop, and Dispatch Routines.
10	DCTSP	The address of the device stop routine. The address must be assembled into the table. See Device Stop, Start and Dispatch Routines.
11	DCTFL	The device flags. These flag bits are maintained by the global SOS subroutines. Three flags are currently defined: DCACT=1B15 - Device is active (executing I/O). Must be off to perform a SOS reset. DCACP=1B8 - A keyboard input device may accept a character.

SOS DEVICE CONTROL TABLE (DCT) (Continued)

Equivalence	Displacement	Meaning
11	DCTFL (continued)	DCKMD=1B0 - A keyboard input device is in echo mode. Echo the input character.
12	DCTBS	The size of the device buffer (in bytes for character devices, in words for full word devices). It must be assembled into the table.
13	DCTBF	Buffer first byte (word) address. If the device is a full word device (DCFWD characteristic), then this location must contain the beginning word address of the buffer. For character devices, this word must contain the beginning byte address of the buffer.
14	DCTBL	Buffer last byte (word) address.
15	DCTIP	Buffer current input pointer. For an output device, this is the byte address at which to store the next byte sent to the device from the user program. For an input device, this is the byte address at which to store the next byte received from the device. This word is maintained by the global SOS subroutines.
16	DCTOP	Buffer current output pointer. For an output device, this is the byte address from which to fetch the next byte for output. For an input device, this is the byte address from which to fetch the next byte requested by the user program. This word is maintained by the global SOS subroutines.
17	DCTCN	Count of active data in the buffer, i.e., bytes not yet sent to the device or bytes not yet moved to the user program for output and input devices respectively. This word is maintained by the global SOS subroutines.

SOS DEVICE CONTROL TABLE (DCT) (Continued)

Equivalence	Displacement	Meaning
20	DCTTO DCTCC	Timeout constant (all input devices). Column counter (all output devices). For input devices this word represents the maximum time interval during which they may have outstanding data following a start pulse. The parameter "SCTIM" defined on the RDOS User Parameter Tape corresponds to a time of 1 millisecond on the Supernova SC. Then, if a device requires 6 milliseconds to timeout, the word can be assembled as: 6*SCTIM For output devices, this word is maintained by the global SOS subroutines.
21	DCTRC	Restart constant (all input devices). Line counter (all output devices). For input devices, if the active data count is less than this constant, another start pulse should be sent to the device. This word must be assembled into the table. For output devices, this word is maintained by the global SOS subroutines.
22	DCTAT	Device attributes. Fixed bit settings that are always returned to the user in AC0 on a .GTATR command. Attributes include attribute protected, permanent, read protected, and write protected. See RDOS User Parameter Tape.
23	DCTFC	Device fixed characteristics. These characteristics, from the RDOS User Parameter Tape, always become the active characteristics (DCTCH) after the device is OPENed, unless they are suppressed by the AC1 mask.

SOS DEVICE CONTROL TABLE (DCT) (Continued)

Equivalence	Displacement	Meaning
24-32		Device Interrupt Frame. The machine state at the time of a device interrupt is saved in these DCT locations. The layout of the interrupt frame is as follows:
24	IAC0	Saved AC0.
25	IAC1	Saved AC1.
26	IAC2	Saved AC2.
27	IAC3	Saved AC3.
30	IPC	Program Counter. (Location 0 when the interrupt was taken.)
31	IRLOC	Volatile SOS linkage cell.
32	IMSK	Interrupt enable mask when the interrupt was taken. The carry bit is saved in bit 14 of this word.

DCT displacements 6-23 (DCTDT - DCTFC) are not referenced by Level 1 devices, with the exception of DCTFL, bit 15, which must be off to perform a SOS reset command, and DCTSP, which is executed on .SYSI and .RESET commands. For Level 2 devices, the Dispatch Table (DCTDT) must be defined. Use of the remaining elements depends on the definitions of this table; if any of the global SOS routines are invoked, then any or all of these elements may be referenced.

SOS INTERRUPT SEARCH LIST

This list is established when a SOS initialization (system command ".SYSI") is performed; this "list" itself is merely the order in which DCTs are linked together. (See DCT Link Words, page A-10.) The order in which the DCTs are linked determines the order in which the SOS devices are searched for a matching code on an interrupt. This link order is derived from a table that is normally embedded in SOS-MAIN. If the EXTERNAL NORMAL symbol .OPPP in SOS-MAIN is resolved however, then a user supplied table is used to establish the links. The table consists of a list of pointers to DCT addresses. Each of the pointers is an ENTRY point in SOS-MAIN. The added level of pointers makes it possible for the user to define this

table and load it before the SOS library without affecting the conditional loading of any of the device drivers. The SOS-MAIN table is set up as follows:

```
.RTCP
.U00P
.U01P
.PTRP
.CDRP
.MTAP
.CTAP
.TTRP
.TTIP
.TI1P
.PTPP
.LPTP
.PLTP
.TTOP
.TO1P
.U02P
.U03P
.U04P
 0          Table Terminator
```

Each of the symbols ending in "P" is declared an ENTRY in the SOS-MAIN program. Each ENTRY point contains the corresponding DCT address. Whenever the DCT is not loaded, the next symbol is examined to find a loaded DCT to link to the previous one. The table is always terminated with a zero word. This table reflects a descending priority level of SOS devices, beginning with the Real Time Clock and ending with user defined device number 4. If the user wishes to use this existing table to establish his device priorities, he must accept a priority that is either higher than all devices except the Real Time Clock or else lower than all system devices. Up to five devices may be defined; two may occupy the highest priority positions and the other three occupy the low priority positions. The user must then assign the appropriate (low) SOS Channel Numbers to the devices if they are Level Two devices. If the user wishes device priorities other than these, he must supply his own table, beginning at ENTRY point .OPPP .

As an example of a user supplied table, consider the addition of six devices, UD00 - UD05. UD00, UD02, and UD04 require the top three priority slots respectively. UD01 and UD03 may occupy the lowest priority slots and UD05 requires a priority higher than the Magnetic Tape Controller but lower than the Card Reader.

SOS INTERRUPT SEARCH LIST (Continued)

The critical program declarations to achieve this priority scheme would appear as follows:

```
        .ENT    .OPPP
        .EXTN   .PTRP, .CDRP, .TTRP, .PTPP, .LPTP
        .EXTN   .TTOP, .MTAP, .CTAP, .TTIP, .TI1P
        .EXTN   .PLTP, .TO1P, .RTCP


        .NREL
.OPPP:  .U00P
        .U02P
        .U04P
        .RTCP
        .PTRP
        .CDRP
        .U05P
        .MTAP
        .CTAP
        .TTRP
        .TTIP
        .TI1P
        .PTPP
        .LPTP
        .PLTP
        .TTOP
        .TO1P
        .U01P
        .U03P
        0           ;table terminator
        .
        .
        .
.U00P:  .U00D       ;pointer to UD00 DCT
.U01P:  .U01D       ;pointer to UD01 DCT
.U02P:  .U02D       ;pointer to UD02 DCT
.U03P:  .U03D       ;pointer to UD03 DCT
.U04P:  .U04D       ;pointer to UD04 DCT
.U05P:  .U05D       ;pointer to UD05 DCT
```

All pointer symbols for device drivers that are never desired may be omitted from this table.

SOS CHANNEL NUMBER TO DEVICE MAP

This table is referenced by the SOS command dispatch routine and is normally embedded in the SOS-MAIN program. Within this table, there are SOS Channel Numbers available for up to five additional devices. These SOS Channels are 0, 1, 2, 3, and 4. If this number of positions is adequate and the SOS Channel numbers acceptable then the user need not define his own map. He needs merely to resolve the DCT addresses that are declared EXTERNAL NORMALS in the SOS-MAIN program. This is done by declaring the symbol an ENTRY point and resolving it as the beginning location of the DCT. Hence to add five devices UD00 - UD04 on SOS Channels 0-4, the following declarations would be necessary:

```
                .ENT            .U00D, .U01D, .U02D, .U03D, .U04D
                 .
                 .              ; These names must be used for the DCT
                 .              ; beginning locations
                .NREL

                ;UD00 DCT
.U00D:          60              ; device code 60
                -1              ; mask out all other devices
                 .
                 .

                ;UD01 DCT
.U01D:          61              ; device code 61
                 .
                 .

                ;UD02 DCT
.U02D:          62              ; device code 62
                 .
                 .

                ;UD03 DCT
.U03D           63              ; device code 63
                 .
                 .

                ;UD04 DCT
.U04D:          64              ; device code 64
                 .
                 .
```

If more than five devices are being added or if SOS Channel Numbers other than 0-4 are desired for the devices, then the Channel-Number-to-Device Map must be supplemented by the user. The supplementary map is referenced when a Channel

SOS CHANNEL NUMBER TO DEVICE MAP (Continued)

Number outside of the legal SOS range has been specified in an I/O command and the EXTERNAL NORMAL symbol .OPTP has been resolved. The address contained at this location is expected to point to a list of DCT addresses which correspond to devices using SOS Channel Numbers beginning at 40 (HCHNO+1 from the SOS parameters). Thus if a user wishes to add 8 devices on SOS Channels 0-4 and 40-42, besides the declarations shown above that are necessary for five of the eight devices, the following declarations are necessary for the remaining 3 devices (UD05 - UD07):

```
                .ENT        .OPTP
                 .
                 .
                 .
                .NREL

.OPTP:          .+1
.U05P:          .U05D       ; pointer to U05 DCT
.U06P:          .U06D       ; pointer to U06 DCT
.U07P:          .U07D       ; pointer to U07 DCT
```

Devices UD05 - UD07 would then occupy SOS Channel positions 40-42. Only Level Two devices require the Channel-Number-to-Device Map. The supplementary map is expandable from SOS Channel Number HCHNO+1 to number 76 .

SOS FILENAME TABLE

This table is used by the SOS command dispatch routine whenever a file/device is .OPENed in an RDOS-SOS environment. The table is simply a text string which begins at the ENTRY point .FNAM . A zero word terminates the table. Each four-character file/device name recognized by SOS is contained in this table; the first four character string represents the SOS Channel 0 device, followed by the SOS Channel 1 device, and so on to the SOS Channel 37 device. Whenever a ".OPEN" command is executed, this table produces the SOS Channel Number from the passed file/device name. The user may wish to modify this table in one or both of two ways:

1. By changing one or more of the existing names to a more descriptive name for his own device(s).
2. By adding names at the end of the table which correspond to devices added on SOS Channels 40, 41, etc. (See previous section.)

In either case, new filenames must be exactly four characters in length.

SOS FILENAME TABLE (Continued)

When the table has been modified and assembled, the user may REPLACE the existing SOS Library table by using the LFE and generating a new library. This table is only necessary for users running in an RDOS-SOS environment. Figure A-1 shows the current FNAME program in the SOS Library.

```
0001 FNAME MACRO REV 02                    12:13:32 04/22/74
                        .TITLE  FNAME   ;RDOS FILE NAME STRINGS
02                      .ENT    .FNAM
03                      .NREL
04      000001          .TXTM   1
05      000001          .TXTN   1
06
07  000001 052504 .FNAM: .TXT    /UD00UD01UD02UD03UD04TTI1/
08      030060
09      052524
10      030061
11      052504
12      030062
13      052524
14      030063
15      052524
16      030064
17      052124
18      044461
19  000014 022120         .TXT    /$PLTTT01$TTP$CDR$TTO$TTI$LPT$PTR$PTP$TTR/
20      046124
21      052124
22      047461
23      022124
24      052120
25      022103
26      042122
27      022124
28      052117
29      022124
30      052111
31      022114
32      050124
33      022120
34      052122
35      022120
36      052120
37      022124
38      052122
39  000041 046524         .TXT    /MT0:MT1:MT2:MT3:MT4:MT5:MT6:MT7:/
40      030072
41      046524
42      030472
43      046524
44      031072
45      046524
46      031472
47      046524
48      032072
49      046524
50      032472
51      046524
52      033072
53      046524
54      033472
55  000066 041524         .TXT    /CT0:CT1:CT2:CT3:CT4:CT5:CT6:CT7:/
56      030072
57      041524
58      030472
59      041524
60      031072
```

Figure A-1 SOS FNAME Program

```
 1MAC FNAME
01          041524
02          031472
03          041524
04          032072
05          041524
06          032472
07          041524
08          033072
09          041524
10          033472
11          [illegible]          0                    ;TABLE TERMINATOR
12
13                               .END
```

Figure A-1 SOS FNAME Program

SOS INTERRUPT HANDLING

When an interrupt is taken, the SOS interrupt dispatcher preserves the interrupted machine state. The DCT interrupt frame (IAC0 - IMSK) is utilized in these procedures, and when the device interrupt handler gains control, the "save" is complete and the mask in the device's Control Table (DCTMS) is active. The following functions are the responsibility of the interrupt routine (DCTIS):

1. Clearing the done flip-flop in the device.

2. Storing/retrieving the next character in the device buffer.

3. Restarting the device when appropriate.

4. Returning to the SOS interrupt module.

The SOS stack mechanism may not be invoked at interrupt processing time. The SOS modules .IBUF and .OBUF may be called, however, since they do not require a stack frame (see SOS global subroutines).

Two simple interrupt routines, one for output device $PTP and one for input device $PTR, illustrate the above points:

```
PTRS:   DIAC    1,PTR       ;RETRIEVE CHARACTER/
                            ;CLEAR DONE
        JSR     @ADRIB      ;STORE CHARACTER IN THE
                            ;DEVICE'S BUFFER
        JMP     .           ;AN IMPOSSIBLE RETURN-
                            ;BUFFER ALREADY FULL
        NIOS    PTR         ;RESTART $PTR.  THE
                            ;BUFFER IS NOT YET FULL
        JMP     @.+1        ;DON'T RESTART $PTR.  THE
                            ;BUFFER BECAME FULL
        .EXTN   .DISM
        .DISM               ;RETURN TO THE SOS
                            ;INTERRUPT MODULE
        .EXTN   .IBUF
ADRIB:  .IBUF               ;ENTRY POINT
```

```
PTPS:   NIOC    PTP         ;CLEAR DONE
        JSR     @ADROB      ;RETRIEVE NEXT CHARACTER
                            ;FROM THE DEVICE'S BUFFER
        DOAS    1,PTP       ;RESTART DEVICE AND SEND THIS
                            ;CHARACTER IF THE RETURN CAME
                            ;HERE
        JMP     @.+1        ;OTHERWISE DON'T RESTART
        .DISM               ;RETURN TO THE SOS INTERRUPT
                            ;MODULE
        .EXTN   .OBUF       ;
ADROB:  .OBUF               ;ENTRY POINT
```

DEVICE START, STOP, AND DISPATCH ROUTINES

Device Start Routine

The address of this routine is at displacement DCTST in the DCT. For output devices, this routine should send a start pulse plus the character from AC1. If the device will not interrupt as a result of this action, return to one location beyond the normal return location. Otherwise, return to the normal location. AC3 points to the normal return location. As an illustration, consider the line printer start routine:

```
LPTST:  DOAS    1,LPT       ;START LPT, OUTPUT THE
                            ;CHARACTER
        SKPBZ   LPT         ;WILL IT INTERRUPT?
        JMP     0, 3        ;YES
        JMP     1, 3        ;NO
```

For input devices, this routine should send a start pulse and return to the normal return location. For example, the $PTR start routine is:

```
PTRST:  NIOS    PTR         ;SEND START PULSE
        JMP     0, 3        ;RETURN
```

Device Stop Routine

The address of this routine is at displacement DCTSP in the DCT. This routine should simply send a clear pulse and return to the normal return location. Using the $PTR as an example:

Device Stop Routine (Continued)

```
PTRSP:  NIOC    PTR     ;SEND CLEAR PULSE
        JMP     0, 3    ;RETURN
```

This routine is executed for each device on any .RESET or .SYSI.

Device Dispatch Routines

The device dispatch table address is at displacement DCTDT in the DCT as previously described. If the global SOS routines are not invoked for any of the four functions, then the following points should be noted in the dispatch routine:

1. AC3 points to the error return location.
   Increment by one for a success return.

2. The contents of the user accumulators are:

   AC0 - page zero displacement "CAC0"
   AC1 - page zero displacmenet "CAC1"
   AC2 - UST displacement "USTA2"

   If an accumulator is being returned to the user, then the appropriate location must be changed.

3. AC2 points to the DCT when the dispatch routine gains control.

All of the above equivalences are defined in the SOS Parameter Tape.

If the global SOS routines are used, then they may be invoked directly (by assembling their addresses into the dispatch table).

SOS LINKAGE

A simple stack mechanism is employed in SOS at the non-interrupt level. This mechanism provides a means of saving the calling routine's accumulators and of operating on variables stored in the stack. No stack mechanism is provided for interrupt processing but the state of the current stack must be preserved whenever an interrupt is processed. The SOS interrupt dispatch routine performs this service.

The linkage scheme used in SOS makes use of several page zero locations and of a fixed block of core assembled into the SOS-MAIN module. The size of this core block permits a "depth" of six calls from the common SOS entry point at the start of

the program. The page zero locations and equivalences that are used in the linkage mechanism include:

SAVE - (JSR @3) invokes the routine which saves accumulators and updates the stack pointer.

RTRN - (JMP @4) invokes the routine which restores the caller's accumulators and returns to him.

CSP - (Page Zero Location) always points to the stack frame currently in use.

RLOC - (Page Zero Location) used as a temporary by the SAVE routines; may also be used as a temporary by any routine in lieu of allocating a stack frame.

The stack frame is a fixed size with the following displacements defined:

RTLOC - The return location in the current subroutine (location which it last "called").

AC0 - Contents of accumulator 0 of the current subroutine at the last call which it made.

AC1 - Same as above; accumulator 1.

TMP - Available for use by the current subroutine as a temporary.

OAC0 - Contents of caller's accumulator 0.

OAC1 - Contents of caller's accumulator 1.

OTMP - Caller's temporary.

ORTN - Caller's return location.

Following a SAVE or a RTRN, AC3 always points to the current stack frame, and each of the above locations may be referenced as displacements from it. Otherwise, the CSP may be loaded into an index accumulator in order to reference the locations. (Accumulator 2 is never saved since it usually contains a Device Control Table address and is passed as an argument from subroutine to subroutine.)

The typical procedures executed in using the mechanism are as follows:

1. A routine (B) is called by another routine (A) through the JSR instruction:

```
        JSR     B
            or
        JSR     @ADDRB
        :
        .
ADDRB:  B
```

2. Routine B saves the caller's return location when it begins execution with:

```
        STA     3,@CSP
```

3. Routine B may, at any time thereafter, perform a:

```
        SAVE
```

   to save the caller's accumulators, allocate a stack frame, and update the CSP appropriately.

4. To return to A, routine B simply performs a:

```
        RTRN
```

   If no frame is required by B, it may save the caller's return locations in RLOC:

```
        STA     3,RLOC
```

   It may then use any accumulator and operate on A's frame (by loading CSP into an index accumulator) if it wishes. It should perform the return in the following manner:

```
        LDA     3,CSP
        JMP     @RLOC
```

   so that when A regains control, accumulator 3 is pointing to its frame. Note that the accumulators upon this return are exactly as they were left by routine B, rather than as they were when A called B. All of the above equivalences are defined in the SOS Parameter Tape.

GLOBAL SOS SUBROUTINES

The global routines defined as ENTRIES in SOS are as follows:

.OPN	-	Open
.CLS	-	Close
.WRSE	-	Write sequential
.WRLI	-	Write line
.RDSE	-	Read sequential
.RDLI	-	Read line
.ACHR	-	Send a character
.RCHR	-	Read a character
.IBUF	-	Input a character to buffer
.OBUF	-	Output a character from buffer
.STB	-	Store a byte
.LDB	-	Load a byte
.DISM	-	Dismiss an interrupt
.IDCT	-	Initialize DCT
.SYSE	-	System error

GENERALIZED SOS SUBROUTINES

These routines are available for use with any programs loaded with SOS. The calling procedures and brief descriptions are given below:

.OPN

Calling Sequence: JSR .OPN

(All references to JSR .XXX instructions, where .XXX is an entry point, are equivalent to the following instruction sequence:

```
        JSR     @XXX
         .
         .
         .
XXX:    .XXX                                    )
```

Arguments: AC0 = Byte address of the file name if the device is an intervention device. Otherwise, AC0 is ignored.

AC2 = Pointer to the DCT.

Return Sequence: Always returns to calling location +2 with the accumulators unchanged.

.OPN (Continued)

Description: The device's active characteristics are derived by ANDing the complement of the user's AC1 and the device's fixed characteristics (DCTFC); they are stored at DCTCH. Then, if the device being opened is an intervention device, a prompt message is typed. If the device requires leader/trailer, it is punched. The device is always cleared (DCT stop routine) and the Device Control Table is initialized.

.CLS

Calling Sequence: JSR .CLS

Arguments: AC2 = Pointer to the Device Control Table.

Return Sequence: Always returns to calling location +2 with the accumulators unchanged.

Description: If the device being closed requires leader/trailer, it is punched. When the device is no longer active, it is cleared and its Device Control Table is initialized.

.WRSE

Calling Sequence: JSR .WRSE

Arguments: "CAC0" * = Beginning byte address for transfer

"CAC1" * = Byte count for transfer

AC2 = Pointer to Device Control Table

Return Sequence: Always returns to calling location +2 with the accumulators unchanged.

Description: The specified number of bytes are inserted into the device's buffer for output.

* Page zero displacements from the SOS Parameter Tape.

.WRLI

Calling Sequence: JSR .WRLI

Arguments: CAC0 = Beginning byte address for the transfer.

AC2 = Pointer to the Device Control Table.

Return Sequence: The accumulators are unchanged, except CAC1 which contains the count of bytes written. A return to the calling location +1 indicates the maximum line length was exceeded. A return to calling location +2 is normal.

Description: The specified ASCII characters are inserted into the device's buffer for output. The character string that is output is terminated by:

1. carriage return
2. form feed
3. null byte

Line editing is done, based on the characteristics of the device.

.RDSE

Calling Sequence: JSR .RDSE

Arguments: CAC0 = Beginning byte address for the transfer

CAC1 = Byte count for the transfer

AC2 = Pointer to Device Control Table

Return Sequence: A return to calling location +1 indicates end of file. A return to calling location +2 is normal. The accumulators are unchanged. CAC1 contains the partial count read on an EOF return.

Description: The specified number of bytes are placed in the user area from the device's buffer.

.RDLI

Calling Sequence: JSR .RDLI

Arguments: CAC0 = Beginning byte address for the transfer.

AC2 = Pointer to Device Control Table.

Return Sequence: A return to calling location +1 indicates either end of file, line length exceeded, or parity error. In this case, CAC1 contains the partial count of bytes read and CAC2** contains the error code. A return to calling location +2 is normal, with accumulators unchanged except CAC1, which contains the count of characters read.

Description: The specified ASCII characters are inserted into the user area from the device's buffer. The character string is terminated by:

1. carriage return
2. form feed
3. null

Rubouts and line feeds are ignored.

.RCHR

Calling Sequence: JSR .RCHR

Arguments: AC2 = Pointer to Device Control Table.

** UST displacement USTA2 from the SOS Parameters (PARA.SR).

.RCHR (Continued)

Return Sequence: A return to calling location +1 indicates device timeout (end of file). In this case, the accumulators are unchanged with the error code in CAC2. A return to calling location +2 is normal. In this case, the right justified byte (word)* that is read is in AC1 with the other accumulators unchanged.

Description: The next available byte is read from the device's buffer into AC1. If necessary, a start pulse is issued to the device.

.ACHR

Calling Sequence: JSR .ACHR

Arguments: AC1 = Right justified byte to be sent.

AC2 = Pointer to the Device Control Table.

Return Sequence: A return is always made to calling location +1 with the accumulators unchanged.

Description: The byte is inserted into the device's buffer for output. If necessary, a start pulse is issued to the device.

.IBUF

Calling Sequence: JSR .IBUF

Arguments: AC1 = Right justified byte to be inserted into buffer.

AC2 = Pointer to Device Control Table.

*Input devices with the characteristic, DCFWD, always operate on words rather than bytes.

.IBUF (Continued)

Return Sequence: A return to the calling location +1 indicates the buffer is already full. A return to calling location +2 indicates the character was inserted and the buffer is not full. A return to calling location +3 indicates the character was inserted and the buffer became full. In every case, the accumulators are unchanged, except AC0, which is destroyed.

Description: The byte (word) is placed in the appropriate buffer slot and the Device Control Table is updated accordingly. This routine is used at the interrupt processing level by input devices and at the non-interrupt level by output devices.

.OBUF

Calling Sequence: JSR .OBUF

Arguments: AC2 = Pointer to Device Control Table.

Return Sequence: A return to calling location +1 indicates the buffer is empty. A return to calling location +2 indicates the buffer is not empty. In this case, the next available byte (word) is fetched from the buffer and returned right justified in AC1. In both cases, AC0 is destroyed and the other accumulators are unchanged.

Description: The byte (word) is fetched from the appropriate buffer slot into AC1 and the Device Control Table is updated accordingly. This routine is used at the interrupt processing level by output devices (to fetch their next byte for output) and at the non-interrupt level by input devices (to retrieve bytes from their buffers).

.STB

Calling Sequence: JSR .STB

Arguments: AC0 = Destination byte address.

AC1 = Right justified byte.

.STB (Continued)

Return
Sequence: The return is always made to calling location +1 with AC0 incremented and the other accumulators unchanged.

Description: The passed byte is stored at the specified address.

.LDB

Calling
Sequence: JSR .LDB

Arguments: AC0 = Source byte address.

Return
Sequence: The return is always made to calling location +1 with the right justified byte in AC1. The other accumulators are unchanged.

Description: The byte at the specified address is loaded and returned in AC1.

.DISM

Calling
Sequence: JMP .DISM

Arguments: AC2 = Pointer to Device Control Table.

Return
Sequence: No return.

Description: This routine restores the machine to the state it was in before the device interrupted. Control is passed to this point when the interrupt from the device has been serviced.

.IDCT

Calling
Sequence: JSR .IDCT

Arguments: AC2 = Pointer to the Device Control Table.

.IDCT (Continued)

Return
Sequence: Always returns to calling location +1 with the accumulators unchanged.

Description: The device stop routine (DCT displacement DCTSP) is called; DCT locations DCTCN and DCTFL are zeroed; DCT locations DCTOP and DCTIP are set to the contents of DCTBF.

.SYSE

Calling
Sequence: JMP .SYSE

Arguments: AC0 = System Error Code to be returned to ".SYSTM" caller.

Return
Sequence: No return.

Description: An error return is made to the last system command with the passed error code stored in caller's AC2.

DEVICE DRIVER EXAMPLE

Figure A-2 shows the device driver for the high speed paper tape punch ($PTP). This driver is similar to those of other character oriented output devices. The listing shows (in order) the:

1. Global declarations
2. Device Control Table
3. Interrupt service routine
4. Device start routine
5. Device stop routine
6. Device Dispatch Table
7. Device I/O buffer

```
!0002 PTPDR
                        .TITLE PTPDR     ; $PTP DRIVER                          1
02                      .ENT .PTPD                                              1
03                      .EXTN .OBUF,.OPN,.CLS,.WRSF,.WRLT,.DISM                 1
04                      .NREL                                                   1
05
06      000050 PTPSZ=   50               ; BUFFER CHARACTER SIZE
07
08
09  00000'000013 .PTPD: PTP                                                     2
10  00001'000015        MSLPT+MSTTO+MSPTP        ; MASK                         2
11  00002'000001        .BLK 1           ; TEMPORARY CHARACTERISTICS            2
12  00003'000001        .BLK 1           ; LINK                                 2
13  00004'000033'       PTPS             ; INTERRUPT SERVICE ADDRESS            2
14  00005'000001        .BLK 1           ; INTERRUPT SAVE-STATE LINK            2
15  00006'000045'       PTPDT            ; DISPATCH TABLE ADDRESS               2
16  00007'000040'       PTPST            ; START ADDRESS                        2
17  00010'000042'       PTPSP            ; STOP ADDRESS                         2
18  00011'000001        .BLK 1           ; FLAGS                                2
19  00012'000050        PTPSZ            ; BUFFER SIZE                          2
20  00013'000122"       PTPB*2           ; BUFFER BEG. BYTE ADDRESS             2
21  00014'000172"       PTPB*2+PTPSZ     ; BUFFER END. BYTE ADDRESS             2
22  00015'000003        .BLK 3           ; VARIABLE INFORMATION                 2
23  00020'000002        .BLK 2           ; COLUMN AND LINE COUNTERS             2
24  00022'100002        ATPER+ATRP       ; READ PROTECTED                       2
25  00023'001301        DCLAC+DCCPO+DCNAF+DCRAT ; FIXED CHARACTERISTICS         2
26  00024'000007        .BLK IFRL        ; INTERRUPT FRAME                      2
27
28  00033'060213 PTPS:  NIOC PTP         ; CLEAR THE PUNCH                      3
29  00034'006410        JSR @OBUFX       ; GIVE ME ANOTHER CHAR.                3
30  00035'065113        DOAS 1,PTP       ; OUTPUT IT, IF I GOT IT               3
31  00036'002401        JMP @.+1         ; ALL DONE                             3
32  00037'177777        .DISM                                                   3
33
34
35  00040'065113 PTPST: DOAS 1,PTP       ; START UP DEVICE                      4
36  00041'001400        JMP 0,3                                                 4
37
38  00042'060213 PTPSP: NIOC PTP         ; STOP DEVICE                          5
39  00043'001400        JMP 0,3                                                 5
40  00044'177777 OBUFX: .OBUF            ; ENTRY POINT                          5
41
42
43  00045'177777 PTPDT: .OPN             ; DISPATCH TABLE                       6
44  00046'177777        .CLS                                                    6
45  00047'177777        .WRSF                                                   6
46  00050'177777        .WRLT                                                   6
47  00051'000024 PTPB:  .BLK PTPSZ/2                                           7
48
49                      .END
```

Figure A-2. Device Driver for $PTP

APPENDIX B

USER APPLICATION ROUTINES

SAVE-RESTORE PROGRAM (SAVRE)

This program maintains a user-supplied stack to save the caller's registers (3 accumulators, 2 temporary storage locations, and the caller's return location) when a subroutine is called, and the program restores the information on the stack upon returns. The User Stack Pointer (USP) always points to the executing subroutine's stack frame. This subroutine may then access any of the caller's registers, as well as any of its own, by using the stack displacements defined in the SOS Application Parameter Tape (Appendix C).

The user must supply the stack before any of the SOS user application routines can be used. To provide the stack, the user merely stores a beginning address for the stack into the USP. The address stored in the cell must point to a core block large enough to meet the user stack requirements. No stack overflow check is made. If the stack is used only with the SOS library routines, then the value of SOSEC in the SOS User Application Parameter Tape gives the maximum number of 6-location frames necessary.

The various features of the SAVRE program are illustrated in the following example of step-by-step use of the stack:

1. To supply the stack, the user may allocate a fixed block of core starting at NMAX:

```
        :
        .SYSTM          ;GET NMAX INTO AC1
        .MEM
        JMP .           ;ERROR
        STA 1, USP      ;INITIALIZE START OF STACK
        LDA 0, SRKSZ
        .SYSTM          ;UPDATE NMAX
        .MEMI
        JMP .           ;NOT ENOUGH CORE AVAILABLE
        :
STKSZ:  SOSEC+3*SSEL    ;TOTAL STACK SIZE = SOS ENTRY COUNT +
                        ;USER'S ENTRY COUNT * ENTRY LENGTH
```

1. (Continued)

   Or, instead of starting at NMAX, the user may alternatively allocate any fixed block of core:

```
        LDA 0, STACK        ;LOAD STACK POINTER
        STA 0, USP          ;STORE IN USP
             :
             :
STACK:  .+1
        .BLK SOSEC*SSEL     ;SOS SIZE IS ADEQUATE
```

2. To save the caller's registers when a subroutine is entered via a JSR SUBR instruction:

```
        .EXTN SAVR, RETR
SUBR:   STA  3, @USP        ;SAVE-RETURN LOCATION IN STACK
        SAVR                ;SAVE REGISTERS
          :
          :                 ;AC3 ⟶ SUBR's FRAME
```

3. To access the caller's registers (which are intact when the subroutine is entered):

```
STA 2, OAC0, 3          ;RETURN AC2 IN CALLER'S AC0
ISZ OAC2, 3             ;INCREMENT THE RETURNED AC2
ISZ ORTR, 3             ;INCREMENT RETURN LOCATION COUNTER
DSZ OT1, 3              ;DECREMENT CALLER'S TEMPORARY
STA 1, T0, 3            ;STORE MY OWN TEMPORARY
```

4. To return to the calling subroutine:

```
RETR                    ;RETURN TO CALLER
```

5. To pop multiple stack levels, i.e., make an exceptional return, the external command, ERETR, has been defined. To illustrate use of this feature, consider five subroutines: A, B, C, D, and E, executing at three levels: 1, 2, and 3.

   Subroutine A executes at level 1. Subroutines B, C, and D execute at level 2. Subroutine E executes at level 3. Subroutine A calls B, C, and D, each of which may call E.

5. (Continued)

When a special condition is encountered by B, C, D, or E, subroutine A may want to regain immediate control. This is achieved by performing the ERETR in any of these subroutines provided that subroutine A has performed the logic shown below.

```
        .EXTD ERAD, ERSUP
        .EXTN ERETR
A:      STA 3, @USP          ;SAVE REGISTERS
        SAVR
          .
          .
          .
        LDA 0, AERAD
        STA 0, ERAD          ;STORE EXCEPTIONAL RETURN ADDRESS
        STA 3, ERUSP
          .
          .
          .
        JSR  B
          .
          .
          .
        JSR  C
          .
          .
          .
        JSR  D
          .
          .
          .
AERAD:  SPECR                ;EXCEPTION RETURN ADDRESS
          .
          .
          .
SPECR:  LDA 0, VAR           ;REGAIN CONTROL HERE ON EXCEPTIONAL
        STA 0, OAC1, 3       ;CONDITION AND CONTINUE
          .
          .
          .
  VAR:  .BLK  1
```

This mechanism returns control unconditionally to location SPECR upon the execution of an ERETR by subroutine B, C, D, or E. The stack pointer will be A's original pointer and the caller's registers are unchanged but the registers returned to A are indeterminate. One case in which this feature could be used is in string processing, where special characters may require special processors or signal the termination of a set of processors.

SAVE-RESTORE PROGRAM (SAVRE) (Continued)

5. (Continued)

Following is a list summarizing the external declarations necessary to use the Save-Restore program:

```
.EXTN SAVR, RETR        ;USE OF THESE FEATURES IS OPTIONAL

.EXTN ERETR             ;USE OF THIS FEATURE IS OPTIONAL.
                        ;HOWEVER, IF USED, IT REQUIRES THE
                        ;EXTERNAL DISPLACEMENTS THAT FOLLOW

.EXTD ERUSP, ERAD       ;MUST BE DECLARED AND INITIALIZED
                        ;TO USE EXCEPTIONAL RETURN
```

COMMAND TABLE BUILDER (CTB) PROGRAM

This program reads a command line from the $TTI into a user-supplied core block and dissects the line into a table of string (byte) pointers and flag bit settings. Blank characters in the line are considered string separators. The table is the effective SOS equivalent of the RDOS COM.CM file (see 093-000075, Appendix D). The table is intended only for the SOS user employing the RDOS to SOS Interface Program, described in Chapter 1.

The table produced by this program, the translate table used to derive the flag bit settings, and the input buffer for reading the command line must be supplied by the calling program. The SOS user stack (see Save-Restore Program) must also be supplied by the caller.

The command line that is read may be continued by typing a SHIFT N (up arrow) character one position before the carriage return. The RUBOUT key (echoed as a back arrow) causes the preceding character to be deleted from the line. The current line may be deleted by typing SHIFT L (backslash). A facility to output a prompt message before reading the command line also exists in the program.

One command table entry is created for each unique string in the input line. Each of these strings may be modified by multiple alphabetic switches (/a where a is a single letter) and/or one numeric switch (/n where n is a digit 0-9). The slash which designates the switch character may follow the string directly or be separated by one or more blanks. The same applies to succeeding switches modifying the same string. The switch character must directly follow the slash, the next character examined will be considered the start of a new string. Any characters

which follow the switch character are ignored. These features are illustrated by the following equivalent argument strings:

```
$PTR/2    /F      /A
$PTR/A/F/2
$PTR/A      /FINAL     /2
$PTR     /ASCII   /F     /2
```

The layout of the command table built by the program is given in the SOS User Application Parameter Tape, 090-000889. Each unique string scanned in the command line causes a two-word entry to be added to the table. The first word at displacement CTSW contains bit settings which correspond with switches that modified the string. This correspondence is established in part by another user-supplied table. Bits 11-15 of this word are reserved for the following use:

Bits 12-15	-	A four-bit octal number derived from the numeric switch modifying the string. If no numeric switch is specified, no bits are set. Maximum value is octal 11.
Bit 11	-	Set if one or more undefined switches modify the string.

The user-supplied translate table (TRT) establishes the definition of bits 0-10 of this table. This allows eleven user specified alphabetic switches. The address of translate table is one of the arguments passed to the program. The table occupies at most 11 locations which specify the bit position/alphabetic switch correspondence. The following examples illustrate the use of the table:

```
TRT:    "C      ;/C = BIT 0
        "E      ;/E = BIT 1
        "B      ;/B = BIT 2
        "X      ;/X = BIT 3
        "A      ;/A = BIT 4
        "M      ;/M = BIT 5
        "T      ;/T = BIT 6
        "Z      ;/Z = BIT 7
        "H      ;/H = BIT 8
        "P      ;/P = BIT 9
        "S      ;/S = BIT 10
        -1      ;TERMINATES TRANSLATE TABLE
```

The example of a translate table shown on the previous page, defines eleven alphabetic switches - C, E, B, X, A, M, T, Z, H, P, and S - to map respectively to bits 0-10 of the switch word. All other alphabetic switches result in the setting of bit 11 of the switch word.

```
TRT:    "A          ;/A = BIT 0
        "L          ;/L = BIT 1
        "B          ;/B = BIT 2
        "N          ;/N = BIT 3
        -1          ;TERMINATES TABLE
```

The previous translate table defines four alphabetic switches - A, L, B, and N - to map to bits 0-3 of the switch word. All other alphabetic switches result in the setting of bit 11 of the switch word. The utility of the table may be increased by inserting the following equivalences into the program:

```
SWA = SW0       SWB = SW2
SWL = SW1       SWN = SW3
```

Symbols SW0, SW1, SW2, and SW3 are defined in the SOS User Application Parameter Tape.

The following sample sequence of code may then be executed to examine the flag bits in each table entry:

```
        LDA  2, .CT         ;AC2 ⟶ COMMAND TABLE
        LDA  1, MASK        ;AC1 = MASK WORD
LOOK:   LDA  0, CTSW, 2     ;AC0 = SWITCH WORD
        AND  1, 0, SZR      ;SWITCH "A" OR "N" ON?
        JMP  OUT            ;YES
        INC  2, 2           ;NO
        INC  2, 2           ;LOOK AT NEXT ENTRY
        JMP  LOOK
OUT:      .
          .
          .
MASK:   SWA + SWN           ;"A" AND "N" SWITCH BITS
.CT:    CT                  ;ADDRESS OF COMMAND TABLE
```

The calling procedure necessary to invoke the program is the following:

```
JSR  .CTB
arg 1
arg 2
arg 3
arg 4
arg 5
return location
```

The arguments passed are the following:

arg 1 — The byte address of any prompt message to be typed on the $TTO. If this argument is a -1, then no message is typed.

arg 2 — The byte address of the input buffer used to read the command line.

arg 3 — The maximum byte size of the input buffer. If this length is exceeded through use of continued input lines, the read is terminated and reissued beginning with the prompt message.

arg 4 — The beginning address of the command table to be created by the program. The maximum size of this table is determined by the maximum input buffer length. The table may have at most one entry for every word in the input buffer. This is the case in which every other input character is a string separator.

arg 5 — The beginning address of the translate table to be used to interpret the string switches.

The arguments returned are:

1. The new Command Table, formatted as in the above description.

2. The Command Table entry count in AC0.

Accumulators 1 and 2 are unchanged; accumulator 3 contains USP. All channels will be closed. Channels 0 and 1 are used to read and write the command line and must therefore be closed when the program is entered. The SOS user stack must have been supplied and initialized as shown in the preceding section.

A sample program which calls the CTB program and repeats the call if less than three unique strings are input is coded below.

```
        .EXTN  .CTB              ;.CTB MUST BE DECLARED
           :
           :
        LDA  1, CN3              ;LOAD COMPARAND
START:  JSR  @CTB                ;CALL CT BUILDER
        PRMPT*2                  ;BYTE ADDRESS OF PROMPT MESSAGE
        IBUF*2                   ;BYTE ADDRESS OF INPUT BUFFER
        300                      ;MAXIMUM COMMAND LINE LENGTH
        CT                       ;COMMAND TABLE ADDRESS
        STRT                     ;TRANSLATE TABLE ADDRESS
        SUBZ# 1, 0, SNC          ;3 OR MORE TABLE ENTRIES?
        JMP  START               ;NO.  RE-PROMPT
        STA  0, T0, 3            ;YES.  SAVE COUNT IN STACK
           :
           :
    CN3:    3
    CTB:    .CTB
    PRMPT:  .TXT /**<15>/
    IBUF:   .BLK 140
    CT:     .BLK 140
    STRT:   "A                   ;ONLY 4 SWITCHES RECOGNIZED
            "B
            "C
            "D
            -1
```

```
;*****************************************************************
;
; NAME: PARU.SR                              PART NUMBER: 090-000883
;
;
; DESCRIPTION: RDOS USER PARAMETERS
;
;
; REVISION HISTORY:
;
;       REV.            DATE
;
;       00              07/25/72
;       01              12/20/72
;       02              06/15/73
;       03              11/30/73
;
;
; COPYRIGHT (C) DATA GENERAL CORPORATION, 1972, 1973
; ALL RIGHTS RESERVED.
;
; LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION.
;*****************************************************************
```

```
; RDOS REVISION 03 USER PARAMETERS


; DEFINE THE CLI STACK DISPLACEMENTS

.DUSR SSLGT=    -7      ; VARIABLE LENGTH OF CALLING'S FRAME
.DUSR SSOSP=    -6      ; PREVIOUS STACK POINTER
.DUSR SSRTN=    -5      ; RETURN ADDRESS OF CALLING PROGRAM
.DUSR SSEAD=    -4      ; ENTRY ADDRESS OF CALLED ROUTINE
.DUSR SSCRY=    -3      ; CARRY
.DUSR SSAC0=    -2      ; SAVE STORAGE FOR CALLING'S ACCUMULATORS
.DUSR SSAC1=    -1
.DUSR SSAC2=    0       ; (DON'T MODIFY THIS DISPLACEMENT!!)
```

```
;
; UFT ENTRY
;

; USER FILE DEFINITION (UFD) OF UFT

.DUSR UFTFN=0              ;FILE NAME
.DUSR UFTEX=5              ;EXTENSION
.DUSR UFTAT=6              ;FILE ATTRIBUTES
.DUSR UFTLK=7              ;LINK ACCESS ATTRIBUTES
.DUSR UFLAD=7              ;LINK ALTERNATE DIRECTORY
.DUSR UFTBK=10             ;NUMBER OF LAST BLOCK IN FILE
.DUSR UFTBC=11             ;NUMBER OF BYTES IN LAST BLOCK
.DUSR UFTAD=12             ;DEVICE ADDRESS OF FIRST BLOCK (0 UNASSIGNED)
.DUSR UFTAC=13             ;YEAR-DAY LAST ACCESSED
.DUSR UFTYD=14             ;YEAR-DAY CREATED
.DUSR UFLAN=14             ;LINK ALIAS NAME
.DUSR UFTHM=15             ;HOUR-MINUTE CREATED
.DUSR UFTP1=16             ;UFD TEMPORARY
.DUSR UFTP2=17             ; "     "
.DUSR UFTUC=20             ;USER COUNT
.DUSR UFTDL=21             ;DCT LINK

; DEVICE CONTROL BLOCK (DCB) OF UFT

.DUSR UFTDC=22             ;DCT ADDRESS
.DUSR UFTUN=23             ;UNIT NUMBER
.DUSR UFTCA=24             ;CURRENT BLOCK ADDRESS
.DUSR UFTCB=25             ;CURRENT BLOCK NUMBER
.DUSR UFTST=26             ;FILE STATUS
.DUSR UFTEA=27             ;ENTRY'S BLOCK ADDRESS
.DUSR UFTNA=30             ;NEXT BLOCK ADDRESS
.DUSR UFTLA=31             ;LAST BLOCK ADDRESS
.DUSR UFTDR=32             ;SYS.DR DCB ADDRESS
.DUSR UFTFA=33             ;FIRST ADDRESS

; DCB EXTENSION

.DUSR UFTBN=34             ;CURRENT FILE BLOCK NUMBER
.DUSR UFTBP=35             ;CURRENT FILE BLOCK BYTE POINTER
.DUSR UFTCH=36             ;DEVICE CHARACTERISTICS
.DUSR UFTCN=37             ;ACTIVE REQ COUNT
                           ;B0 INDICATES Q, 0=DSQ1,1=DSQ2

.DUSR UFTEL=UFTCN-UFTFN+1        ;UFT ENTRY LENGTH
.DUSR UFDEL=UFTDL-UFTFN+1        ;UFD ENTRY LENGTH

.DUSR UDBAT=UFTAT-UFTDC ;NEGATIVE DISP. TO ATTRIBUTES
.DUSR UDBAD=UFTAD-UFTDC ;NEGAVIE DISP. TO FIRST ADDRESS
.DUSR UDBBK=UFTBK-UFTDC ;NEGATIVE DISP. TO LAST BLOCK
.DUSR UDBBN=UFTBN-UFTDC ;POSITIVE DISP. TO CURRENT BLOCK
```

```
;
; FILE ATTRIBUTES
;

.DUSR ATRP =180          ;READ PROTECTED
.DUSR ATCHA=181          ;CHANGE ATTRIBUTE PROTECTED
.DUSR ATSAV=182          ;SAVED FILE
.DUSR ATNRS=187          ;CANNOT BE A RESOLUTION ENTRY
.DUSR ATUS1=189          ;USER ATTRIBUTE # 1
.DUSR ATUS2=1810         ;USER ATTRIBUTE # 2
.DUSR ATPER=1814         ;PERMANENT FILE
.DUSR ATWP =1815         ;WRITE PROTECTED

;
; FILE CHARACTERISTICS
;

.DUSR ATLNK=183          ;LINK ENTRY
.DUSR ATPAR=184          ;PARTITION ENTRY
.DUSR ATDIR=185          ;DIRECTORY ENTRY
.DUSR ATRES=186          ;LINK RESOLUTION (TEMPORARY)
.DUSR ATDIO=188          ;DIRECT I/O ONLY
.DUSR ATCON=1812         ;CONTIGUOUS FILE
.DUSR ATRAN=1813         ;RANDOM FILE


;
; DEFINE SWITCHES
;

.DUSR   A.Sw=   1800
.DUSR   B.Sw=   1801
.DUSR   C.Sw=   1802
.DUSR   D.Sw=   1803
.DUSR   E.Sw=   1804
.DUSR   F.Sw=   1805
.DUSR   G.Sw=   1806
.DUSR   H.Sw=   1807
.DUSR   I.Sw=   1808
.DUSR   J.Sw=   1809
.DUSR   K.Sw=   1810
.DUSR   L.Sw=   1811
.DUSR   M.Sw=   1812
.DUSR   N.Sw=   1813
.DUSR   O.Sw=   1814
.DUSR   P.Sw=   1815
.DUSR   Q.Sw=   1800
.DUSR   R.Sw=   1801
.DUSR   S.Sw=   1802
.DUSR   T.Sw=   1803
.DUSR   U.Sw=   1804
.DUSR   V.Sw=   1805
.DUSR   W.Sw=   1806
.DUSR   X.Sw=   1807
.DUSR   Y.Sw=   1808
.DUSR   Z.Sw=   1809
```

```
;
; SYSTEM CONSTANTS.
;

.DUSR SCWPB=255.          ;WORDS PER BLOCK
.DUSR SCDBS=256.          ;SIZE OF DISK BLOCK
.DUSR SCLLG=132.          ;MAX LINE LENGTH
.DUSR SCAMX=24.           ;MAX ARGUMENT LENGTH IN BYTES
.DUSR SCFNL=UFTEX-UFTFN+1 ;FILE NAME LENGTH
.DUSR SCMER=10.           ;MAX ERROR RETRY COUNT
.DUSR SCSTR=16            ;SAVE FILE STARTING ADDRESS
.DUSR SCTIM=-80.          ;RINGIO 1 MS. LOOP TIME (SN)
.DUSR SCNSO=56            ;NUMBER OF SYSTEM OVERLAYS
.DUSR SCPPL=0             ;PRIMARY PARTITON LEVEL
.DUSR SCPPA=6             ;PRIMARY PARTITION BASE ADDRESS
.DUSR SCSYS=0             ;SYS.DR ADDRESS OFFSET
.DUSR SCPNM=4             ;MAX NUMBER OF PUSH LEVELS
.DUSR SCBPB=SCSYS+1       ;RELATIVE BACKROUND PUSH BASE
.DUSR SCFPB=SCBPB+SCPNM   ;RELATIVE FOREGROUND PUSH BASE
.DUSR SCMAP=SCPNM*2+1     ;RELATIVE BASE ADDRESS OF MAP.DR
.DUSR SCEXT=UFTEX-UFTFN   ;EXTENSION OFFSET IN NAME AREA
.DUSR SCRRL=64.           ;WORDS PER RANDOM RECORD
.DUSR SFINT=100           ;INTERRUPT FLAG
.DUSR SFBRK=1815          ;BREAK FLAG


; DEFINE SYSTEM BOOTSTRAP CONSTANTS

.DUSR    SCTBP=0          ; TEXT STRING BYTE POINTER
.DUSR    SCFUL=SCTBP+1    ; FULL INIT INDIRECT
.DUSR    SCPSA=2          ; PROGRAM START ADDRESS INDIRECT
.DUSR    SCPAR=SCFUL+1    ; PARTIAL INIT ADDRESS INDIRECT
.DUSR    SCPOV=SCPAR+1    ; PARTIAL WITH OVERLAYS INDIRECT
.DUSR    SCCLI=SCPOV+1    ; ADDRESS OF END OF CLI CONSTANT
.DUSR    SCZMX=SCCLI+1    ; SQUASHED/UNSQUASHED FLAG
.DUSR    SCCPL=SCZMX+1    ; CURRENT PARTITION LEVEL
.DUSR    SCPBA=SCCPL+1    ; PARTITION BASE ADDRESS
.DUSR    SCOFA=SCPBA+1    ; OVERLAY FILE BASE ADDRESS
.DUSR    SCKEY=SCOFA+1    ; MASTER DEVICE KEY
.DUSR    SCUNI=SCKEY+1    ; MASTER DEVICE UNIT NUMBER
.DUSR    SCBAS=SCUNI+1    ; BASE OF INFORMATION BLOCK
.DUSR    SCIDV=20         ; INITIAL DEVICE CODE
```

```
; DEFINE THE SYSTEM ERROR CODES

.DUSR ERFNO=    0       ; ILLEGAL CHANNEL NUMBER
.DUSR ERFNM=    1       ; ILLEGAL FILE NAME
.DUSR ERICM=    2       ; ILLEGAL SYSTEM COMMAND
.DUSR ERICD=    3       ; ILLEGAL COMMAND FOR DEVICE
.DUSR ERSV1=    4       ; NOT A SAVED FILE
.DUSR ERWR0=    5       ; ATTEMPT TO WRITE AN EXISTENT FILE
.DUSR EREOF=    6       ; END OF FILE
.DUSR ERRPR=    7       ; ATTEMPT TO READ A READ PROTECTED FILE
.DUSR ERWPR=    10      ; WRITE PROTECTED FILE
.DUSR ERCRE=    11      ; ATTEMPT TO CREATE AN EXISTENT FILE
.DUSR ERDLE=    12      ; A NON-EXISTENT FILE
.DUSR ERDE1=    13      ; ATTEMPT TO ALTER A PERMANENT FILE
.DUSR ERCHA=    14      ; ATTRIBUTES PROTECTED
.DUSR ERFOP=    15      ; FILE NOT OPENED
.DUSR ERUFT=    21      ; ATTEMPT TO USE A UFT ALREADY IN USE
.DUSR ERLLI=    22      ; LINE LIMIT EXCEEDED 0
.DUSR ERRTN=    23      ; ATTEMPT TO RESTORE A NON-EXISTENT IMAGE
.DUSR ERPAR=    24      ; PARITY ERROR ON READ LINE
.DUSR ERCM3=    25      ; TRYING TO PUSH TOO MANY LEVELS
.DUSR ERMEM=    26      ; NOT ENUF MEMORY AVAILABLE
.DUSR ERSPC=    27      ; OUT OF FILE SPACE
.DUSR ERFIL=    30      ; FILE READ ERROR
.DUSR ERSEL=    31      ; UNIT NOT PROPERLY SELECTED
.DUSR ERADR=    32      ; ILLEGAL STARTING ADDRESS
.DUSR ERRO=     33      ; ATTEMPT TO READ INTO SYSTEM AREA
.DUSR ERDIO=    34      ; FILE ACCESSIBLE BY DIRECT I/O ONLY
.DUSR ERDIR=    35      ; FILES SPECIFIED ON DIFF. DIRECTORIES
.DUSR ERDNM=    36      ; DEVICE NOT IN SYSTEM
.DUSR EROVN=    37      ; ILLEGAL OVERLAY NUMBER
.DUSR EROVA=    40      ; FILE NOT ACCESSIBLE BY DIRECT I/O
.DUSR ERTIM=    41      ; USER SET TIME ERROR
.DUSR ERNOT=    42      ; OUT OF TCB'S
.DUSR ERXMT=    43      ; SIGNAL TO BUSY ADDR
.DUSR ERSOF=    44      ; FILE ALREADY SQUASHED ERROR
.DUSR ERIBS=    45      ; DEVICE ALREADY IN SYSTEM
.DUSR ERICB=    46      ; INSUFFICENT CONTIGUOUS BLOCKS
.DUSR ERSIN=    47      ; QTY ERROR
.DUSR ERQTS=    50      ; ERROR IN USER TASK QUEUE TABLE
.DUSR ERNMD=    51      ; NO MORE DCB'S
.DUSR ERIDS=    52      ; ILLEGAL DIRECTORY SPECIFIER
.DUSR ERDSN=    53      ; DIRECTORY SPECIFIER NOT KNOWN
.DUSR ERD2S=    54      ; DIRECTORY IS TOO SMALL
.DUSR ERDDE=    55      ; DIRECTORY DEPTH EXCEEDED
.DUSR ERDIU=    56      ; DIRECTORY IN USE
.DUSR ERLDE=    57      ; LINK DEPTH EXCEEDED
.DUSR ERFIU=    60      ; FILE IS IN USE
.DUSR ERTID=    61      ; TASK ID ERROR
.DUSR ERCMS=    62      ; COMMON SIZE ERROR
.DUSR ERCUS=    63      ; COMMON USAGE ERROR
.DUSR ERSCP=    64      ; FILE POSITION ERROR
.DUSR ERDCH=    65      ; INSUFFICIENT ROOM IN DATA CHANNEL MAP
```

```
.DUSR ERDNI=    66      ; DIRECTORY NOT INITIALIZED
.DUSR ERNDD=    67      ; NO DEFAULT DIRECTORY
.DUSR ERFGE=    70      ; FOREGROUND ALREADYS EXISTS
.DUSR ERMPT=    71      ; ERROR IN PARTITON SET
.DUSR ERDPD=    72      ; DIRECTORY IN USE BY OTHER PROGRAM
.DUSR ERUSZ=    73      ; NO ROOM FOR UFTS ON EXEC/EXFG
.DUSR ERMPR=    74      ; ADDR ERROR ON .SYSTM PARAM
.DUSR ERNLE=    75      ; NOT A LINK ENTRY
.DUSR ERNTE=    76      ; CURRENT BG IS NOT CHECKPOINTABLE
.DUSR ERSDE=    77      ; SYS.DR ERROR
.DUSR ERMDE=    100     ; MAP.DR ERROR
.DUSR ERDTO=    101     ; DEVICE TIME OUT
.DUSR ERENA=    102     ; ENTRY NOT ACCESSIBLE VIA LINK
.DUSR ERMCA=    103     ; MCA REQUEST OUTSTANDING
.DUSR ERSPR=    104     ; INCOMPLETE TRANSMISSION CAUSED BY RECIEVER
.DUSR ERSDL=    105     ; SYSTEM DEADLOCK
.DUSR ERCLO=    106     ; I/O TERMINATED BY CHANNEL CLOSE
.DUSR ERSFA=    107     ; SPOOL FILE(S) ACTIVE
.DUSR ERABT=    110     ; TASK NOT FOUND FOR ABORT
```

```
; DEFINE THE CLI ERROR CODES

.DUSR CNEAR=300          ;NOT ENOUGH ARGUMENTS
.DUSR CILAT=301          ;ILLEGAL ATTRIBUTE
.DUSR CNDBD=302          ;NO DEBUG ADDRESS
.DUSR CNCTD=303          ;NO CONTINUATION ADDRESS
.DUSR CNSAD=304          ;NO STARTING ADDRESS
.DUSR CCKER=305          ;CHECKSUM ERROR
.DUSR CNSFS=306          ;NO SOURCE FILE SPECIFIED
.DUSR CNACM=307          ;NOT A COMMAND
.DUSR CILBK=310          ;ILLEGAL BLOCK TYPE
.DUSR CSPER=311          ;NO FILES MATCH SPECIFIER
.DUSR CPHER=312          ;PHASE ERROR
.DUSR CTMAR=313          ;TOO MANY ARGUMENTS
.DUSR CTMAD=314          ;TOO MANY ACTIVE DEVICES
.DUSR CILNA=315          ;ILLEGAL NUMERIC ARGUMENT
.DUSR CSFUE=316          ;FATAL SYSTEM UTILITY ERROR
.DUSR CILAR=317          ;ILLEGAL ARGUMENT
.DUSR CCANT=320          ;IMPROPER OR MALICIOUS INPUT

.DUSR CCMAX=CCANT                ;MAX CLI ERROR CODE
.DUSR ERML=20.           ;MAXIMUM ERROR MESSAGE LENGTH
; DEFINE THE PANICS

.DUSR    PNMPE=  @1      ; MAP.DR ERROR
.DUSR    PNSDE=  @2      ; SYSTEM DIRECTORY ERROR
.DUSR    PNCSO=  @3      ; SYSTEM STACK OVERFLOW
.DUSR    PNIDA=  @4      ; INCONSISTENT SYSTEM DATA
.DUSR    PNMDD=  @5      ; MASTER DEVICE DATA ERROR
.DUSR    PNMDT=  @6      ; MASTER DEVICE TIME OUT
.DUSR    PNDPE=  @7      ; MOVING HEAD DISK ERROR
.DUSR    PNCUI= @10      ; UNCLEARABLE UNDEFINED INTERRUPT
```

```
        ; DEFINE THE DEVICE CHARACTERISTICS

.DUSR   DCCPD=  1815    ; DEVICE REQUIRING LEADER/TRAILER
.DUSR   DCSTO=  1815    ; USER SPECIFIED TIME OUT CONSTANT (MCA)
.DUSR   DCCGN=  1814    ; GRAPHICAL OUTPUT DEVICE WITHOUT TABBING
                        ; HARDWARE
.DUSR   DCIDI=  1813    ; INPUT DEVICE REQUIRING OPERATOR INTERVENTION
.DUSR   DCCNF=  1812    ; OUTPUT DEVICE WITHOUT FORM FEED HARDWARE
.DUSR   DCTO=   1811    ; TELETYPE OUTPUT DEVICE
.DUSR   DCKEY=  1810    ; KEYBOARD DEVICE
.DUSR   DCNAF=  1809    ; OUTPUT DEVICE REQUIRING NULLS AFTER FORM FEEDS
.DUSR   DCRAT=  1808    ; RUBOUTS AFTER TABS REQUIRED
.DUSR   DCPCK=  1807    ; DEVICE REQUIRING PARITY CHECK
.DUSR   DCLAC=  1806    ; REQUIRES LINE FEEDS AFTER CARRIAGE RTN
.DUSR   DCSPO=  1805    ; SPOOLABLE DEVICE
.DUSR   DCFWD=  1804    ; FULL WORD DEVICE (ANYTHING GREATER THAN
.DUSR   DCFFO=  1803    ; FORM FEEDS ON OPEN
.DUSR   DCLTU=  1802    ; CHANGE LOWER CASE ASCII TO UPPER
.DUSR   DCC80=  1801    ; READ 80 COLUMS
.DUSR   DCDIO=  1800    ; SUSPEND PROTOCOL ON TRANSMIT (MCA)
.DUSR   DCSPC=  1800    ; SPOOL CONTROL
                        ; SET = SPOOLING ENABLED
                        ; RESET = SPOOLING DISABLED
```

```
; USER STATUS TABLE (UST) TEMPLATE

.DUSR   UST=    400     ; START OF BACKGROUND USER STATUS AREA

.DUSR   USTP=12         ;PZERO LOC FOR UST POINTER
; NOTE- USTP MUST CORRESPOND TO PARS PZERO ALLOCATIONS

.DUSR   USTPC=  0
.DUSR   USTZM=  1       ; ZMAX
.DUSR   USTSS=  2       ; START OF SYMBOL TABLE
.DUSR   USTES=  3       ; END OF SYMBOL TABLE
.DUSR   USTNM=  4       ; NMAX
.DUSR   USTSA=  5       ; STARTING ADDRESS
.DUSR   USTDA=  6       ; DEBUGGER ADDRESS
.DUSR   USTHU=  7       ; HIGHEST ADDRESS USED
.DUSR   USTCS=  10      ; FORTRAN COMMON AREA SIZE
.DUSR   USTIT=  11      ; INTERRUPT ADDRESS
.DUSR   USTBR=  12      ; BREAK ADDRESS
.DUSR   USTCH=  13      ; # TASKS (LEFT), # CHANS (RIGHT)
.DUSR   USTCT=  14      ; CURRENTLY ACTIVE TCB
.DUSR   USTAC=  15      ; START OF ACTIVE TCB CHAIN
.DUSR   USTFC=  16      ; START OF FREE TCB CHAIN
.DUSR   USTIN=  17      ; INITIAL START OF NREL
.DUSR   USTOD=  20      ; OVLY DIRECTORY ADDR
.DUSR   USTSV=  21      ; FORTRAN STATE VARIABLE SAVE ROUTINE (OR 0)
.DUSR   USTEN=  USTSV   ; LAST ENTRY

.DUSR   UFPT=   30      ; SAVE SOS

;
; DEFINE TASK CONTROL BLOCK (TCB) TEMPLATE
;


.DUSR TPC=      0       ;USER PC + CARRY
.DUSR TAC0=     1       ;AC0
.DUSR TAC1=     2       ;AC1
.DUSR TAC2=     3       ;AC2
.DUSR TAC3=     4       ;AC3
.DUSR TPRST=    5       ;STATUS BITS (LEFT BYTE) + PRIORITY (RIGHT BYTE)
.DUSR TSYS=     6       ;SYSTEM CALL WORD
.DUSR TLNK=     7       ;LINK WORD
.DUSR TUSP=     10      ;USP
.DUSR TELN=     11      ;TCB EXTENTION ADDR
.DUSR TID=      12      ;TASK ID
.DUSR TTMP=     13      ;SCHEDULER TEMPORARY

.DUSR TLN=TTMP-TPC+1    ;LENGTH OF TCB

; DEFINE TASK STATUS BITS
;
; 180 = TASK PENDED
; 181 = WAITING FOR OVERLAY AREA OR .XMTW/.REC
; 182 = SUSPENDED
; 183 = WAITING FOR .TRDOP (READ OPERATOR INPUT)
```

```
;
; DEFINE OVERLAY NODE TABLE
;


.DUSR   OVNDS=0         ;DIRECTORY NODE TABLE START
.DUSR   OVRES=1         ;CURRENT OVLY+USER COUNT
.DUSR   OVDIS=2         ;NUMBER OF OVERLAYS (LEFT BYTE)
                        ;SIZE IN BLOCKS (RIGHT BYTE)
.DUSR   OVBLK=3         ;STARTING BLOCK
.DUSR   OVNAD=4         ;CORE ADDR FOR NODE

; OFFSETS FOR USER TASK QUEUE TABLE

.DUSR   QPC=    0       ;STARTING PC
.DUSR   QNUM=   1       ;NUMBER OF TIMES TO EXEC
.DUSR   QTOV=   2       ;OVERLAY
.DUSR   QSH=    3       ;STARTING HOUR
.DUSR   QSMS=   4       ;STARTING SEC IN HOUR
.DUSR   QPRI=   TPRST   ;MUST BE SAME
.DUSR   QRR=    6       ;RERUN TIME INC IN SEC
.DUSR   QTLNK=  TLNK    ;MUST BE SAME
.DUSR   QOCH=   10      ;CHAN OVERLAYS OPEN ON
.DUSR   QCOND=  11      ;TYPE OF LOAD
.DUSR   QTLN=   QCOND-QPC+1




        .EOT
```

```
; COPYRIGHT (C) DATA GENERAL CORPORATION, 1971, 1973
; ALL RIGHTS RESERVED.


;
;       SOS PARAMETERS
;

;               LINKAGE

.DUSR   SAVE=   JSR     @3
.DUSR   RTRN=   JMP     @4
.DUSR   RTLOC=  0
.DUSR   AC0=    1
.DUSR   AC1=    2
.DUSR   TMP=    3
.DUSR   SLGT=   TMP+1
.DUSR   OAC0=   AC0-SLGT
.DUSR   OAC1=   AC1-SLGT
.DUSR   OTMP=   TMP-SLGT
.DUSR   ORTN=   RTLOC-SLGT
.DUSR   NFRAM=  6.
.DUSR   SSZ=    NFRAM*SLGT

;               PAGE ZERO

.DUSR   RLOC=   6
.DUSR   CMSK=   7
.DUSR   CSP=    10
.DUSR   CDCT=   11      ;IN SERVICE DCT
.DUSR   BDCT=   13      ;BEGINNING OF DCT CHAIN
.DUSR   CAC0=   14
.DUSR   CAC1=   15


;               ADDITIONAL UST DEFINITIONS

.DUSR   USTA0=  20
.DUSR   USTA1=  USTA0+1
.DUSR   USTA2=  USTA1+1
.DUSR   USTA3=  USTA2+1
.DUSR   USTCY=  USTA3+1
.DUSR   USTIS=  USTCY+1
.DUSR   USTWA=  USTIS+1
.DUSR   USTRS=  USTWA+1


;               ADDITIONAL DEVICE CHARACTERISTICS

.DUSR   DCDIR=  180     ; SOS DATA CHANNEL DEVICE
```

```
; DEVICE CONTROL TABLE (DCT) LAYOUT

; COMMON TO ALL DEVICES

.DUSR DCTCD=    0       ; DEVICE CODE
.DUSR DCTMS=    1       ; MASK OF LOWER PRIORITY DEVICES

        ;DEFINE THE MASK BITS

        .DUSR   MSTTO=  1B15
        .DUSR   MSTTI=  1B14
        .DUSR   MSPTP=  1B13
        .DUSR   MSLPT=  1B12
        .DUSR   MSCDR=  1B10
        .DUSR   MSPLT=  1B12
        .DUSR   MSMTA=  1B10
        .DUSR   MSPTR=  1B11

.DUSR DCTCH=    2       ; DEVICE CHARACTERISTICS
.DUSR DCTLK=    3       ; LINK TO NEXT DCT
                        ; (-1 TERMNATES THE CHAIN)
.DUSR DCTIS=    4       ; INTERRUPT SERVICE ROUTINE ADDRESS
.DUSR DCTIL=    5       ; INTERRUPT MACHINE STATE LINK


.DUSR DCTDT=    6       ; COMMAND DISPATCH TABLE ADDRESS WORD
.DUSR DCTST=    7       ; ADDRESS OF DEVICE START ROUTINE
.DUSR DCTSP=    10      ; ADDRESS OF DEVICE STOP ROUTINE
.DUSR DCTFL=    11      ; FLAGS (ACTIVE, ATTACHED, ETC.)

        ; DEFINE THE FLAGS
        .DUSR   DCACT=  1B15    ; ACTIVE FLAG
        .DUSR   DCACPT= 1B8     ; ACCEPT CHARACTER FLAG
        .DUSR   DCKMD=  1B0     ; TTY KEYBOARD MODE FLAG
```

```
; COMMON TO DEDICATED DEVICES (I.E. SINGLE USER/SINGLE BUFFER)

.DUSR DCTBS=    12      ; BUFFER SIZE ( BYTES )
.DUSR DCTBF=    13      ; BUFFER FIRST ADDRESS (BYTE )
.DUSR DCTBL=    14      ; BUFFER LAST  ADDRESS
.DUSR DCTIP=    15      ; BUFFER INPUT POINTER (BYTE )
.DUSR DCTOP=    16      ; BUFFER OUTPU POINTER
.DUSR DCTCN=    17      ; COUNT OF ACTIVE DATA
.DUSR DCTTO=    20      ; TIMEOUT WORD (ALL INPUT DEVICES)
.DUSR DCTCC=    20      ; COLUMN COUTER (ALLOUTPUT DEVICES)
.DUSR DCTRC=    21      ; RESTART CONSTANT (ALL INPUT DEVICES)
.DUSR DCTLC=    21      ; LINE COUNTER (ALL OUTPUT DEVICES)
.DUSR DCTAT=    22      ;DEVICE ATTRIBUTES
.DUSR DCTFC=    23      ;DEVICE FIXED CHARACTERISTICS

.DUSR LCHNO=    6       ; LOWEST LEGAL CHANNEL #
.DUSR HCHNO=    37      ; HIGHEST LEGAL CHANNEL #
                        ; NOTE - ONE OR BOTH OF THESE EQUI-
                        ;        VALENCES MAY BE CHANGED TO ADD
                        ;        DEVICE DRIVERS

; MAG TAPE PARAMETERS

.DUSR   MTBWZ=  377     ;BUFFER WORD SIZE
.DUSR   MTBBZ=  MTBWZ*2 ;BUFFER BYTE SIZE
.DUSR   CTBWZ=  MTBWZ
.DUSR   CTBBZ=  MTBBZ


;               INTERRUPT FRAME TEMPLATE

.DUSR   IAC0=   DCTFC+1
.DUSR   IAC1=   IAC0+1
.DUSR   IAC2=   IAC1+1
.DUSR   IAC3=   IAC2+1
.DUSR   IPC=    IAC3+1
.DUSR   IRLOC=  IPC+1
.DUSR   IMSK=   IRLOC+1
.DUSR   IFRL=   7       ;INTERRUPT FRAME LENGTH

        .EOT
```

```
; COPYRIGHT (C) DATA GENERAL CORPORATION, 1972, 1973
; ALL RIGHTS RESERVED.


;SOS USER APPLICATION PARAMETERS (PARUA)

;STACK DISPLACEMENTS

.DUSR   SSEL=   6               ;ENTRY LENGTH
.DUSR   SOSEC=  5               ;ENTRY COUNT FOR SOS USER ROUTINES
.DUSR   RTR=    0               ;FRAME LAYOUT:
.DUSR   T0=     1               ;  RETURN LOCATION
.DUSR   T1=     2               ;  TEMPORARIES
.DUSR   SAC0=   3               ;  SAVE ACCUMULATORS
.DUSR   SAC1=   4
.DUSR   SAC2=   5
.DUSR   OAC0=   SAC0-SSEL       ;THESE DISPLACEMENTS PERMIT
.DUSR   OAC1=   SAC1-SSEL       ; "CALLEE" TO GET AT "CALLER'S"
.DUSR   OAC2=   SAC2-SSEL       ;  REGISTERS
.DUSR   ORTR=   RTR-SSEL
.DUSR   OT0=    T0-SSEL
.DUSR   OT1=    T1-SSEL

        ;COMMAND TABLE DISPLACEMENTS

.DUSR   CTBP=   0               ;STRING BYTE POINTER
.DUSR   CTSW=   1               ;SWITCH WORD
.DUSR   CTNBP=  2               ;NEXT STARTING BYTE POINTER
.DUSR   CTEL=   2               ;ENTRY LENGTH

        ;COMMAND TABLE SWITCHES

.DUSR   SW0=    1B0             ;  EACH OF THESE BIT SETTINGS
.DUSR   SW1=    1B1             ;MAY BE EQUIVALENCED TO A MEAN-
.DUSR   SW2=    1B2             ;INGFUL VALUE IN THE USER APPLI-
.DUSR   SW3=    1B3             ;CATION PROGRAM.  THE ARRANGEMENT
.DUSR   SW4=    1B4             ;OF THE TRANSLATE TABLE (TRT)
.DUSR   SW5=    1B5             ;DETERMINES THE PRECISE MEANING
.DUSR   SW6=    1B6             ;OF EACH SWITCH.
.DUSR   SW7=    1B7
.DUSR   SW8=    1B8
.DUSR   SW9=    1B9
.DUSR   SW10=   1B10
.DUSR   SW11=   1B11

        .EOT
```

△ (space) 3-8
) (carriage return or form feed) 3-7, 3-8
↑ (carriage return mask) 3-8
; (command delimiter) 3-8
\ (SHIFT L) 3-8
, (argument separator) 3-8
/ (switch indicator) 3-8
← (RUBOUT) 3-8
(, , ,) in command line 3-13
(form feed) 3-8
* editor prompt 3-42
# CILW prompt 3-5

.ACHR A-31
adding device handlers App. A
APPEND 3-18
ASM 3-42
assembler
command 3-42
configuring 5-7
on master reel 5-12

BOOT 3-19, 1-3
bootstrapping CILW 3-4
BPUNCH 3-20
break (CTRL A, CTRL C) 3-2
byte
load A-33
pointer 4-4
store A-32

card reader 2-1, 4-27
cassette
configuring utilities for 5-8
file protect tab 2-4
files 2-4
physical characteristics 2-4
response to SOS 4-28
tape library 1-1, 2-3
tapes supplied for 5-1 to 5-3
$CDR 2-1, 4-27
CDRDR (.CDRD) 2-3
channel number to device map A-17
channel numbers
list of fixed for standard SOS devices 2-1
range 1-2
character
getting (.GCHAR) 4-17
putting (.PCHAR) 4-17
reading (.RCHR) A-30
sending (.ACHR) A-31
CILW (see core image loader/writer)
CLI (see command line interpreter)
.CLOSE 4-10
.CLS routine A-28
command delimiter (;) 3-8
command line
argument separators 3-8
character erase (←) 3-8
comma parentheses convention 3-13
deletion (\) 3-8
definition and usage 3-9 ff
long _ _ 3-10
stacking commands on 3-10
switches
command line interpreter
activation 3-7
command descriptions 3-18 ff
definition 1-3, 3-7
list of commands 3-17
messages from 3-14 ff
on master reel 5-10
prompt (R) 3-7
symbols and conventions 3-8
command table builder 2-3, App. B
commands (CLI)
APPEND 3-18
BOOT 3-19
BPUNCH 3-20
COPY 3-21
DEB 3-22
DUMP 3-23
FILCOM 3-25
GTOD 3-26
INIT 3-27
LOAD 3-28
MKASS 3-30
MKSAVE 3-31
PRINT 3-32
PUNCH 3-33
RELEASE 3-34
SDAY 3-35
STOD 3-36
TYPE 3-37
XFER 3-38
commands (system)
.CLOSE 4-10
.DELAY 4-21
.EOPEN 4-8
.GCHAR 4-17
.GCHN 4-9
.GDAY 4-20
.GHRZ 4-22
.GTATR 4-5
.GTOD 4-20
.IDEF 4-22
.IRMV 4-23
.MEM 4-18
.MEMI 4-19
.OPEN 4-8
.PCHAR 4-17
.RDL 4-10
.RDS 4-12

commands (system) (continued)
.RESET 4-10
.ROPEN 4-8
.SDAY 4-21
.STOD 4-20
.SYSI 4-4
.SYSTM 4-1
.WRL 4-15
.WRS 4-16
communicating with SOS 1-3
configuring SOS Chapt. 5
configuring for cassette/mag tape 5-8
configuring for paper tape 5-6
configuring utilities
assembler 5-7
FORTRAN IV 5-14
other utilities 5-6
core image loader/writer
bootstrap 3-4
configuring 5-6
definition 3-4
error conditions 3-5, 3-6
installation 3-4
on master reel 5-9
operation 3-5 ff
prompt (#) 3-5
versions 3-4
core
available to user 1-5, 1-6
determining available 4-18
requirements 1-1
CTA (cassette) 4-28
CTADR (.CTAD) 2-3
CTB (.CTB) 2-3, 5-5, App. B
CTRL A 1-3, 3-2
CTRL C 1-3, 3-2
CTRL L 3-7
CTU1-CTU7 (.CTU1-.CTU7) 2-3
CT0-CT7 2-1

DCT A-10
DEB 3-22
.DELAY 4-21
device
definition 2-1
list of 2-1
device control table (DCT) A-10, A-3
device dispatch routines A-24
device handler
adding App. A
approaches, table of A-8
device start routine A-23
device stop routine A-23
.DISM routine A-33
DUMP 3-23

EOF on file 2-4
.EOPEN 4-8
error messages (CLI) 3-15
external (.EXTN) declarations
list of device/program names 2-3
need for 2-2
FILCOM 3-25
file
attributes (.GTATR) 4-5
closing (.CLOSE, .RESET) 4-10
definition 2-1
of utilities 3-2
opening (.OPEN) 4-8
filename table A-18 ff
FORT 3-51
FORTRAN IV
command 3-51
configuring 5-14

.GCHAR 4-17
.GCHN 1-2, 4-9
.GDAY 4-20
.GHRZ 4-22
global subroutines A-27
.ACHR A-31
.CLS A-28
.DISM A-33
.IBUF A-31
.IDCT A-34
.LDB A-33
.OBUF A-32
.OPN A-30
.RCHR A-30
.RDLI A-30
.RDSE A-29
.STB A-32
.SYSE A-34
.WRLI A-29
.WRSE A-28
.GTATR 4-5
GTOD 3-26
.GTOD 4-20

HMA 1-6

.IBUF routine A-31
.IDCT routine A-33
.IDEF 4-22
INIT 3-27
interrupt
considerations in adding device App. A
handling A-22
mask A-2
search list A-14
types of 1-3, 3-2

I/O messages (CLI) 3-14
.IRMV 4-23

.LDB routine A-33
level one devices A-1
level two devices A-1
LFE 3-45
library file editor
    command 3-45
    configuring 5-6, 5-13
library tapes 1-1
line
    reading (.RDL) 4-10
    writing (.WRL) 4-15
line printer 2-1, 4-28
linkage A-5, A-7, A-24ff
LOAD 3-28
loaded program in core 1-5, 1-6
loader (see relocatable loader or core image loader)
loading library files 1-2, 2-2
$LPT 2-1, 4-28
LPTDR (.LPTD) 2-3
LP132 2-3

magnetic tape
    configuring utilities for 5-8
    files 2-4
    library 1-1
    physical characteristics 2-4
    response to SOS 4-29
    tapes supplied for 5-1, 5-2, 5-4
    write permit ring 2-4
master reel of utilities 5-8
.MEM 4-18
.MEMI 4-19
memory available to user 1-5, 1-6
MKABS 3-30
MKSAVE 3-31
MTA (mag tape) 4-29
MTADR (.MTAD) 2-3
MTU1-MTU7 (.MTU1-.MTU7) 2-3
MT0-MT7 2-1

NMAX
    changing 4-19
    definition 1-6

.OBUF routine A-32
.OPEN 1-2, 4-8
.OPN routine A-27
.OPPP A-14, A-15

paper tape
    configuring utilities for 5-6
    tapes supplied for 5-1, 5-2
    paper tape punch 2-1, 4-29
    paper tape reader 2-1, 4-29
    parameters (RDOS, SOS) App. C
    PARA.SR App. C
    PARU.SR App. C
    PARUA.SR App. C
.PCHAR 4-17
    plotter 2-1, 4-26
    $PLT 2-1, 4-26
    PLTDR 2-3
    PRINT 3-32
    $PTP 2-1, 4-29PTPD
    PTPDR 2-3
    $PTR 2-1, 4-28
    PTRDR (.PTRD) 2-3
    PUNCH 3-33

    R (ready) message 1-3, 3-7
    .RCHR routine A-30
    .RDL 4-10
    .RDLI routine A-30
    RDOS-compatible SOS 1-2
    RDOS to SOS interface 1-2
    .RDS 4-12
    RDSI 1-2
    .RDSE A-29
    .RDSI 1-2
    real time clock drivers 2-3
    RELEASE 3-34
    relocatable loader
        command 3-39
        configuring 5-6, 5-9
    repetitive argument storage 3-13
    .RESET 4-10
    RLDR 3-39
    .ROPEN 4-8
    RTCT1-RTCT5 (.RTC1-.RTC5) 2-3
    RUBOUT 3-8

    save-restore program App. B
    SAVRE (.SAVR) 2-3 App. B
    SDAY 3-35
    .SDAY 4-21
    sequential
        read (.RDS) 4-12
        write (.WRS) 4-16
    SETRG 5-5
    SHIFT L 3-8
    SOS
        library 1-1, 2-3
        main program (SOS-MAIN) 2-3
        organization in core 1-5, 1-6
        RDOS-compatible 1-2
        (.SOS) 1-2, 2-3
        standard 1-2
        system generation 1-4, Chapt. 5
        user interface 1-3

stack frame A-25
standard SOS
    definition 1-2
    list of devices/channels 2-1
.STB routine A-32
STOD 3-36
.STOD 4-20
STTYDR (.STTY) 2-3
switch
    alphabetic 3-12
    global 3-12
    indicator 3-8
    local 3-12
    numeric 3-11
.SYSE routine A-34
SYSG 3-49, 5-5
SYSGEN
    command 3-49
    configuring 5-6, 5-14
    used to produce trigger 5-5
    utility 2-2
.SYSI 4-4
system calls 1-3, Chapt. 4
.SYSTM 4-1

tapes 1-1
    supplied to all 5-1
    supplied to cassette users 5-2, 5-3
    supplied to mag tape users 5-2, 5-4
    supplied to paper tape users 5-2

teletypewriter 2-1
    TTI1 2-1, 4-26
    $TTI 2-1, 4-27
    TTO1 2-1, 4-26
    $TTO 2-1, 4-27
    $TTR 2-1, 4-28
text editor
    command 3-41
    configuring 5-6, 5-11
trigger
    definition 2-2
    producting a 2-2, 5-5
$TTI 2-1, 4-27
TTI1 2-1, 4-26
$TTO 2-1, 4-27
TTO1 2-1, 4-26
$TTP 2-1, 4-26
TTY1 (.TTI1) 2-3
TYPI 3-37

user application routines App. B
user status table 4-30

utility program
    assembler 3-42, 5-7, 5-12
    command line interpreter 3-7 ff, 5-6, 5-10
    core image loader/writer 3-4ff, 5-6, 5-9
    definition 1-3
    FORTRAN IV 3-51, 5-14
    library file editor 3-45, 5-6, 5-13
    operation, general 3-1, 3-2
    relocatable loader 3-39, 5-1, 5-9
    SYSGEN 3-49, 2-2, 5-5, 5-6, 5-14
    text editor 3-41, 5-6, 5-11

.WRL 4-15
.WRLI routine A-29
.WRS 4-16
.WRSE routine A-28

XFER 3-38

DataGeneral

PROGRAMMING DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code

Form No. 10-24-004

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary If Mailed In The United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Programming Documentation

FOLD UP SECOND FOLD UP

STAPLE