Reference Manual
SEL 810B
General Purpose Computer

November, 1968

This publication supersedes SEL 95118D,
SEL 810B General Purpose Computer
Reference Manual, dated April, 1968.

6901 West Sunrise Blvd., Fort Lauderdale, Florida 33310
Area Code 305 587-2900

301-095118-004
Price: $10.00

LIST OF EFFECTIVE PAGES

The total number of pages in this manual is 116, consisting of the following:

Page Number	Issue	Page Number	Issue
Title	Original		
A	Original		
i thru vi	Original		
1-1 thru 1-10	Original		
2-1 thru 2-26	Original		
3-1 thru 3-12	Original		
4-1 thru 4-6	Original		
5-1 thru 5-8	Original		
6-1 thru 6-20	Original		
7-1 and 7-2	Original		
A-1 and A-2	Original		
B-1 thru B-4	Original		
C-1 and C-2	Original		
D-1 and D-2	Original		
E-1 and E-2	Original		
F-1 thru F-10	Original		
G-1 and G-2	Original		

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF TABLES (Cont'd)

LIST OF RELATED PUBLICATIONS

The following publications contain information not included in this manual but necessary for a complete understanding of the 810B Computer System.

Publication Title	Publication No.
Technical Manual 810B General Purpose Computer	303-095019-000
Reference Manual 810A/810B Assembler	323-095052-001
Technical Manual 810A/810B Loader Program	322-095055-001
Technical Manual 810/840 Computer Series Library Subroutines	322-095057-001
Technical Manual 810A/810B Diagnostic Programs	322-095061-001
Technical Manual 810A/810B FORTRAN IV Compiler	322-095062-001
Technical Manual 810A/810B Operator	302-096064-002
Reference Manual 810A/810B Operating	312-095071-000
Technical Manual 810A/810B Assembler Program	322-095094-001
Drawings Manual 810B General Purpose Computer	304-095116-000
Design Manual 810B Input/Output Interface	310-095117-000

SECTION I
GENERAL DESCRIPTION

INTRODUCTION

The SEL 810B Computer shown in the frontispiece, is a fast, general-purpose, 16-bit binary computer. The low cost, speed, and highly flexible input/output structure of this computer make it especially well suited to real-time data collection, processing, and control applications. The 810B computers are designed to meet field requirements such as the following:

Industrial process control (Direct Digital Control)

Factory test automation

Missile and aircraft test data collection

Data logging and display

Real-time and post-test data processing

Telemetry data processing and simulation

Flight simulation

In addition to the basic computer, the SEL 810B system consists of a large variety of standard peripheral devices, data acquisition and display subsystems, and a comprehensive software package.

GENERAL CHARACTERISTICS

810B COMPUTER

All silicon monolithic integrated logic circuits

Sixteen-bit word length plus parity

8192-word memory

750-nanosecond full cycle time

Memory parity bit with parity generator/checker

Fully parallel operation

Computation time including access and indexing:

Add, Subtract	1.5 microseconds
Multiply	4.5 microseconds
Divide	8.25 microseconds

Double-length Accumulator

Hardware index register (lower B-Accumulator)

I/O structure capable of handling 64 peripheral device controllers (drivers and terminators for 16 controllers supplied with the basic computer)

Two separate levels of priority interrupt

Sixteen sense switches

Switch-addressable program halt

Power fail safe

ASR-33 typewriter with paper tape reader and punch mounted on stand beside the computer

Computer size - 24 inches wide, 62 inches high, 30 inches deep (45 inches deep including optional desk top)

Typewriter size - 22 inches wide, 35 inches high, 18 inches deep

Temperature Environment, Operating:

810B Computer (excluding Teletypewriter) - 0° to 55° C (32° to 131° F)

Teletypewriter - 10° to 35° C (50° to 95° F)

COMPUTER OPTIONS

Up to eight automatic block transfer control units capable of transferring up to 1,333,000 words per second

Additional hardware index register

Memory expandable to 32K

Program protect and instruction trap feature for guarding blocks of memory against modification and for preventing execution of privileged instructions

Up to 98 individual levels of priority interrupts

Variable base register-increases direct addressing capability

ASR-35 console typewriter in place of ASR-33

I/O parity checker and generator

Real-time clock

Computer graphics processor

Stall alarm

Auto start

STANDARD SOFTWARE

Full ASA FORTRAN compiler - operates in 8K memory

FORTRAN library

Assembler - relocatable object format, Macro capability, and extensive set of pseudo-operations

Compiler/assembler loader

Utility routines - debugging aids, I/O handlers, tape editor

Maintenance routines - complete set for computer and peripheral units

PERIPHERAL DEVICES

Card reader - 200 and 400 card/minute

Card punch - 100 cards/minute

Paper tape reader (photoelectric) - 300 characters/second

Paper tape punch - 110 characters/second

Magnetic tape control unit - handles up to eight tape units

Magnetic tape units - 45, 75, 120, 150 inches/second; 200, 556, 800 characters/inch; 7 and 9 track

Movable head disc file - 1.5 million words storage, 150 ms maximum track access time (track 00 to track 99)

Fixed head disc files - up to 909K 16-bit word storage, 8.3 ms average access time

Typewriters - ASR-33, KSR-33, ASR-35, KSR-35, RO-33, RO-35, 10 characters/second

Line printers - 300, 600, 1000 lines/minute, 120 columns/line

Incremental plotters - 12 inch-chart width (300 steps/second) and 31-inch chart width (200 steps/second)

CRT display - 10 x 10-inch display area including vector generator with the following options:

Alphanumeric character generator
Function switches
Light pen
Line texture control

Interval timer

Interface subsystem components

Multiplexer - low-level and high-level, solid-state and relay switching

Sample and hold units

Analog/digital converter - up to 15 bits binary. Word rates to 50K words/second

Digital/analog converter - up to 12 bits binary

Customer interfaces

APPLICATIONS PROGRAMMING

The Systems Engineering Laboratories Programming Group has developed both total and basic sets of applications programs for many 810B systems. Capability and experience exists in the areas of:

Real-time executives - monitor systems

Data collection, corrections, recording and logging

Industrial process control

Time-shared operations

Data display

Data analysis and scientific computation

COMPUTER ORGANIZATION

The SEL 810B Computer is formed by four major units: memory, control, arithmetic and input/output (see figure 1-1). The memory unit stores the instruction words which define the operation of the computer and the data words on which the computer operates. The control unit calls up the instruction words, decodes them and issues commands

to operate the computer. The arithmetic unit performs computation with data words supplied by the input/output unit and the memory unit under the direction of the control unit. The input/output unit transmits data words, commands, and status reports between the computer and peripheral equipment. The computer operates on, and from, 16-bit binary words which are transferred in parallel between the computer units. Arithmetic operations are performed using two's complement binary arithmetic with negative words stored in the two's complement form. The combined control and arithmetic units are often called the mainframe section.

MEMORY UNIT

The memory unit is composed of one, two, three or four separate modules. Each module has 8192 addressable storage locations. Each location consists of one 16-bit data or instruction word plus a parity bit. The total number of storage locations can range from 8192 provided by the basic 8K module to 32,768 available with four 8K modules.

Individual modules are composed of these four elements:

a. 8K x 17-bit Magnetic Core Memory

b. 13-bit Memory Address Register

c. 17-bit Data Register

d. Self-contained Timing and Control

Instruction words and data words are loaded into specific addresses prior to the program execution. Loading may be performed manually through the panel controls or automatically from peripheral units through the use of the supplied loader program. Each input word is transferred to the memory data register and the accompanying storage address is transferred to the memory address register. When both registers have been loaded, a "write" command is issued by the program control unit and the 17 bits in the memory data register are written into the 17 magnetic cores addressed by the memory address register.

When the entire group of instruction words forming a program is loaded and execution is started, addresses selected by the control unit are sent to the memory address register and a "read" command is issued. The state of each core at that address is sensed and transferred to the memory data register. The sensing of the cores sets them all to the same state, so the memory word now in the memory data register is immediately rewritten into its original memory location so as to be available for later use. The word is also transferred to the control unit to be decoded or to

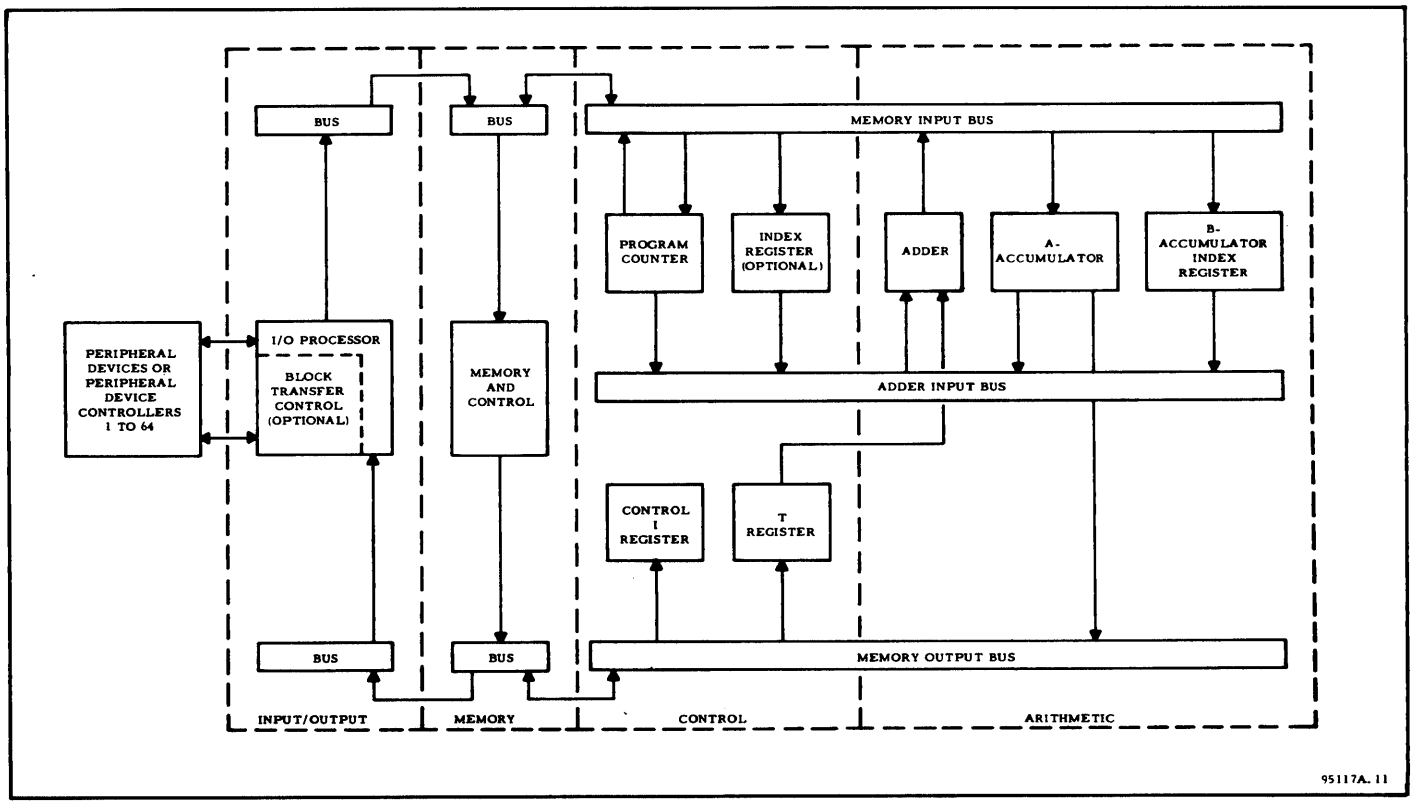


Figure 1-1. SEL 810B Block Diagram

the arithmetic unit for computation. The memory read and write cycles are completely automatic so that only the memory address and source or destination must be supplied by the program.

CONTROL UNIT

The control unit contains a 15-bit binary PROGRAM COUNTER capable of directly addressing 32, 768 memory locations. This counter supplies the addresses of the instruction words from which the computer operates. The counter is initially set to the address minus one of the first instruction of a program when the computer is started. It is then automatically advanced by each instruction until a Halt, Branch or Conditional Skip instruction is read from memory. The Halt instruction stops the computer while the Branch instructions change the contents of the program counter to the operand address contained in the instruction. The Skip instructions cause the program counter to be advanced by either one or two locations, depending on the value of the Skip condition specified by the instructions.

The instruction words are read from memory into the INSTRUCTION REGISTER and automatically restored in memory. The binary digits forming the instruction word are then applied to the OPERATION CONTROL circuits. The unique codes assigned to each instruction are then decoded and used to provide timing and gating signals to the remainder of the machine. The signals from switches on the CONTROL CONSOLE are also connected to the OPERATION CONTROL circuits. External PRIORITY INTERRUPTS will cause the control circuits to switch the program counter to programs designed to process the external demand.

The memory cycle during which instruction words are read and decoded is referred to as the "Instruction Cycle". Some instructions, called memory reference instructions, contain a memory address which specifies the location of an "operand" which is to be operated on by the computer. For these instructions, one or more additional memory cycles, called "Execution Cycles", are required. During the instruction cycle, the memory address is supplied in part by the "operand address" contained in the instruction word and by the program counter. The operand is read from memory and operated upon according to signals provided by the operation code. Most memory reference instructions are accessed and executed in a total of two cycles. However, instructions such as multiply and divide require more than one execution cycle.

Many instruction words require no operand from memory and are executed completely within the instruction cycle. Others, while requiring no operand from memory, do require one or more execution cycles for completion. Chief among this latter group are the shift instructions. For these instructions, a group of bits within the instruction word defines the number of shifts to be performed while the operation code of the word defines the type of shifting to be done. Other instructions, notably the input/output control instructions, are composed of two instruction words; one defining the type of operation and the unit and the other defining the actual operand or the operand memory location. The words forming these input/output instructions are automatically unloaded from memory in the proper sequence.

ARITHMETIC UNIT

The arithmetic unit consists of a 16-bit adder and several accessory storage registers. Two of these registers, the A-ACCUMULATOR and the B-ACCUMULATOR, may be loaded and unloaded by program control. The A-ACCUMULATOR is the primary arithmetic register and derives its name from its function of accumulating results of the arithmetic operations. Because only one word may be taken from the memory and input/output units by each instruction, the second operand in add and subtract operations must be loaded in a register prior to the add and subtract instructions. The A-ACCUMULATOR fulfills this function and also provides temporary storage for the result of the arithmetic operation. The B-ACCUMULATOR holds the multiplier during multiply operations and stores the least significant bits of the product. In addition to these strictly arithmetic functions, the two accumulators provide a convenient storage area for rearranging data words through shifting and logical operations.

A third register connected to the adder is the T-REGISTER which holds the operand unloaded from the memory. This 16-bit register plus the 16-bit A and B-ACCUMULATORS supply inputs to the 16-bit binary ADDER. When an add instruction is performed, the data words are simply added according to the rules of two's complement binary arithmetic.

The basic data format of the 810B computer is a 16-bit binary single-precision fixed point word. (See figure 1-2). This format contains the sign bit in bit position 0, with bit position 1 holding the most significant data bit and bit position 15 holding the least significant bit. Two's complement representation is used for negative numbers. This format is defined as an integer with an imaginary binary point located to the right of bit position 15. The 810B set of library integer subroutines assumes this representation. The programmer can, of course, scale single-precision words in any desired

manner and utilize the extensive shift and test instruction repertoire to maintain the binary point location.

The 810B Computer also accommodates double-precision data words (figure 1-2B) of 30 bits plus a sign through the use of the extended B-Accumulator. Each double-precision data word is normally stored in two adjacent memory locations with the most significant half stored in the first (lower) address. The product generated by a single-precision multiply is located in the A and B-Accumulators in this format. The dividend is assumed to be in this double-precision format prior to the execution of the DIVIDE instruction.

Three floating point data formats are utilized by the 810B Computer library. The single-precision floating point format consists of two words (figure 1-2C). The first word contains the sign and 15 most significant bits of the fractional mantissa; the second word contains the six least significant mantissa bits and the signed 8-bit exponent. The words are stored in adjacent memory locations with the first word located in the lower memory address. Both the mantissa and the exponents carry separate signs so that the mantissa can be positive or negative independent of the sign of the exponent. Two's complement representation is used for negative numbers.

Double-precision floating point format consisting of three memory words is provided for use with the set of double-precision floating point library subroutines (figure 1-2D).

The third floating point data format (complex floating point data) is provided for the set of FORTRAN IV subroutines dealing with complex numbers (figure 1-2E).

The arithmetic unit includes two single-bit registers which are addressable by the program. The first of these is the OVERFLOW latch which can be set during addition, subtraction and division operations. The overflow for an add or subtract occurs when the result exceeds the accumulator capacity. A divide overflow occurs if the divisor is equal to or smaller than the dividend. This latter overflow is due to the fact that the machine treats all divide arguments as double-precision numbers by scaling the single-precision divisor by 2^{15}. If the dividend is larger than the scaled divisor, the quotient will necessarily be a number greater than 2^{15}. Such a number exceeds the capacity of the 15-bit A-Accumulator in which the quotient is to be stored; this produces a false divide. The overflow latch lights the OVERFLOW indicator on the control console and remains set until tests, and reset, by an SOF (skip no overflow) instruction. Because the latch remains set until tested, such a test should be made immediately following an arithmetic process when an overflow condition could result. This prevents the possibility of a second overflow being undetected by the already set latch. The overflow latch can also be set with an OVS instruction.

The second addressable arithmetic latch is the CARRY latch which connects to the least significant bit of the parallel adder. This latch is set in the regular arithmetic processes to produce a two's complement number (one's complement of the number plus one). The latch is used in the addition and subtraction of double-precision numbers formed in the A and B-Accumulators. The least significant words of the double-precision numbers are processed and stored in the B-Accumulator. If a carry or borrow is generated, it will cause the sign of the B-Accumulator to change. A CSB (copy sign of B) instruction is used to set the carry latch to the state of the B-Accumulator sign bit and then reset the B sign bit to zero (as required in the double-precision format). If the operation is addition, the True output of the carry latch is added together with the most significant word; if a subtract operation is in process the False output of the carry latch is added to the most significant word (effectively subtracting the borrow).

The CSB instruction should be followed immediately by the AMA or SMA instruction which operates on the most significant half of the double-precision operand, since the carry latch is cleared at the end of the execution of all instructions except CSB.

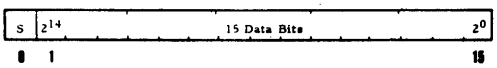
INPUT/OUTPUT UNIT

The basic input/output unit contains an input/output processor that communicates with peripheral device control units over 64 parallel direct information channels. Each device control unit is assigned a unit number that corresponds to the number of the direct information channel that is used as a communication path between the computer and the device control unit. Each device control unit can control or communicate with several peripheral devices; therefore, the number of individual peripheral devices that can communicate with the computer, or be under the control of the computer, is virtually unlimited.

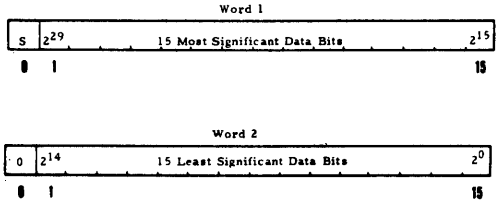
Data transfer instructions are provided that enable word transfers directly between the computer memory (or the A-Accumulator) and the peripheral device, through the device control unit. In addition, external unit commands and test instructions are provided.

The I/O instruction set is particularily powerful because each instruction causes several functions

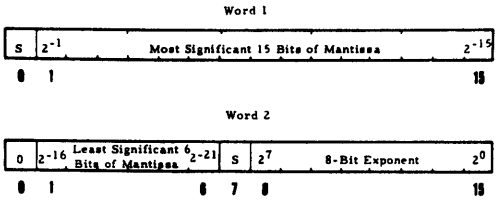
A Single-Precision Fixed Point Data



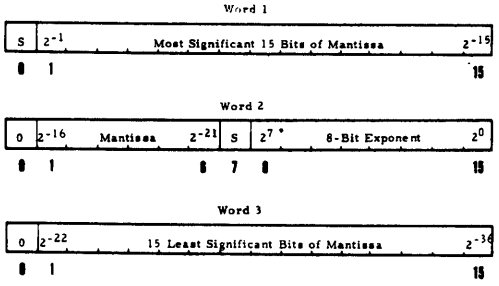
B Double-Precision Fixed Point Data



C Single-Precision Floating Point Data



D Double-Precision Floating Point Data



E Complex Floating Point Data

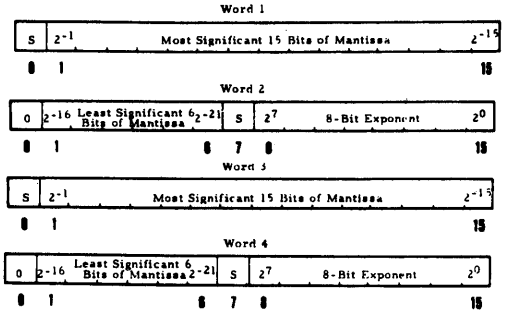


Figure 1-2. 810B Computer Basic Data Formats

to be performed. First, execution of each I/O instruction causes a device to be connected to the computer. The device (unit) number (direct information channel number) is contained in each I/O instruction.

Second, an automatic test is made of the device which determines if the device can execute the instruction. Third, the data or command transfer is made if the device is "ready". Fourth, the device is disconnected. If the device is not ready when tested, the computer will either wait until the device is ready and then transfer, or it will disconnect the device and advance the program counter to a "reject" location. A "Wait Flag" is provided in each I/O instruction, except the test instruction (TEU), to enable the programmer to specify the "Wait" or "Skip" mode of execution. The normal time required to perform the complete connect, test, transfer, and disconnect operation is only four machines cycles plus wait.

In addition to the basic I/O structure, up to eight fully buffered, block transfer units can be added to the computer. These units permit transfer of blocks of data up to 32,768 words in length between the computer and peripheral devices. Block transfer is made under hardware control at rates up to 1,333,000 words per second. A single cycle is stolen per word transfer. An automatic reinitialization feature is provided which enables chaining of block transfers. Also up to six Computer Graphics Processors (CGP) may be added to the SEL 810B Computer. The CGP is similar to the BTC with the exception of its specialized operating characteristics and added control functions. Unlike the BTC, the CGP examines each word from memory and either interprets the word as data or as an instruction.

A priority interrupt system is provided which enables the computer to have up to 98 individual levels of priority interrupt. Programmable interrupts can be selectively enabled and disabled under program control. A unique memory location is assigned to each level.

An ASR-33 typewriter, paper tape punch and reader are supplied with the basic computer. The reader operates at 20 characters per second and the punch and printer operates at 10 characters per second. The ASR-33 can be operated either on-line of off-line. When operating on-line, the input and output unit operate independently, which enables, for example, a paper tape to be read and a separate set of characters to be printed at the same time. Other console typewriters, such as the ASR-35, can be supplied in place of the ASR-33.

SEL 810B SOFTWARE SYSTEM

A comprehensive, fully-integrated, well-documented and completely checked-out program preparation, library, debugging and utility system is supplied with the SEL 810B Computer system.

Specific features of this standard package are described in detail but a briefing in regard to the philosophy behind the software system package design is mentioned in the following paragraphs.

In determining the optimum software package for the type of equipment under consideration, the following factors were deemed to be of prime importance:

a. The large variety of equipment configuration which will be delivered.

b. The type of application which will be programmed for the equipment.

c. The large amount of programming personnel time which will be involved in developing and debugging operational programs.

d. The need to utilize programs and routines which may already exist on other equipment.

e. The quality and completeness of the documentation supplied with the software and library routines.

In order to satisfy these objectives, two basic types of program preparation systems are provided; a symbolic Macro assembler and a full FORTRAN IV compiling system. Depending upon the specific requirements of a specific portion of an operational package, these two programming systems provide the user with an optimum capability where tradeoffs between coding and checkout time and program running time are involved.

The fact that a specific portion of a program can be coded in either language is most significant to the user. The loader will accept both FORTRAN and assembler generated coding in any sequence.

This feature together with the very comprehensive debug package will significantly reduce the coding and checkout time required to produce operational programs.

The FORTRAN IV language specified for this system is the standard ASA FORTRAN IV language; thus, the FORTRAN IV supplied will provide direct compatibility with the majority of other manufacturer supplied FORTRAN IV systems.

In order to satisfy the requirements that all of the supplied software system will operate on a wide variety of computer configurations, especially in the area of peripheral equipment, all of the supplied packages are written in a modular form with a standard program interface specification.

SEL 810B ASSEMBLY PROGRAM

All computer instructions are accepted by the assembler and addresses can be expressed in symbolic, decimal, or octal formats, including address arithmetic with combinations of these.

The following special pseudo-ops are also processed:

BSS	Reserve block of storage name at start
BES	Reserve block of storage name at end
EQU	Define symbolic name
ORG	Set next storage address
ZZZ	Set instruction bits to zero
REL	Set assembly mode to relative
ABS	Set assembly mode to absolute
CALL	Call library subroutine
NAME	Define subroutine name
DATA	Define octal, decimal (fixed or floating) or alphanumeric data
MOR	Pause in assembly process
END	End of program
FORM	Sets bit assignment for "FDAT" pseudo-op
FDAT	Same as "Data" with bits assigned by "Form
DAC	Used to generate direct address constant
EAC	Used to generate extended address constants
NOLS	Used to stop program listing
LIST	Used to continue program listing
MACR	Used to name a Macro
EMAC	Used to terminate a Macro.

A symbolic side-by-side listing complete with error messages is output (operator option) along with the object output tape.

SEL 810B LOADER

The SEL 810B object program loader is designed to be compatible with the FORTRAN IV Compiler and the Assembly program.

The program provides for relocatable and absolute instructions. The capability of using pre-compiled subroutine libraries is included in a manner which allows that a given routine will only be loaded once, regardless of the number of times it is referenced in the program.

The system has been designed with the joint aims of (a) minimizing indirect addressing for those program elements which will operate most frequently; (b) establishing uniform subroutine construction and linkage; (c) relieving the user from over-concern with any complexities introduced by the MAP addressing scheme.

SEL 810B FORTRAN IV

Ease of use was a prime consideration in the design of this compiler. As a result, programmers are free of the restrictions often found in other systems. Convenience features include:

a. One-pass Operation - From source language to machine language object code is a standard feature.

b. No Reserved Identifiers - All names are available for use as identifiers.

c. Optional Tracing - This feature allows selective object code tracing for diagnostic purposes.

d. Optional Mapping - This feature provides a listing of the subprograms required for execution and the names or values and relative location assignments of all variable-array names and constant values used by the program.

e. Optional Chaining - This feature provides for sequential loading and execution of segmented programs.

SEL 810B DEBUG

The debug program is a utility program designed to help a programmer debug a program while it is in memory. The following functions are provided:

a. Type the contents of specified memory in octal or command format.

b. Modify the specified memory; input being in octal format.

c. Dump specified memory areas onto paper tape in a format (non-relocatable) that can be loaded using the loader resident in Debug.

d. Enter breakpoints in order to "leap-frog" trace a program.

e. Clear specified areas of memory to zero.

f. Search memory for references to specified areas.

g. Initiate branches (or Halt and Branch) to any part of memory.

h. Load a binary tape that was dumped using Debug.

Each of these functions are initiated by typing a keyword through the console typewriter keyboard.

SEL 810B UPDATE

Correction of errors in card decks is a relatively easy procedure, consisting of pulling out the bad cards and inserting new cards. However, symbolic source programs on paper tapes or magnetic tapes are not so easily corrected or modified.

The UPDATE program is designed to allow the computer operator to easily correct or modify a symbolic source program tape by providing the following functions:

a. Deletion of a specified line or group of lines.

b. Insertion of a new or replacement line or lines.

All references to the symbolic source tape are made by referring to a sequence number. This number is present on all assembly listings.

SEL 810B LIBRARY PACKAGE

The SEL 810B library package includes the complete set of ASA FORTRAN subroutines in the following categories:

Single-Precision Floating Point Functions

Double-Precision Floating Point Functions

Complex Floating Point Functions

Integer Functions

Input/Output Functions

Control Functions

SEL 810B MAINTENANCE ROUTINES

The SEL 810B Checkout Program is a complete package designed to give the operator the ability to exercise the memory, the mainframe logic, the input/output channels and associated peripheral equipment.

The memory exerciser routine generates various types of worst case bit patterns and exercises the memory with these patterns while monitoring for errors. Provisions are made for automatic relocating of the exerciser program to allow the entire memory to be included in all tests. Also included are certain branch/skip instructions which are sequenced and executed through each location in the memory.

The mainframe exerciser routine executes the entire instruction repertoire individually in a large variety of sequence while monitoring the results for errors. Errors are indicated by halts. Pertinent information concerning the instruction that failed and the nature of the failure can be obtained from the A and B-Accumulator displays, the program counter and certain selected memory locations.

The programs for the I/O channels and associated peripheral equipment test the ability of the various I/O units to generate or receive all acceptable characters. A selected input is used and visual monitoring of the control panel or output unit is required by the operator for verification of proper operation. Equipment tested includes standard Teletypewriter output, input, punch and reader as well as optional card punch, card reader, line printer, high-speed paper tape equipment, magnetic tape units and other units as needed for a particular application.

POWER FAIL SAFE

The power fail safe feature provides an "override" interrupt to allow program storage of the contents of all data registers in the event that power drops below 80 volts. This standard feature assures that no information will be destroyed when power is disrupted. The program can be conveniently resumed after power is restored either manually or automatically by means of the optional Auto Start feature.

SECTION II
MACHINE LANGUAGE PROGRAMMING

INTRODUCTION

The 810B Computer is operated by a series of instruction words stored in the magnetic core memory. The instruction words are successively read from memory locations addressed by the program counter. Each word specifies one operation; transferring a data word from an input unit to a memory location, adding a memory word to the word in the A-Accumulator, shifting the contents of the A-Accumulator, etc. The program counter is normally advanced one count after each instruction to access the instruction word located in the next sequential memory address. The program counter may be preset to any count by Branch/Skip instructions, which detect certain conditions such as A-Accumulator sign positive, overflow condition, input word ready, etc. The program counter then continues its sequential advance, but starts from the new address until again preset. The branch may be to either a higher or lower count so that portions of a program may be repeated until the branch condition is no longer present.

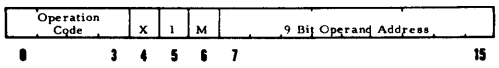
A list of instructions is provided for the 810B Computer that includes Load/Store instructions which transfer words between the memory and the accumulators, Arithmetic instructions, Shift instructions which allow moving of the bits within words, Logical instructions (AND, OR, NEGATE, etc.), Control instructions (HALT, etc.), Branch/Skip instructions to provide program modification and Input/Output instructions to command peripheral devices and transfer data into and out of the computer.

Each instruction word is formed by 16 bits, each of which performs a particular function; defining the operation to be performed, addressing a memory location, defining the number of shifts, etc. The function of a particular bit will vary in different types of instructions. For example, in some words, bit 14 forms part of a memory address; in others, bit 14 forms part of the operation code. The function of the bits depends on the instruction word type defined by the four-bit operation code located in bits 0 - 3 of the first word of each instruction.

There are two types of instruction words used by the SEL 810B; those containing memory addresses within the instruction word and those containing additional code bits in lieu of the address bits.

MEMORY REFERENCE INSTRUCTIONS

The memory reference instructions access the magnetic core memory for an operand. These words contain a four-bit binary operation code, a nine-bit partial memory address and three address modifiers.



Memory Reference Instruction Word

The four-bit binary operation codes for the memory reference instructions vary from 0001 (01_8) to 1110 (16_8), not including 1011 (13_8). Codes 00_8, 13_8 and 17_8 are reserved for augmented instructions (described in later paragraphs). The 13-memory reference instructions contain a nine-bit operand address field (m) that may be coded to obtain 512 unique locations. The memory, whether it is formed by a single memory module or by several modules, is divided into 512-address memory address partitions (MAPs) for addressing purposes. Each MAP extends from memory address $XX000_8$ to $XX777_8$, where XX consists of the six most significant address bits defining the MAP address. In a memory with 8,192 addresses, there are 16 MAPs beginning with MAP 00_8 and extending through MAP 17_8. In a maximum memory of 32,768 addresses, the MAP designations range from 00_8 to 77_8 with the addresses arranged in the following sequence:

MAP 00_8	00000_8 - 00777_8
MAP 01_8	01000_8 - 01777_8
MAP 02_8	02000_8 - 02777_8
.	. .
.	. .
.	. .
MAP 77_8	77000_8 - 77777_8

The state of the MAP designator bit (shown as M in the word format diagram) determines the MAP that will contain the operand address. If the MAP designator bit is a ZERO, the operand address will be in MAP 00_8. If the MAP designator bit is a ONE, the operand address will be in the MAP containing

the instruction word currently being executed. The MAP address of the instruction is supplied by the program counter which advances sequentially across the imaginary MAP boundaries. The program count ranges from 00000_8 in MAP 00_8 to 77777_8 in MAP 77_8 thus including all possible MAP designations. The upper six bits (two octal digits) of the program counter can therefore, add the necessary MAP designation to the nine-bit (three octal digits) operand address to provide a complete 15-bit memory address. The addressing of MAP 00_8 requires only that zeros be put in the upper six bit positions of the 15-bit address to produce addresses 00000_8 to 00777_8 from the basic nine-bit operand address.

The significance of the MAP 00_8 address lies in the fact that these addresses are directly addressable by all instructions irrespective of the MAP location of those instructions. This allows the storage of constants, input/output locations, subroutines, etc., to be stored in this common address area where they can be directly accessed by any portion of the program.

The index flag, shown as X in bit position 4 of the memory reference instruction word format diagram (figure 2-1), is set to one to cause the 15-bit concatenated MAP and operand addresses to be added to the contents of the current index register. The current index register can be either the B-Accumulator, or the optional hardware index register, depending on the condition of the index pointer. The index count can be any 15-bit binary number ranging from 00000_8 to 77777_8. The addition of this number to the concatenated address allows the addressing of any memory location within the full-size 32,768-address memory. If, for example, the instruction being executed is in MAP 12_8, and the index count is 02022_8, the 9-bit operand address is 724_8 and both the MAP designator and index flag are ones; the resulting effective address is $12724_8 + 02022_8$ (X) = 14746_8. If the MAP designator were a zero and the index flag a one, the resulting effective address is $00724_8 + 02022_8$ (X) = 02746_8.

The B-Accumulator when used as an index register serves another important function in that the register can be incremented by one with an instruction. The optional hardware index register can be incremented by any quantity from zero to fifteen. The incrementing instructions also test the register for negative signs and generate a skip (an extra advance count) to the program counter if the register is not negative. This feature allows the programmer to load a basic negative number into the B-Accumulator or the hardware index register, append the index flag to the instruction and create an iterative subroutine that will access a series of sequential memory addresses. Such a subroutine or "loop" using the B-Accumulator as the index register and written in assembly language is shown in table 2-1. In assembly language a "1" is used to indicate an indexed instruction, an apostrophe (') indicates an octal number and either absolute (220) or symbolic (LOOP, input) addresses can be used. A complete description of the assembly language is presented in Section III.

This series of indexed instructions beginning with location LOOP serves to add 20 pairs of numbers and to store the resulting sums in 20 memory locations. The routine assumes that the index pointer has been set to the B-Accumulator. The first pair of numbers is taken from locations 200

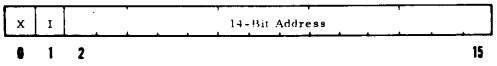
Table 2-1. Sample Listing

Location	Operation	Address	Comments
INDX	LBA	=-20	Load an index count of -20 in the B-Accumulator.
LOOP	LAA	220,1	Load the A-Accumulator with data word from location (220 + index count).
	AMA	320,1	Add to the contents of the A-Accumulator to the contents of location (320 + index count).
	STA	420,1	Store the sum from the A-Accumulator in location (420 + index count).
	IBS		Increment B-Accumulator, test for index count of zero, skip next instruction if zero.
	BRU	LOOP	Take next instruction from location LOOP.
PROG	LAA	INPUT	(Next instruction after index loop.)

added to the index count by the IBS instruction. The resulting -19 count does not equal zero, so the next instruction is executed. This instruction is an unconditional branch instruction which sets the program counter to location LOOP. The next two arguments are taken from locations 201 and 301 and stored in location 401 and the cycle is repeated.

After adding 20 sets of numbers, the final IBS instruction reduces the index count to 00. This causes the next instruction (BRU to LOOP) to be skipped. The program counter now calls a new set of instructions from memory beginning with location PROG.

The third address modification flag contained in all memory reference instructions is the Indirect Address flag (bit 5.) When present, this bit causes the address, contained in the instruction, as modified by the index and MAP bits, to be interpreted as the location in which the operand address is contained, rather than as the location of the operand itself.



Indirect Address Word

The indirect address word (shown above) contains 14 address bits which are merged with the most significant bit from the program counter. The indirect address may be in the same memory half (of 16K) as the program counter.

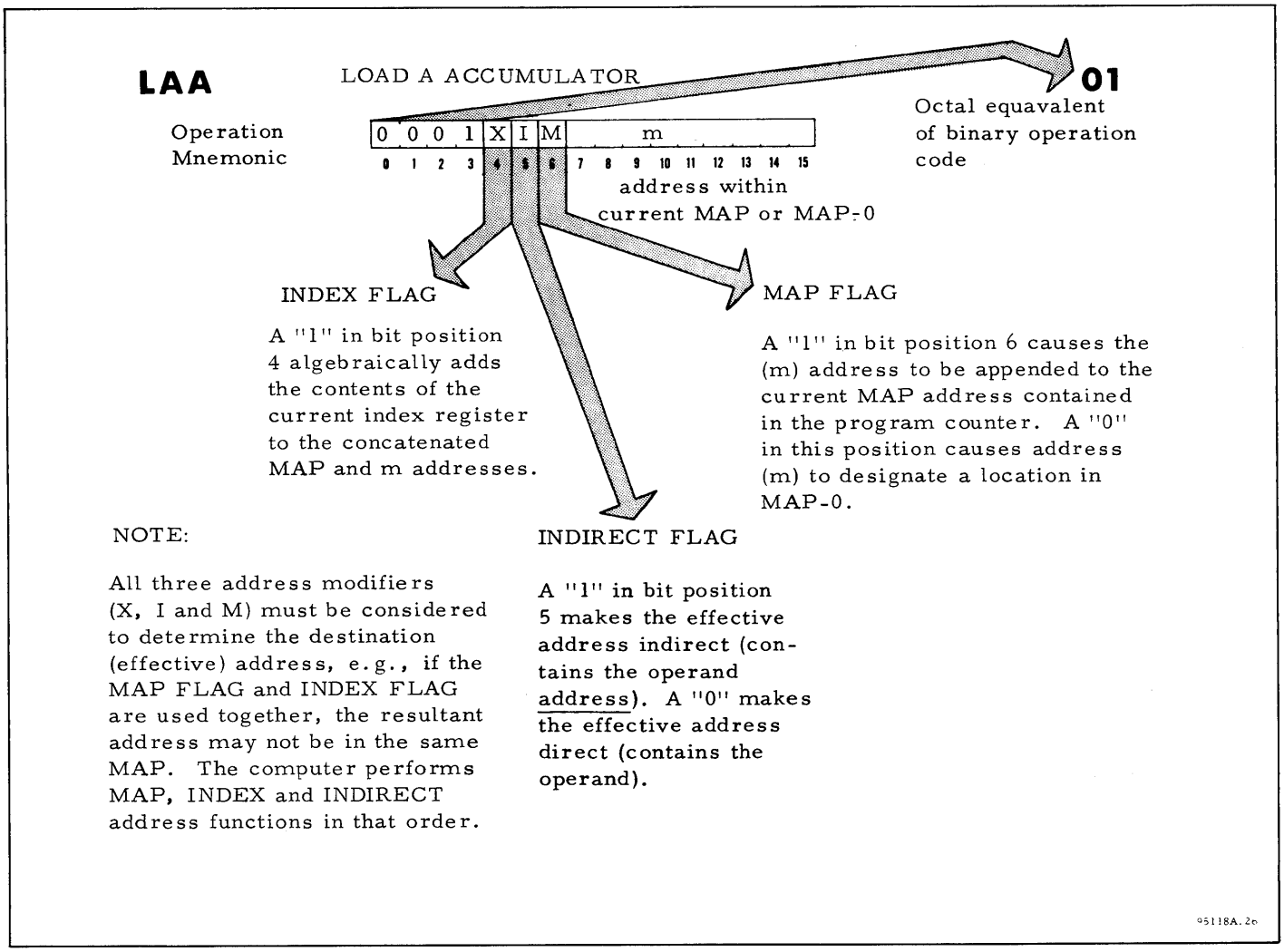


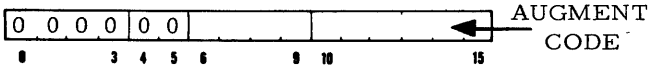
Figure 2-1. Typical Memory Reference Instruction Word Format Diagram

The indirect word format also contains an index flag which, if a one, adds the index count to the indirect address. The index count may be added to the address in the instruction word, and/or the indirect address depending on the presence or absence of an index bit in the instruction and indirect words. The indirect address also includes an indirect flag bit permitting multi-level indirect addressing.

Memory reference descriptions consist of the three-letter mnemonic and a two-character octal operation code. The permissible address modifiers are also shown. An example for a memory reference instruction (LAA) is shown on figure 2-1.

AUGMENTED INSTRUCTIONS

Augmented instructions contain no memory address bits in the first word but do contain additional (augmenting) operation code bits. The augmented instructions have operation codes of 00,13 or 00,17.



Augmented 00 Instruction Word

The detection of the 00 operation code in the instruction register gates the six augment code bits into a special decoding matrix.

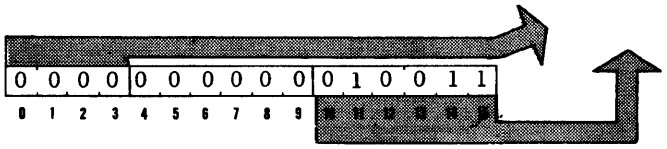
The other augmented operation codes, the 13_8 and the 17_8 codes, are also augmented with additional code bits. These instructions have word formats that vary slightly and some include two words to complete the instruction. Two-word instructions are stored in sequential memory locations with the second word called automatically by the machine. If the indirect flag in the first word is a zero, the second word is interpreted as the operand itself. If the indirect flag is a one, the second word is coded in the indirect address word format and is interpreted as the address of the operand. If the MAP bit is a one, the most significant bit of the program counter becomes the 15th bit of the indirect address when the indirect flag is used; if the MAP bit is a zero, the 15th bit of the indirect address is set to zero.

The augmented 13_8 operation code words are used for disabling and enabling interrupts, testing the condition of the sense switches and testing and commanding of external I/O units. The augmented 17_8 operation code words are used for Input/Output instructions only.

Augmented instruction octal codes consist of a two-digit operation code 00, 13, or 17 followed by a hyphen and one or two-digits showing the augmenting code. For example:

SAN **00-23**

SKIP IF A-ACCUMULATOR IS NEGATIVE



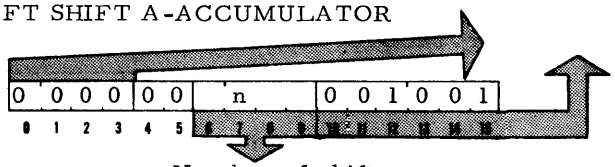
NOTE

All Augmented 00 instructions contain the augment code in bit positions 10-15.

Shift instruction words use bit position 6 through 9 to hold the number of shifts to be performed by the instruction. For example:

LSA -1 **00-11**

LEFT SHIFT A-ACCUMULATOR



Number of shifts to be performed (in binary code)

Input/Output Instructions contain both 13_8 and 17_8 operation codes. The augmenting code bits for these instructions appear as shown in figure 2-2. Bits 10 through 15 always contain the peripheral unit number in binary code.

In the IMMEDIATE MODE, the second instruction word is treated as the operand. In executing MOP, CEU and TEU instructions, the contents of the instruction's second word are transferred to the specified unit. MIP execution consists of transferring a word or character from a specified unit into the instruction's second word location.

In the ADDRESS MODE, the instruction's second word is interpreted as the operand address. The indirect address format is used in the instruction's second word. Therefore, indexing and indirect chaining may be used in addressing the operand.

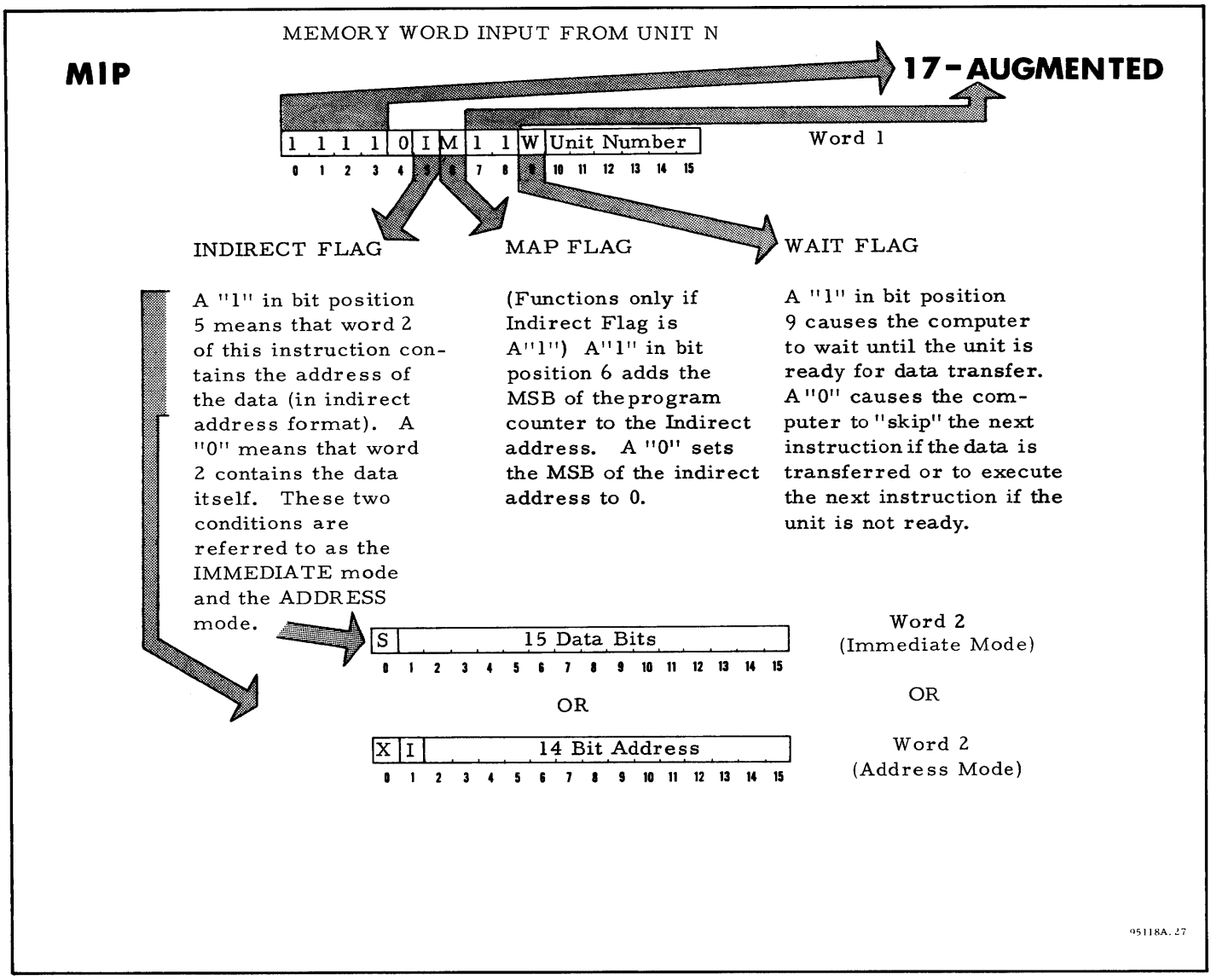


Figure 2-2. Input/Output Instruction Word Format Diagram

The addressing mode is specified in the instructions first word by the value of the Indirect Address Flag (I). If I is a ONE, the Address Mode is executed. In this mode the most significant program counter bit is appended to the most significant end of the 14-bit indirect address if the MAP Flag is a ONE, and ZERO is appended to the most significant end of the indirect address if the MAP Flag is a ZERO.

MACHINE LANGUAGE INSTRUCTION SET

The instruction words causing the various SEL 810B machine operations are described in detail on the following pages. The descriptions include the operation-mnemonic and octal machine code in bold type. The binary word format shows bit assignments for operation code, augment code, operand address and flags (MAP, INDEX, INDIRECT, WAIT, etc.). A brief explanation of the functions, register(s) affected, memory cycles required, indicators (if any) and special notes complete the description.

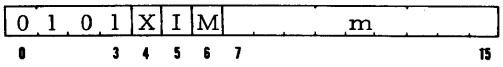
ARITHMETIC INSTRUCTIONS

All arithmetic functions of the computer are performed by this group of seven instructions. The AMA (add) instruction calls a word from memory and adds it to the word previously loaded into the A-Accumulator. The memory word called in the SMA (subtract) instruction is two's complemented and added to the A, Accumulator word. The MPY (multiply) instruction repeatedly adds the memory and the A-Accumulator words according to the value of the word in the B-Accumulator. The DIV

(divide) instruction repeatedly subtracts (adds the complement of) the memory word from the double-length word in the A- and B-Accumulators. The results of these operations are stored in the A-Accumulator (sum, difference, quotient and most significant half of the product) and the B-Accumulator (remainder and least significant half of the product). The augmented 00$_8$ RNA (Round A-Accumulator) instruction is used to round-off the most significant half of the product in the A-Accumulator according to the value of the least significant half of the product in the B-Accumulator. The AMB (add) instruction calls a word from memory and adds it to the word previously loaded into the B-Accumulator. The OVS (set overflow) instruction causes the overflow latch to be set.

AMA 05

ADD MEMORY TO A-ACCUMULATOR



The contents of the effective memory address (addend) are algebraically added to the contents of the A-Accumulator (augend). The sum replaces the previous contents of the A-Accumulator with the sign of the A-Accumulator set to the algebraic sign of the sum.

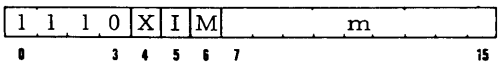
NOTE

The augment must be located in the A-Accumulator prior to the AMA instruction. This may be accomplished through a preceding LAA instruction or the augend may already be properly located as a result of a prior operation.

Timing:	2 cycles
Indicators:	OVERFLOW if the sum exceeds 15 bits plus sign
Registers Affected:	A-Accumulator

AMB 16

ADD MEMORY TO B-ACCUMULATOR



The contents of the effective memory address (addend) are algebraically added to the contents of the B-Accumulator (augend). The sum replaces the previous contents of the B-Accumulator with the sign of the B-Accumulator set to the algebraic sign of the sum.

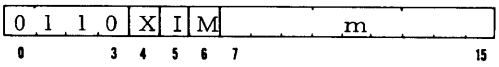
NOTE

The augend must be located in the B-Accumulator prior to the AMB instruction. This may be accomplished through preceding LBA instruction or the augend may already be properly located as a result of a prior operation.

Timing:	2 cycles
Indicators:	OVERFLOW if the sum exceeds 15-bits plus sign
Registers Affected:	B-Accumulator

SMA 06

SUBTRACT MEMORY FROM A-ACCUMULATOR



The contents of the effective memory address (subtrahend) are algebraically subtracted from the contents of the A-Accumulator (minuend). The difference replaces the previous contents of the A-Accumulator and the sign of the A-Accumulator is set to the sign of the algebraic difference.

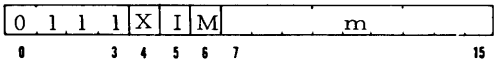
NOTE

The minuend must be located in the A-Accumulator prior to the SMA instruction. This may already be properly located as a result of a prior operation.

Timing:	2 cycles
Indicators:	OVERFLOW if the algebraic difference exceeds 15-bits plus sign
Registers Affected:	A-Accumulator

MPY 07

MULTIPLY



The contents of the effective memory address (multiplicand) are multiplied by the contents of the B-Accumulator (multiplier). The most significant half of the product replaces the previous contents of the A-Accumulator. The least significant half of the product replaces the previous contents of the B-Accumulator. The sign of the A-Accumulator is determined by the algebraic sign of the product; the sign of the B-Accumulator is set to plus. The contents of the memory are unchanged.

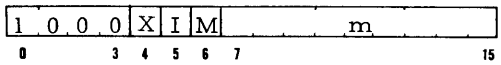
NOTE

If the multiplier and the multiplicand are considered to be integers (binary point to the right of bit 15), the product is a double-precision integer (binary point to the right of bit 15 in the B-Accumulator). If the multiplier is scaled left by 2^a and the multiplicand is scaled left by 2^b, then the product is scaled left by 2 a+b.

Timing:	6 cycles
Indicators:	OVERFLOW if both multiplier and multiplicand are equal to minus full scale.
Registers Affected:	A-Accumulator, B-Accumulator

DIV 10

DIVIDE



The contents of the A- and B-Accumulators (double length dividend) are divided by the contents of the effective memory address (single length divisor). The quotient is stored in the A-Accumulator and the remainder is stored in the B-Accumulator. The sign of the quotient is set to the algebraic sum of the divisor and dividend signs. The sign of the remainder is set to the sign of the original dividend. The contents of the memory are unchanged.

NOTE

The dividend is assumed to be a double-precision quantity (30 bits and sign) which is to be divided by a single-precision quantity (15-bits and sign). The result is two single-precision quantities, the quotient and the remainder. If the part of the dividend contained in the A-Accumulator is greater than or equal to the divisor, an "overflow" will result. The quotient of two single-precision quantities is obtained by the instructions shown below.

```
CLA          Clear A-Accumulator
LBA   DVND   Load Dividend into
             B-Accumulator
DIV   DVSR
```

NOTE (Cont'd)

After the divide instruction is executed, the quotient and remainder are stored in the A- and B-Accumulators, respectively. No "overflow" can occur except when the divisor is equal to zero.

Divide scaling is performed by the simple algorithm given below. Considering the binary point of the operands to be located between the sign bit and the most significant bit, that is at B_0 in single-precision and C_0 in double-precision quantities, the binary point in the quotient is determined by the relationship:

C(M) dividend - B(N) divisor = B(M-N) quotient

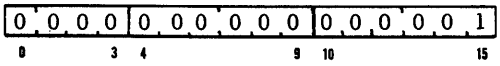
where $-30 \leq M \leq 30$ and $-15 \leq N \leq 15$

If the scale factor is greater than B_{15} an overflow will result.

Timing:	11 cycles
Indicators:	OVERFLOW if the divisor is the portion of the dividend contained in the A-Accumulator
Registers Affected:	A-Accumulator, B-Accumulator

RNA 00-01

ROUND A-ACCUMULATOR

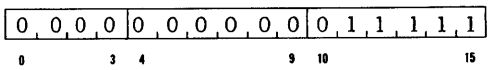


The contents of the A-Accumulator are increased by one if the second most significant bit of the B-Accumulator (B_1) is a one.

Timing:	1 cycle
Indicators:	OVERFLOW if the result in the A-Accumulator exceeds 15-bits.
Registers Affected:	A-Accumulator

OVS 00-37

SET OVERFLOW LATCH



The overflow latch is set by the execution of this instruction.

NOTE

This instruction is used at the exit of interrupt routines to set the overflow latch if it was set when the interrupt occurred.

Timing:	1 cycle
Indicators:	OVERFLOW is set
Registers Affected:	None

LOAD/STORE INSTRUCTIONS

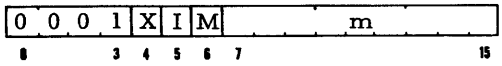
This group of five standard and two optional instructions handles the transfer of data words within the computer. One pair of instructions - LAA (Load A-Accumulator) and STA (Store A-Accumulator) - transfers data between the memory and the A-Accumulator. A second pair - LBA (Load B-Accumulator) and STB (Store B-Accumulator) - communicates between memory and the B-Accumulator. All four words are memory address instructions and, as such, contain MAP, index and indirect address modifiers. The instructions are used primarily to transfer data to the accumulators for use in arithmetic operations and then to store the results of those operations. The B-Accumulator, however, also functions as a hardware index register/counter so that the LBA and STB instructions serve to load and store the index count.

The LCS (Load Control Switches) instruction is an augmented 00_8 word. This instruction is used to transfer the information set into the front panel control switches by the operator to the A-Accumulator. The switches can be used to modify the program in response to external requirements by using the data brought to the accumulator by the LCS instruction to change branch destinations, etc.

The two optional load/store instructions are included as part of the hardware index register option. This pair of instructions is mnemonically labeled LIX and STX. The LIX (Load Index Register) instruction loads data from memory into the hardware index register. The STX (Store Index Register) instruction stores data from the hardware index register into memory. Both of these instructions are two word instructions, with the first word of each an augmented 00_8 word. If the indirect bit (bit 5) of the first word is a zero the second word is the instruction operand (Immediate Mode). If the indirect bit is a one, the second word is the address, in indirect address format, of the operand (Address Mode).

LAA 01

LOAD A-ACCUMULATOR



The contents of the effective memory address replace the previous contents of the A-Accumulator. The contents of the memory are unchanged.

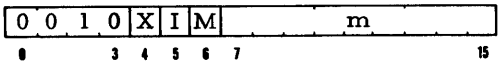
NOTE

The A-Accumulator must be loaded with the augend, minuend and most significant bits of the dividend prior to add, subtract and divide instructions.

Timing:	2 cycles
Indicators:	None
Registers Affected:	A-Accumulator

LBA 02

LOAD B-ACCUMULATOR



The contents of the effective memory address replace the previous contents of the B-Accumulator. The contents of the memory are unchanged.

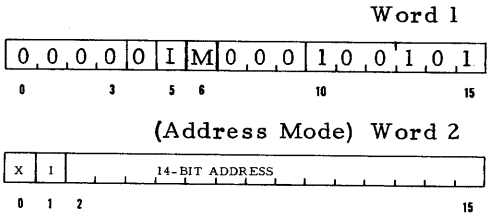
NOTE

This instruction is used to load the index count when the B-Accumulator is to function as the hardware index register. The B-Accumulator must also be loaded with the least significant half of a dividend and the multiplier prior to divide and multiply instructions.

Timing:	2 cycles
Indicators:	None
Registers Affected:	B-Accumulator

LIX 00-45

LOAD INDEX (OPTIONAL)

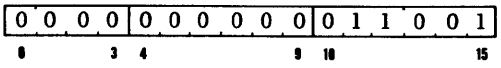


The contents of the effective memory address replace the previous contents of the index register. The contents of the memory are unchanged.

Timing:	2 cycles (Immediate Mode)
Indicators:	None
Registers Affected:	Index Register
Operand Address Mode:	Immediate -I=0, Address-i=1 (in Word 1)

LCS 00-31

LOAD CONTROL SWITCHES

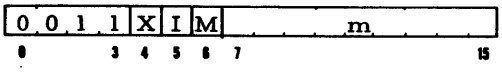


The positions of control panel switches 0-15 are sensed as bits (up-1, center-0) and transferred to the A-Accumulator.

Timing:	1 cycle
Indicators:	None
Registers Affected:	A-Accumulator

STA 03

STORE A-ACCUMULATOR

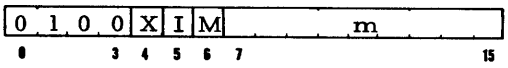


The contents of the A-Accumulator replace the previous contents of the effective memory address. The contents of the A-Accumulator are unchanged.

Timing:	2 cycles
Indicators:	None
Registers Affected:	None

STB 04

STORE B-ACCUMULATOR

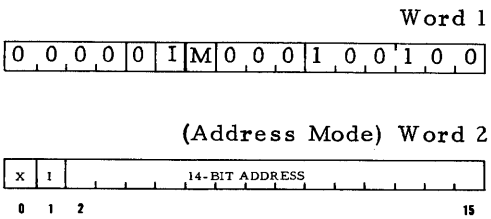


The contents of the B-Accumulator replace the previous contents of the effective memory address. The contents of the B-Accumulator are unchanged.

Timing:	2 cycles
Indicators:	None
Registers Affected:	None

STX 00-44

STORE INDEX (OPTIONAL)



The contents of the index register replace the contents of the effective memory address. The contents of the index register are unchanged.

Timing:	2 cycles (Immediate Mode)
Indicators:	None
Registers Affected:	None
Operand Address Mode:	Immediate -I=0, Address-I=1 (Word 1)

BRANCH/SKIP INSTRUCTIONS

This group of thirteen standard and two optional instructions provides the decision-making capability of the computer. Only four of these instructions - the BRU (Unconditional Branch), SPB (Store Place and Branch), IMS (Increment Memory and Skip) and CMA (Compare Memory to A-Accumulator) are memory reference instructions. The remainder, all skip instructions, are augmented 00_8 instruction words with the single exception of the SNS instruction which is an augmented 13_8 code word.

The three branch instructions, BRU, LOB and SPB, specify a new address that will be transferred to the program counter to move the program to a new address in the core memory. The ten skip instructions are all dependent on the presence or absence of a specific condition such as the sign of the A-Accumulator, set overflow latch or memory

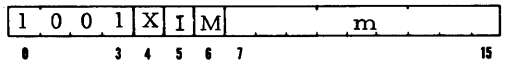
word sign. If the specific condition is present (or absent depending on the instruction as presented below) the next instruction (NI), is skipped and the second successive instruction (NI+1) is executed. The NI is usually a BRU or SPB which branches the program to a new section of the memory. Thus, an SOF instruction would skip a BRU that enters a corrective subroutine, if the overflow latch were not set. If the latch were set (indicating an invalid arithmetic operation), the NI would not be skipped and the corrective subroutine would be entered.

The LOB (Long Branch) instruction consists of two memory words. The first word is an augmented 008 operation code word; the second contains a memory address. The second word is automatically read from memory as part of the normal execution cycle.

The two optional skip instructions (SXB and IXS) are included as part of the index register option. They are used to test the index register and index pointer, respectively.

BRU 11

UNCONDITIONAL BRANCH



The effective address replaces the contents of the program counter.

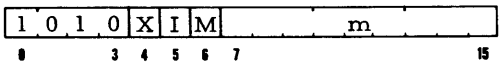
NOTE

If the Program Protect and Instruction Trap option is included (and the Protect Mode switch is ON), when the BRU indirect instruction is executed following a TOI instruction to exit from a priority interrupt routine, bits 2 through 15 of the effective address replace the contents of program counter, and the Protect Latch is set to the state of bit "0" of the effective address.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

SPB 12

STORE PLACE AND BRANCH



The contents of the program counter plus one replace the previous contents of the effective memory address and the effective memory address plus one replaces the previous contents of the program counter.

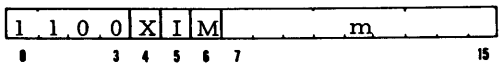
NOTE

Execution of this instruction is modified when caused by a priority interrupt in that the contents of the program counter are unchanged when transferred to the effective memory address. If the Program Protect and Instruction Trap option is included (and the Protect Mode switch is ON), when the SPB indirect instruction is caused by a priority interrupt, the status of the Protect Latch at the time of the interrupt is stored in bit 0 of the effective memory address.

Timing:	2 cycles
Indicators:	None
Registers Affected:	Program Counter

IMS 14

INCREMENT MEMORY AND SKIP



The contents of the effective memory address are increased by one. If the contents of that address then equal zero, the next instruction is skipped.

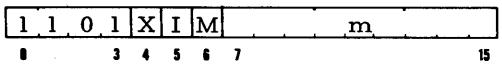
NOTE

This instruction allows any memory cell to be used as an auxiliary index register.

Timing:	3 cycles
Indicators:	None
Registers Affected:	Program Counter

CMA 15

COMPARE MEMORY AND A-ACCUMULATOR (3-WAY)



The contents of the effective memory address are algebraically compared to the contents of the A-Accumulator.

If A M, the program proceeds to the next successive instruction.

If A = M, the next instruction is skipped.

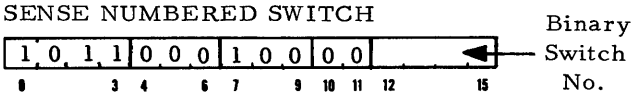
If A M, the next two instructions are skipped.

The contents of the memory and of the A-Accumulator are unchanged.

Timing:	3 cycles
Indicators:	None
Registers Affected:	Program Counter.

SNS 1304

SENSE NUMBERED SWITCH

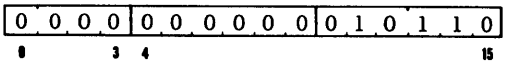


Tests to see if a specific control panel switch (0-15) is set; if switch is NOT set, the next instruction is skipped.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

IBS 00-26

INCREMENT B-ACCUMULATOR AND SKIP



The contents of the B-Accumulator are increased by one. If the contents of the B-Accumulator are then zero or positive, the next instruction is skipped.

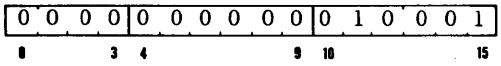
NOTE

This instruction can be used as part of an indexing loop in which the contents of the B-Accumulator (index register) are used to modify the operand address of an instruction, then IBS and branch back to repeat the loop. If the index number has a negative sign, the IBS instruction will eventually decrease the absolute value to zero (positive sign) and will then skip the branch instruction and proceed with the remainder of the program.

Timing:	1 cycle
Indicators:	None
Registers Affected:	B-Accumulator, Program Counter

SAS 00-21

SKIP ON A-ACCUMULATOR SIGN (3-WAY)



If the sign of the A-Accumulator is negative, the next successive instruction is executed.

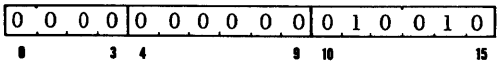
If the contents of the A-Accumulator are zero, the next instruction is skipped.

If the sign of the A-Accumulator is positive and the contents are greater than zero, the next two instructions are skipped.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

SAZ 00-22

SKIP IF A-ACCUMULATOR IS ZERO

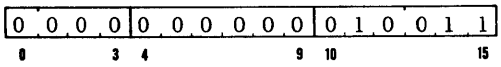


If the contents of the A-Accumulator are zero, the next instruction is skipped.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

SAN 00-23

SKIP IF A-ACCUMULATOR IS NEGATIVE

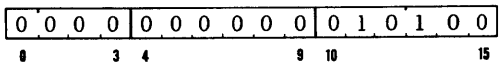


If the sign of the A-Accumulator is negative, the next instruction is skipped.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

SAP 00-24

SKIP IF A-ACCUMULATOR IS POSITIVE

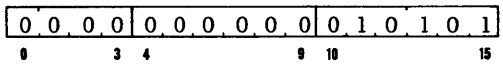


If the sign of the A-Accumulator is positive, the next instruction is skipped.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

SOF 00-25

SKIP NO OVERFLOW



If the arithmetic overflow latch is set, it is reset and the next instruction is executed; if the latch is reset, the next instruction is skipped.

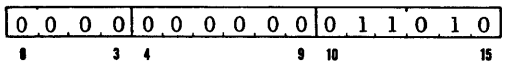
NOTE

This instruction is used as a program check on the magnitude of the results of arithmetic operations. The next instruction (NI), executed in the case of an overflow, is usually a BRU to a corrective subroutine. The second sequential instruction (NI+1) is the next instruction of the normal program.

Timing:	1 cycle
Indicators:	OVERFLOW is reset
Registers Affected:	Program Counter

SNO 00-32

SKIP NORMALIZED A-ACCUMULATOR



If bit A_1 does not equal bit A_0 of the A-Accumulator, the next instruction is skipped.

NOTE

This instruction is used in conjunction with the left arithmetic shift instruction to normalize the contents of the A-Accumulator.

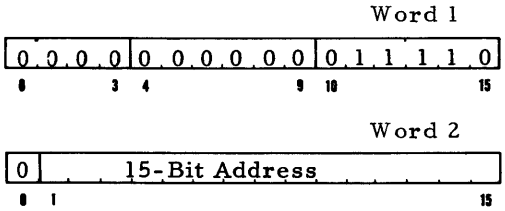
Example:

Loc.	Oper.	Address	Comments
NORM	LSA	1	left shift 1 test
	SNO		for $A_0 = A_1$
	BRU	NORM	shift again if $A_0 = A_1$
	STA		store normalized word
PROG	LAA		remainder of program

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

LOB 00-36

LONG BRANCH



Bits 1 through 15 of the second word replace the contents of the program counter.

NOTE

This instructions allows a branch to any of the 32,768 memory locations available with the full complement of four memory modules. This instruction is

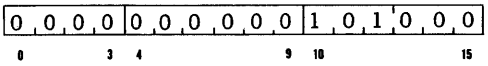
NOTE (Cont'd)

extremely useful as a "return" branch from a subroutine initiated by a Store Place and Branch instruction when the stored program count is in the upper 16,384 memory addresses and the subroutine is operating in the lower 16,384 memory addresses. If the Program Protect and Instruction Trap option is included (and the Protect Mode switch is ON), when the LOB instruction is used following a TOI instruction to exit from a priority interrupt routine, the Protect Latch is set to the state of bit "0" of the effective address.

Timing:	2 cycles
Indicators:	None
Registers Affected:	Program Counter

SXB 00-50

SKIP IF INDEX POINTER IS SET TO B-ACCUMULATOR (OPTIONAL)

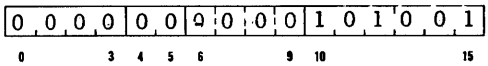


The next instruction is skipped if the index pointer is set to the B-Accumulator.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Program Counter

IXS 00-N-51

INCREMENT INDEX BY N AND SKIP IF POSITIVE (OPTIONAL)



The value of N (0-15, contained in bits 6-9) is added to bits 12-15 of the index register to increase the index register contents by the positive value of N. If the contents of the index register are equal to zero or positive, after the value of N is added, the next instruction is skipped.

NOTE

The option of the IXS instruction is identical to that of the IBS

NOTE (Cont'd)

instruction for the special case where N = 1. The IXS instruction contains the added flexibility of the variable N field. If an N value of zero is specified, only a test of the index register is accomplished.

Timing:	1 cycle if no skip 2 cycles if skip
Indicators:	None
Registers Affected:	Index Register, Program Counter

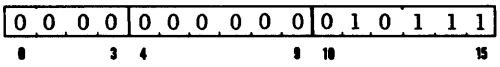
LOGICAL INSTRUCTIONS

The five logical instructions, all augmented 00_8 words, affect only the A- and B-Accumulators. These instructions are provided to allow the logical modification of instruction and data words.

The ABA and OBA instructions are used to mask (logically remove) portions of a single word and merge (logically combine) portions of two words. The contents of the A-Accumulator can be two's complemented through the NEG instruction, while the sign bit of the A-Accumulator can be complemented by the ASC instruction. Sign magnitude form numbers (true binary form with either + or - sign to show polarity) can be converted to two's complement form and the reverse operation can be performed by use of the CNS instruction (Refer to page 2-14). The NEG, ASC, and CNS are used primarily in the arranging of data formats to satisfy input/output requirements, but also find use in creating special flag, indicator and constant words.

ABA 00-27

AND A- AND B-ACCUMULATORS



The contents of the B-Accumulator form a logical product with the contents of the A-Accumulator. The product is stored in the A-Accumulator and the contents of the B-Accumulator are unchanged.

NOTE

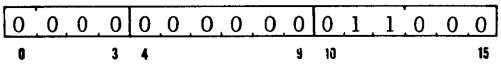
This instruction is used as a masking instruction as follows:

A-Acc.	(0000000011111111)	MASK
B-Acc.	(1010101010101010)	DATA
A-Acc.	(0000000010101010)	LOG. PROD.

Timing: 1 cycle
Indicators: None
Registers Affected: A-Accumulator

OBA 00-30

OR A- AND B-ACCUMULATORS



The contents of the B-Accumulator form a logical sum with the contents of the A-Accumulator. The sum is stored in the A-Accumulator and the contents of the B-Accumulator are unchanged.

NOTE

This instruction is used as a merging instruction as follows:

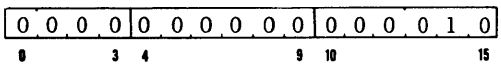
A-Acc (0000000010101010) DATA
B-Acc (1010101000000000) DATA

A-Acc (1010101010101010) LOG. SUM

Timing: 1 cycle
Indicators: None
Registers Affected: A-Accumulator

NEG 00-02

NEGATE THE A-ACCUMULATOR

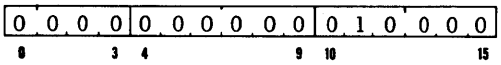


The contents of the A-Accumulator are two's complemented.

Timing: 1 cycle
Indicators: OVERFLOW if operand is minus full scale
Registers Affected: A-Accumulator

ASC 00-20

COMPLEMENT SIGN OF A-ACCUMULATOR

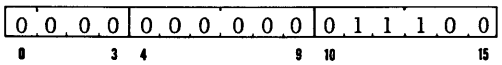


The sign bit of the A-Accumulator is complemented.

Timing: 1 cycle
Indicators: None
Registers Affected: A-Accumulator

CNS 00-34

CONVERT NUMBER SYSTEMS



The least significant bits of a negative-signed number in the A-Accumulator are two's complemented while the sign bit remains unchanged. Positive-signed numbers are not affected.

NOTE

This instruction is used to convert sign-magnitude numbers to two's complement numbers and two's complement numbers to sign-magnitude numbers. Positive-signed numbers are not affected because the form is the same for both number systems.

Timing: 1 cycle
Indicators: OVERFLOW if operand is minus full scale
Registers Affected: A-Accumulator

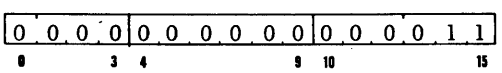
REGISTERS CHANGE INSTRUCTIONS

The five standard instructions are used primarily to manipulate data, create specific formats and perform routine operations connected with double-precision arithmetic. These instructions are also extremely useful in rearranging data in conjunction with shift instructions to achieve necessary word formats.

There are eight optional register change instructions (TBP, TPB, TAX, TXA, TBV, TVB, XPX and XPB). The TBP and TPB instructions are included as part of the memory protect option. The TAX, TXA, XPX and XPB instructions are included as part of the hardware index register option. The TBV and TVB are included as part of the variable base register option.

CLA 00-03

CLEAR A-ACCUMULATOR

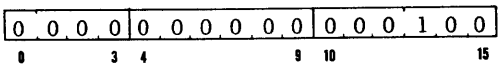


The contents of the A-Accumulator are replaced with all zeros.

Timing: 1 cycle
Indicators: None
Registers Affected: A-Accumulator

TBA 00-04

TRANSFER B-ACCUMULATOR TO A-ACCUMULATOR



The contents of the A-Accumulator are replaced by the contents of the B-Accumulator. The contents of the B-Accumulator are unchanged.

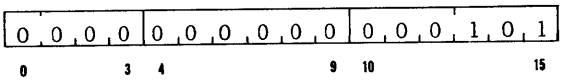
Timing: 1 cycle

Indicators: None

Registers Affected: A-Accumulator

TAB 00-05

TRANSFER A-ACCUMULATOR TO B-ACCUMULATOR



The contents of the B-Accumulator are replaced by the contents of the A-Accumulator. The contents of the A-Accumulator are unchanged.

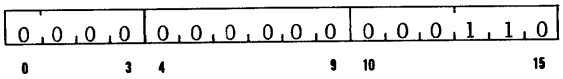
Timing: 1 cycle

Indicators: None

Registers Affected: B-Accumulator

IAB 00-06

INTERCHANGE A- AND B-ACCUMULATORS



The contents of the A-Accumulator are replaced by the contents of the B-Accumulator and the contents of the B-Accumulator are replaced by the contents of the A-Accumulator.

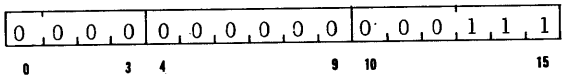
Timing: 1 cycle

Indicators: None

Registers Affected: A-Accumulator
B-Accumulator

CSB 00-07

COPY SIGN OF B-ACCUMULATOR



The CARRY latch is set to the sign of the B-Accumulator and the B-Accumulator sign bit is then set to 0 (plus).

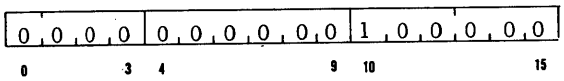
NOTE

This instruction is used to store the carry generated during double-precision addition. If the sign of the B-Accumulator is a one following execution of an AMB instruction, the execution of the CSB causes the one in B_0 to be transferred to the carry latch. An AMA, SMA or NEG instruction must be executed next to insure that the carry bit is added to the contents of A15. No instruction may be executed between the CSB and AMA, SMA or NEG because (1) the carry latch is reset at the end of the execution of all instructions except CSB and (2) the contents of the carry latch are added to the contents of A15 as part of the execution of many instructions.

Timing: 1 cycle
Indicators: None
Registers Affected: B-Accumulator

TBP 00-40

TRANSFER B-ACCUMULATOR TO PROTECT REGISTER (OPTIONAL)



The contents of the Program Protect register are replaced by the contents of the B-Accumulator. The contents of the B-Accumulator are unchanged.

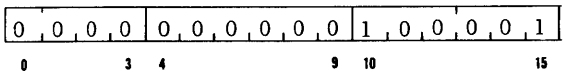
NOTE

See "Program Protect and Instruction Trap (Model 81-080B)" in SECTION VII, OPTIONS, for program protect description.

Timing: 1 cycle
Indicators: None
Registers Affected: Program Protect Register

TPB 00-41

TRANSFER PROTECT REGISTER TO B-ACCUMULATOR (OPTIONAL)



The contents of the B-Accumulator are replaced by the contents of the Program Protect register. The contents of the Program Protect register are unchanged.

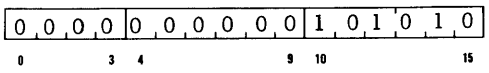
NOTE

See "Program Protect and Instruction Trap (Model 81-080B)" in SECTION VII, OPTIONS, for program protect description.

Timing:	1 cycle
Indicators:	None
Registers Affected:	B-Accumulator

TAX 00-52

TRANSFER A-ACCUMULATOR TO INDEX REGISTER

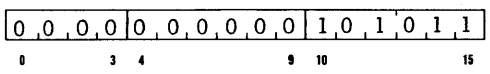


The contents of the index register are replaced by the contents of the A-Accumulator. The contents of the A-Accumulator are unchanged.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Index Register

TXA 00-53

TRANSFER INDEX REGISTER TO A-ACCUMULATOR (OPTIONAL)

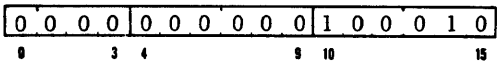


The contents of A-Accumulator are replaced by the contents of the index register. The contents of the index register are unchanged.

Timing:	1 cycle
Indicators:	None
Registers Affected:	A-Accumulator

TBV 00-42

TRANSFER B-ACCUMULATOR TO VARIABLE BASE REGISTER (OPTIONAL)

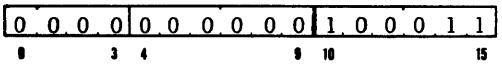


This instruction transfers bits 1 through 6 of the B-Accumulator to bits 1 through 6 of the variable base register. The contents of the B-Accumulator are unchanged.

Timing:	1 cycle
Indicators:	None
Registers Affected:	Variable base register

TVB 00-43

TRANSFER VARIABLE BASE REGISTER TO B-ACCUMULATOR (OPTIONAL)

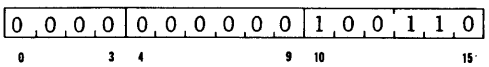


This instruction transfers the 6-bit contents of the variable base register to bit positions 1 through 6 of the B-Accumulator. B-Accumulator bits 0 and 7 through 15, are set to zero. The contents of the variable base register are unchanged.

Timing:	1 cycle
Indicators:	None
Registers Affected:	B-Accumulator

XPX 00-46

SET INDEX POINTER TO INDEX REGISTER (OPTIONAL)

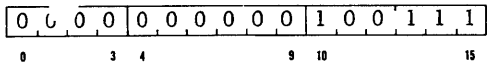


The index pointer flip-flop is set by the execution of this instruction. When this flip-flop is set, the presence of an index flag bit (X=1) in an instruction or indirect address word causes the contents of the index register to be added to the operand address.

Timing:	1 cycle
Indicators:	Index Pointer Light ON
Registers Affected:	None

XPB 00 47

SET INDEX POINTER TO B-ACCUMULATOR (OPTIONAL)



The index pointer flip-flop is reset by the execution of this instruction. When this flip flop is reset, the presence of an index flag bit (X=1) in an instruction or indirect address word causes the contents of the B-Accumulator to be added to the operand address.

NOTE

Operation of the MASTER CLEAR switch resets the index pointer flip-flop.

Timing: 1 cycle
Indicators: Index Pointer Light OFF
Registers Affected: None

SHIFT INSTRUCTIONS

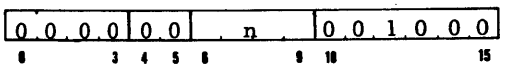
The eight instructions forming the shift group are augmented 008 instructions with bits 6 through 9 containing the binary shift count. While the actual count is in binary code, the number of shifts to be performed by the instruction is usually specified in decimal in symbolic coding. Up to 15 (17$_8$) shifts can be specified for each instruction.

There are two types of shift instructions; arithmetic shifts which bypass the sign bit and logical shifts which move all 16 bits. Right arithmetic shifts move bits from position 1 to 2, 2 to 3, 3 to 4, etc., with bit 1 always set to the same state as the sign bit. The bit originally located in position 15 is shifted off. In left arithmetic shifts, the bits are moved from 15 to 14, 14 to 13, 13 to 12, etc., with zeros being loaded into position 1. The sign bit remains intact.

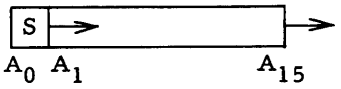
In right logical shifts, the sign is shifted to position 1, 1 to 2, 2 to 3, etc., and zeros are loaded into the sign bit position. If left logical shifts, final zeros are loaded into position 15 and bit 1 shifted to the sign position with the sign bit shifted off.

RSA 00-10

RIGHT SHIFT A-ACCUMULATOR



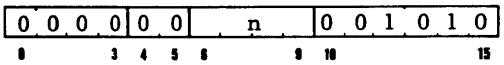
Bits A_1 through A_{15} are shifted right n number of places. The sign bit is unchanged, but supplies the bits (1's if negative, 0's if positive) shifted into A_1 as the original bits are shifted from A_{15}.



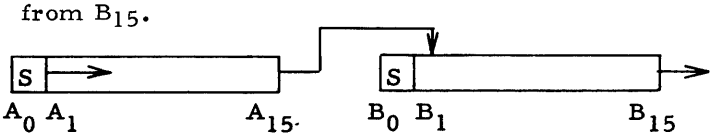
Timing: If n = 1 - 4 2 cycles
n = 5 - 8 3 cycles
n = 9 - 12 4 cycles
n = 13 - 15 5 cycles
Indicators: None
Registers Affected: A-Accumulator

FRA 00-12

FULL RIGHT ARITHMETIC SHIFT



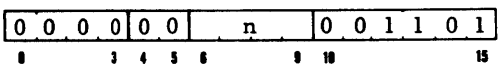
Bits A_1 through A_{15} and B_1 through B_{15} are shifted right n places. Both sign bits are unchanged. A sign supplies bits to A_1 (1's if negative, 0's if positive) while A_{15} supplies bits to B_1. The original B-Accumulator bits are shifted off from B_{15}.



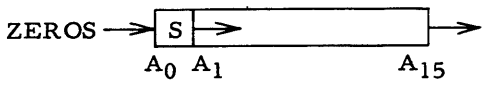
Timing: If n = 1 - 4 2 cycles
n = 5 - 8 3 cycles
n = 9 - 12 4 cycles
n = 13 - 15 5 cycles
Indicators: None
Registers Affected: A-Accumulator, B-Accumulator

RSL 00-15

RIGHT SHIFT LOGICAL A-ACCUMULATOR



Bits A_0 through A_{15} are shifted right n places. Zeros enter A_0 as the original bits are shifted off A_{15}.



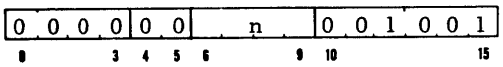
Timing: If n = 1 - 4 2 cycles
n = 5 - 8 3 cycles
n = 9 - 12 4 cycles
n = 13 - 15 5 cycles

Indicators: None

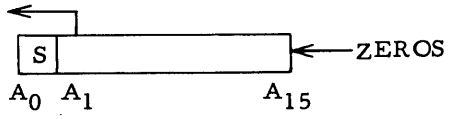
Registers Affected: A-Accumulators

LSA 00-11

LEFT SHIFT A-ACCUMULATOR



Bits A_1 through A_{15} are shifted left n number of places. The sign bit is unchanged. Zeros are shifted into bit A_{15} as the original bits are shifted from A_1.



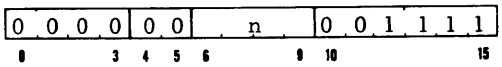
Timing: If n = 1 - 4 2 cycles
n = 5 - 8 3 cycles
n = 9 - 12 4 cycles
n = 13 - 15 5 cycles

Indicators: None

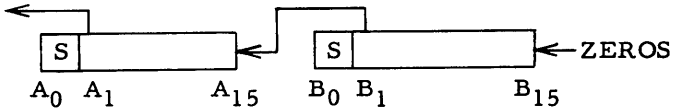
Registers Affected: A-Accumulator

FLA 00-17

FULL LEFT ARITHMETIC SHIFT



Bits A_1 through A_{15} and B_1 through B_{15} are shifted left n places. The signs of the A- and B-Accumulators are unchanged. Zeros are supplied to B_{15}. The output of B_1 is applied to A_{15}, and the original A-Accumulator bits are shifted off from A_1.



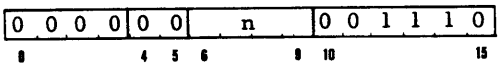
Timing: If n = 1 - 4 2 cycles
n = 5 - 8 3 cycles
n = 9 - 12 4 cycles
n = 13 - 15 5 cycles

Indicators: None

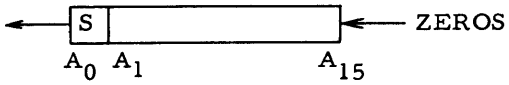
Registers Affected: A-Accumulator,
B-Accumulator

LSL 00-16

LEFT SHIFT LOGICAL A-ACCUMULATOR



Bits A_0 through A_{15} are shifted left n places. Zeros enter A_{15} as the original bits are shifted off A_0.



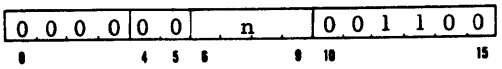
Timing: If n = 1 - 4 2 cycles
n = 5 - 8 3 cycles
n = 9 - 12 4 cycles
n = 13 - 15 5 cycles

Indicators: None

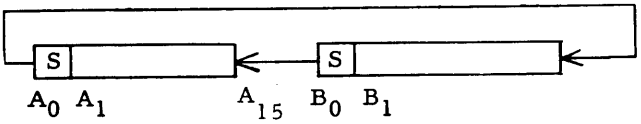
Registers Affected: A-Accumulator

FRL 00-14

FULL ROTATE LOGICALLY



Bits A_0 through A_{15} and B_0 through B_{15} are rotated to the left n number of places. The bits from A_0 enter B_{15} and the bits from B_0 enter A_{15}.



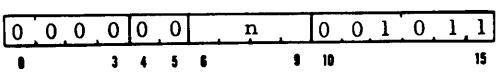
Timing: If n = 1 - 4 2 cycles
n = 5 - 8 3 cycles
n = 9 - 12 4 cycles
n = 13 - 15 5 cycles

Indicators: None

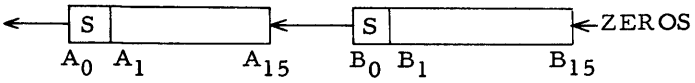
Registers Affected: A-Accumulator,
B-Accumulator

FLL 00-13

FULL LEFT LOGICAL SHIFT



Bits A_0 through A_{15} and B_0 through B_{15} are shifted left n places. The bits from B_0 enter A_{15}, the original A-Accumulator bits are shifted off from A0 and zeros enter B_{15}.



Timing:	If n = 1 - 4 2 cycles n = 5 - 8 3 cycles n = 9 - 12 4 cycles n = 13 - 15 5 cycles
Indicators:	None
Registers Affected:	A-Accumulator, B-Accumulator

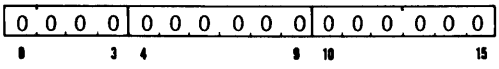
CONTROL INSTRUCTIONS

The five instructions in this group are used for general "housekeeping" functions required by the program. The HLT (Halt) instruction stops the computer after loading the next instruction into the instruction register. the NOP (No Operation) instruction performs no function other than to reserve a program slot for a future addition or to delay the program to match real-time input. The TOI (Turn Off Interrupt), PIE (Priority Interrupt Enable) and PID (Priority Interrupt Disable) allow program control of the priority interrupt circuits as described in the following paragraphs.

The HLT, NOP, and TOI instructions are single-word augmented 00_8 operation code words. The PIE and PID instructions are augmented 13_8 words and both include a second word containing the priority interrupt channel numbers. The second words are automatically read from memory as part of the normal execution cycle of the instruction.

HLT 00-00

HALT



Halts the operation of the computer.

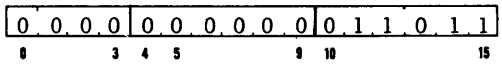
NOTE

The computer stops with the address of the next instruction in the program counter. The START switch is closed to initiate the I cycle of the addressed instruction. The program counter may be manually reset and set to a new starting address prior to closing the START switch.

Timing:	n.a.
Indicators:	HALT
Registers Affected:	None

NOP 00-33

NO OPERATION



No operation is performed.

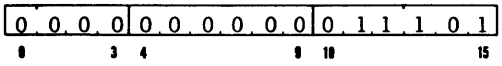
NOTE

This instruction is used to reserve memory locations for instructions to be added within the program encompassing that memory location. It may also be used to delay a program to match a real-time input or output transfer rate.

Timing:	1 cycle
Indicators:	None
Registers Affected:	None

TOI 00-35

TURN OFF INTERRUPT



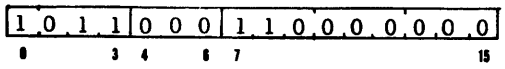
Sets a control latch associated with the highest active priority interrupt so that the interrupt will be reset by the next Long Branch or indirectly addressed Unconditional Branch instruction.

Timing:	1 cycle
Indicators:	None
Registers Affected:	None

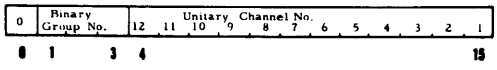
PIE 1306

PRIORITY INTERRUPT ENABLE

Word 1



Word 2



Enables any combination of the 12 priority interrupt levels belonging to the selected priority interrupt group. Bits 15 through 4 of the second word are set to ONES to enable interrupt levels 1 through 12.

NOTE

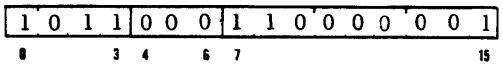
This instruction allows a selected number of 96 priority interrupt levels (12 levels/group x 8 groups) to be enabled for operation.

Timing:	2 cycles
Indicators:	None
Registers Affected:	None

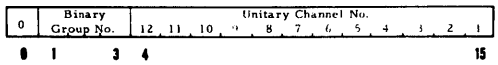
PID 1306-01

PRIORITY INTERRUPT DISABLE

Word 1



Word 2



Disables any combination of the 12 priority interrupt levels belonging to the selected priority interrupt group. Bits 15 through 4 of the second word are set to ones to disable levels 1 through 12.

NOTE

This instruction allows a selected number of 96 priority interrupt levels (12 levels/group x 8 groups) to be disabled.

Timing:	2 cycles
Indicators:	None
Registers Affected:	None

INPUT/OUTPUT INSTRUCTION

Input/output instructions contain 13$_8$ or 17$_8$ operation codes. Several of the input/output instructions are two word instructions. The two instruction words are stored in sequential memory locations with the second word called automatically by the 810B computer. The six I/O instructions are:

COMMAND EXTERNAL UNIT (CEU)

TEST EXTERNAL UNIT (TEU)

ACCUMULATOR WORD OUTPUT TO PERIPHERAL (AOP)

MEMORY WORD OUTPUT TO PERIPHERAL (MOP)

ACCUMULATOR WORD INPUT FROM PERIPHERAL (AIP)

MEMORY WORD INPUT FROM PERIPHERAL (MIP)

Two instructions, A Input (AIP) and A Output (AOP) are provided to enable words or characters to be transferred between the A-Accumulator and peripheral units. These instructions provide a convenient character assembly/disassembly capability. Each of these instructions occupies a single memory location. The two instructions, Memory Input (MIP) and Memory Output (MOP), enable words or characters to be transferred directly between specified memory locations and peripheral units. The instruction Command External Unit (CEU) enables all system devices connected to the computer to be controlled by the program. The CEU instruction is used to initiate Block Transfer Control units as well as to control computer peripheral devices and special system units. The Test External Unit (TEU) instruction is provided to enable system devices to be tested by the computer. The test result causes the instruction following the TEU to be either executed or skipped. Two memory locations are required to store the MIP, MOP, CEU and TEU instructions.

Data or command word transfer instructions can be executed in either of two modes - Wait Mode or Skip Mode, as defined in the following paragraphs.

a. Wait Mode - In this mode, the transfer is not made until the unit sends a "Ready" signal to the computer. The computer continues to test for the Ready signal each machine cycle and then executes the transfer during the first cycle following the recognition of the Ready signal. After the transfer, the device is disconnected and the next instruction in sequence is executed. The specific meaning of the Ready signal is defined in each I/O instruction description.

b. Skip Mode - In this mode, the Ready signal is tested only once. If the Ready signal is present, the transfer is executed. The Program Counter is then advanced to cause the next instruction to be skipped. If the device indicates "Not Ready", the device is disconnected and the Program Counter is advanced to cause the next instruction to be executed. This conditional skip features enables all I/O instructions (except TEU) to perform the total function of "Connect Unit, Test for Ready, Transfer if Ready, and Disconnect Unit".

The flow chart showing the execution of the AIP and AOP instructions is shown in figure 2-3. As shown in the flow chart, the value of the Wait Flag determines whether the instruction is executed in the Wait or Skip Mode. The MIP, MOP and CEU instructions are executed in the same manner, except that the Program Counter is advanced by one before the transfer is made in order to obtain the operand address.

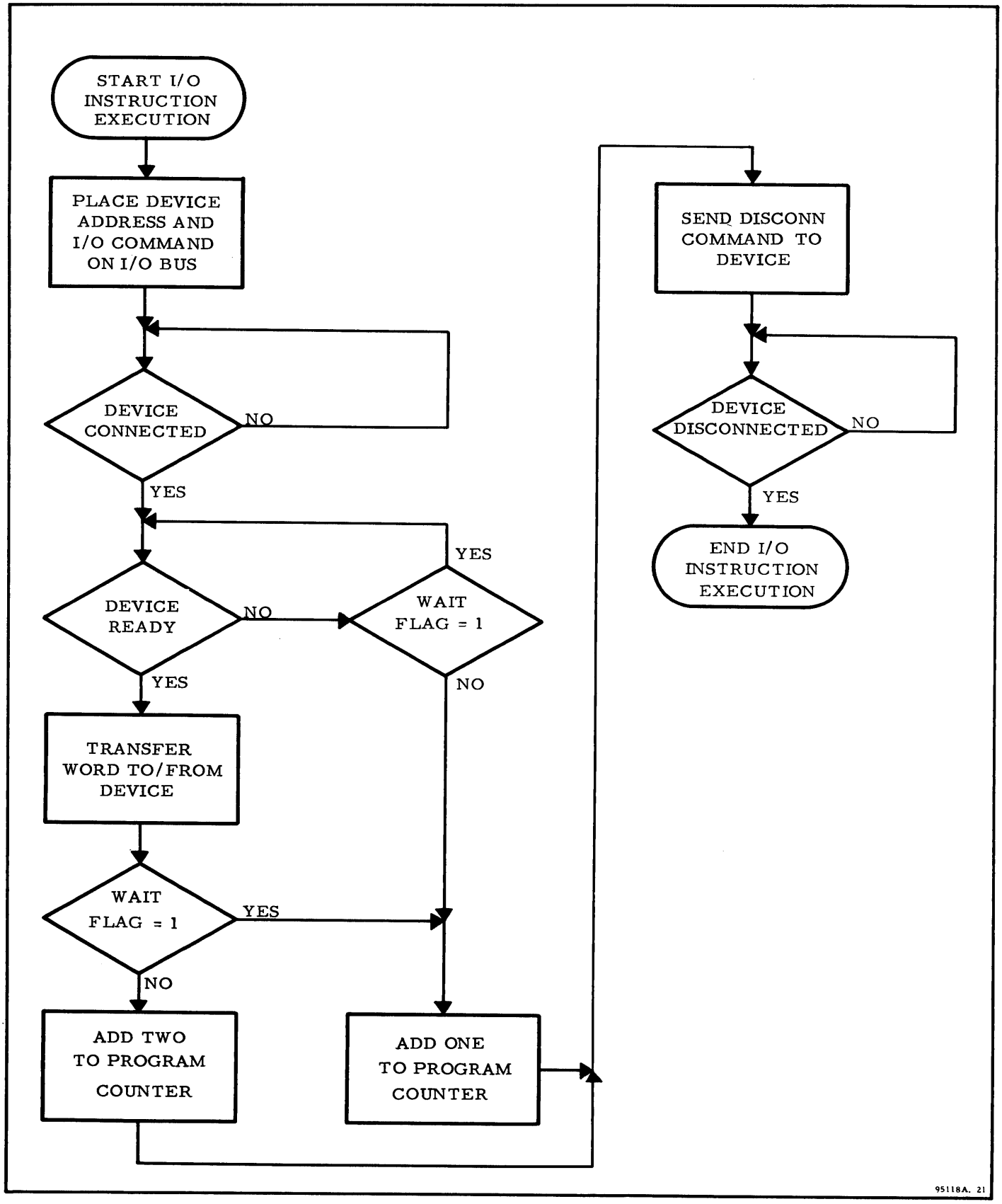


Figure 2-3. AIP/AOP Instruction Execution Flow Chart

Execution of the TEU instruction requires no Ready Test. An on-line unit is always "Ready" to be tested. The Test word is always transferred to the device and a Test Return signal is tested. The result of the test is a conditional skip of the next instruction.

In addition to providing selectable execution modes, the two-word I/O instructions (MIP, MOP, CEU, and TEU) provide two selectable operand addressing modes, Immediate Mode and Address Mode as defined in the following paragraphs.

a. Immediate Mode - In this mode, the second instruction word is treated as the operand. In executing MOP, CEU, and TEU instructions, the contents of the second instruction location are transferred to the specific unit. MIP execution consists of transferring a word or character from a specified device into the second instruction location.

b. Address Mode - In this mode, the second instruction word is interpreted as the operand address. The indirect address format is used in the second instruction word. Therefore, indexing and indirect chaining may be used in addressing the operand. The addressing mode is specified in the first instruction word by the value of the Indirect Address Flag (I). If I is a ONE, the Address Mode is executed.

The format of the I/O instruction words is in figure 2-4. The specific coding used in each instruction is defined in the individual instruction descriptions. Bit definitions are given in table 2-2.

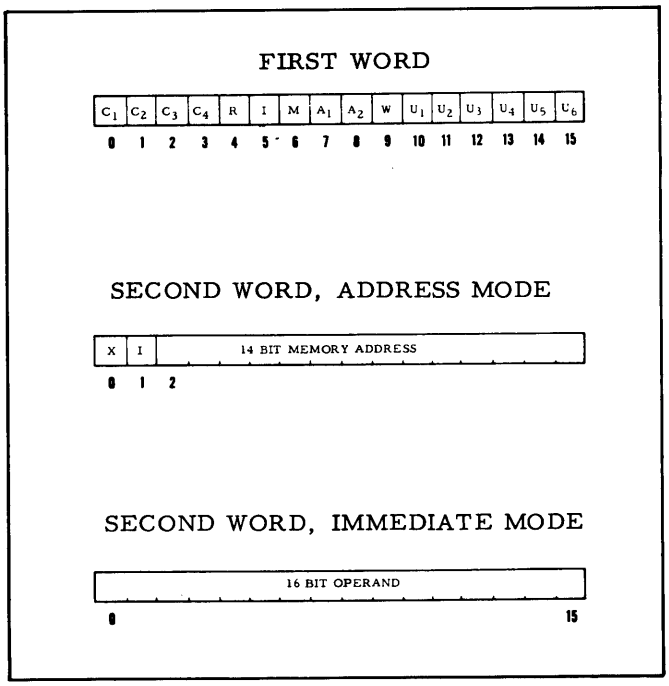


Figure 2-4. I/O Instruction Word Format

Table 2-2. Bit Definitions

Symb.	Definition	Contained In
C	Command Code	All I/O Instructions
R	Character Merge Flag	AIP
I	Indirect Address Flag	MIP, MOP, CEU, TEU
M	Map Bit	MIP, MOP, CEU, TEU
A	Augmented Command Code	All
W	Wait Flag	All except TEU
U	Unit Number (00_8 - 77_8)	All
X	Index Flag	MIP, MOP, CEU, TEU

CEU

1300 Immediate - Skip Mode

1301 Immediate - Wait Mode

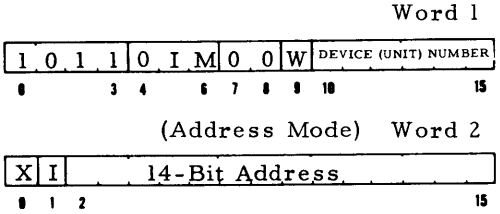
1320 Address - Skip Mode

1321 Address - Wait Mode

1330 Address - Map - Skip Mode

1331 Address - Map - Wait Mode

COMMAND EXTERNAL UNIT



Transfer the command (up to 16 bits) contained in the specified memory location to unit n.

Operand Address Modes:
Immediate - I = 0
Address - I = 1 (First Word)

NOTE

M functions only if the Indirect Flag (bit 5) is a "1". If bit 5 and bit 6 are both "1" bits the MSB

NOTE (Cont'd)

of the program counter is merged with the Indirect Address. If bit 5 is a "1" and bit 6 is a "0" the MSB of the Indirect Address is set to a "0". This feature allows the program to be executed in upper memory (MAP 40 or greater) in the same manner as it is executed in lower memory.

Execution Modes: Skip (W = 0), Wait (W = 1)

Transfer Criterion: A unit answers "Ready" to a CEU test if the unit can immediately start execution of any new function command.

NOTE

The bits in most unit command codes are micro-programmed. Hence, either one or several function commands may be transferred to a unit by execution of a single CEU instruction. Refer to Section VI for the definition of the standard unit command codes.

Timing: 4 cycles + wait
Indicators: I/O WAIT
Registers Affected: None

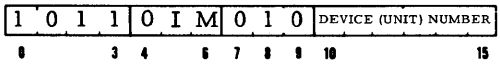
TEU 1302 Immediate Mode

1322 Address Mode

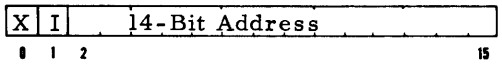
1332 Address, Map Mode

TEST EXTERNAL UNIT

Word 1



(Address Mode) Word 2



Transfers the test code (up to 16 bits) contained in the specified memory location to unit n. A return signal from the unit is then tested, and the program counter is advanced accordingly. If the return signal indicates a "Ready" or "Go" condition, the next instruction in sequence is skipped. A return signal indicating a "Not Ready" or abnormal condition causes the next instruction to be executed.

Operand Address Modes: Immediate - I = 0
Address - I = 1
(First Word)

NOTE

M functions only if the Indirect Flag (bit 5) is a "1". If bit 5 and bit 6 are both "1" bits the MSB of the program counter is merged with the Indirect Address. If bit 5 is a "1" and bit 6 is a "0" the MSB of the Indirect Address is set to a "0". This feature allows the program to be executed in upper memory (MAP 40 or greater) in the same manner as it is executed in lower memory.

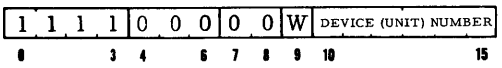
Execution Modes: This instruction is always executed in the same mode. An on-line unit is always "Ready" to accept a test code. Therefore, the code is always transferred and the return is always tested. The Wait Flag is not used.

Timing: 4 cycles + wait
Indicators: None
Registers Affected: None

AOP 1700 Skip Mode

1701 Wait Mode

ACCUMULATOR WORD OUTPUT TO PERIPHERAL



Transfers a word from the A-Accumulator to unit n. Character oriented units accept only bits A_0 - A_7.

Execution Modes: Skip (W = 0), Wait (W = 1)

Transfer Criterion: A unit answers "Ready" to an AOP test if the unit can immediately receive a new word or character.

Timing: 4 cycles + wait
Indicators: I/O WAIT
Registers Affected: None

MOP 1704 Immediate, Skip Mode

1705 Immediate, Wait Mode

1724 Address, Skip Mode

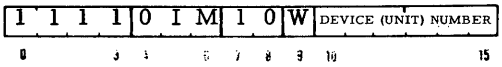
1725 Address, Wait Mode

1734 Address, Map, Skip Mode

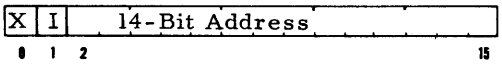
1735 Address, Map, Wait Mode

MEMORY WORD OUTPUT TO PERIPHERAL

Word 1



(Address Mode) Word 2



Transfers a word from the specified memory location to unit n. Character oriented units accept only bits m_0 - m_7 from the specified memory location.

Operand Address Modes:
- Immediate - I = 0
- Address - I = 1 (First Word)

NOTE

M functions only if the Indirect Flag (bit 5) is a "1". If bit 5 and bit 6 are both "1" bits the MSB of the program counter is merged with the Indirect Address. If bit 5 is a "1" and bit 6 is a "0" the MSB of the Indirect Address is set to a "0". This feature allows the program to be executed in upper memory (MAP 40 or greater) in the same manner as it is executed in lower memory.

Execution Modes: Skip (W = 0), Wait (W = 1)

Transfer Criterion: A unit answers "Ready" to an MOP test if the unit can immediate receive a new word or character.

Timing: 4 cycles + wait

Indicators: I/O WAIT

Registers Affected: None

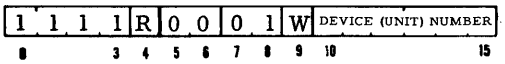
AIP 1702 Skip Mode

1703 Wait Mode

1742 Merge, Skip Mode

1743 Merge, Wait Mode

ACCUMULATOR WORD INPUT FROM PERIPHERAL



Transfers a word or character from unit n into the A-Accumulator. Character oriented units transfer characters into bits A_8 - A_{15}.

NOTE

This instruction contains a convenient provision for character assembly in the A-Accumulator. If the optional Merge Flag (R) is a ONE, the input character is added to the contents of the A-Accumulator. If (R) is zero, the A-Accumulator is cleared prior to the input of a character or word. Therefore, an 8-bit character can be read, with R = 0, shifted left 8 bit positions then merged with the next character read with an R = 1.

Execution Modes: Skip (W = 0), Wait (W = 1)

Transfer Criterion: A unit answers "Ready" to an AIP test if the unit has a word or character ready for immediate transfer.

Timing: 4 cycles + wait

Indicators: I/O WAIT

Registers Affected: A-Accumulator

MIP 1706 Immediate, Skip Mode

1707 Immediate, Wait Mode

1726 Address, Skip Mode

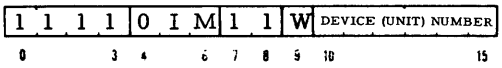
1727 Address, Wait Mode

1736 Address, Map, Skip Mode

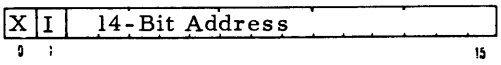
1737 Address, Map, Wait Mode

MEMORY WORD INPUT FROM PERIPHERAL

Word 1



(Address Mode) Word 2



Transfers a word or character from unit n to the specified memory location. Character oriented units transfer characters into bits m_8 - m_{15} of the specified memory location.

Operand Address Modes:	Immediate - I = 0 Address - I = 1 (First Word)

NOTE

M functions only if the Indirect Flag (bit 5) is a "1". If bit 5 and bit 6 are both "1" bits the MSB of the program counter is merged with the Indirect Address. If bit 5 is a "1" and bit 6 is a "0" the MSB of the Indirect Address is set to a "0". This feature allows the program to be executed in upper memory (MAP 40 or greater) in the same manner as it is executed in lower memory.

Execution Modes:	Skip (W = 0), Wait (W = 1)
Transfer Criterion:	A unit answers "Ready" to an MIP test if the unit has a word or character ready for immediate transfer.
Timing:	4 cycles + wait
Indicators:	I/O WAIT
Registers Affected:	None

SECTION III
ASSEMBLY LANGUAGE PROGRAMMING

GENERAL DESCRIPTION

The general format of the SEL 810B Assembly symbolic instruction input (source input) consists of five major fields. These fields are the Location, Operation, Address and Comments - Identification fields. Figure 3-1 shows a sample coding format that may be used for writing source programs in symbolic assembly language. The following paragraphs describe the coding format.

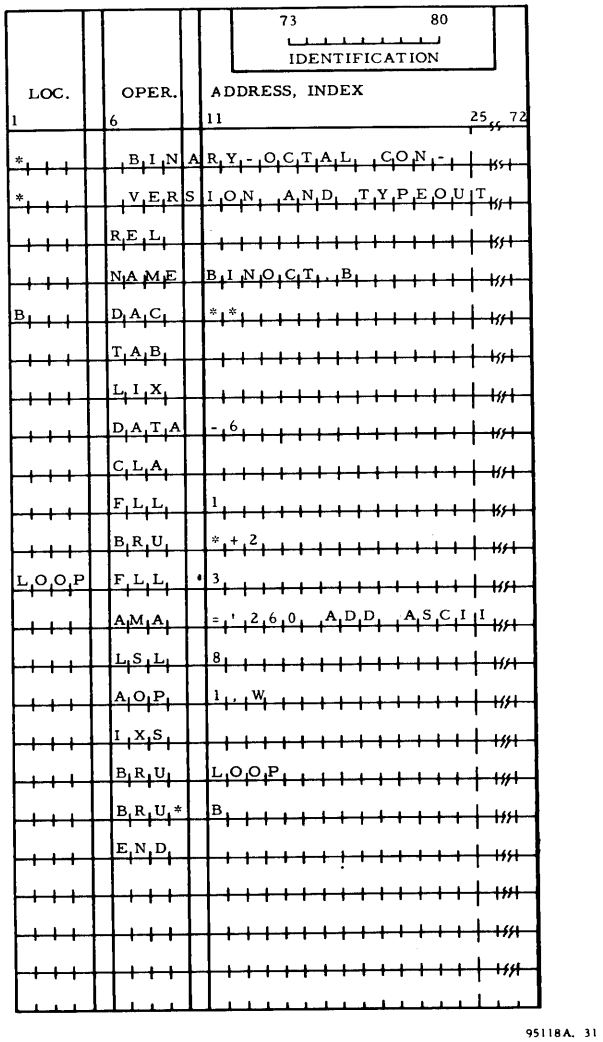


Figure 3-1. Example of Assembler Coding

LOCATION FIELD

The Location field (columns 1 - 4, in figure 3-1) may consist of a symbolic label for the instruction line when it becomes necessary to refer to this location elsewhere in the program. The symbolic label consists of 1 to 4 characters; the first character must be a letter and the remaining characters may be either letters or digits. If no reference to the instruction line is necessary, the Location field may be left blank.

OPERATION FIELD

The Operation field (columns 6 - 9, in figure 3-1) consists of a mnemonic computer instruction or pseudo-operation. A list of mnemonic instructions and pseudo-operations is given in tables 3-2 and 3-4.

Mnemonic computer instructions consist of 3 letters. The mnemonic instruction must be "left-justified" in the Operation field, i.e., written in columns 6, 7 and 8. If the instruction address is to be made indirect, the 3-letter mnemonic is followed by an asterisk (*) written in column 9.

Pseudo-operations consist of 3 or 4 letters and represent either data definition or instructions to the assembly program.

ADDRESS FIELD (VARIABLE FIELD)

Memory reference instructions use the variable field to define the operand address which may be followed by a comma and a one (, 1), to signify indexing. Some other instructions, such as shift and I/O machine operations and some pseudo-operations, have special formats for the variable field which are defined in tables 3-1, 3-2, 3-3 and 3-4. If no address field definition is required the address field is left blank.

Operand Address Formats

No Address. The address field may be left blank if no operand address is required.

Symbolic Address. Consists of 1 to 4 characters starting with a letter.

External Symbolic Address. An external symbolic address consists of a dollar sign ($) followed by 1 to 6 characters, the first of which is a letter. This external variable is presumed not defined

within the program in which it is contained but refers to a subroutine or item located in a different subprogram or located on the library tape. No address arithmetic may be performed on External Symbolic Addresses.

Absolute Address. When reference to a fixed memory location is required or when the address represents a count (such as in a shift instruction). The address consists of digits only and is presumed decimal. Octal addresses are preceded by an apostrophe (').

Current Location. The location of this instruction is used as the instruction's address if a single asterisk (*) appears in the address sub-field. This allows for reference to this or nearby instructions without having to assign a symbolic name to that instruction.

Address Arithmetic. Any current location (*), symbolic (NAME), or absolute ('1234) address may be joined with a constant, current locations (*), symbolic (NAME) or absolute (1234) address by an intervening plus (+) or minus (-) operator to define an effective address (NAME + 4). The above may be extended to more than two operands (A - B + 2).

Literal Address. Literal addresses allow a constant to be defined, assigned to a memory cell and that assignment location used as the address for this instruction. All constants defined in literal addresses will optimize storage so that all identical constants (regardless of their format) will be assigned only once. A literal address consists of an equal sign (=) followed by the constant. Any decimal integer, octal number, single asterisk (current location), previously defined symbolic name or combination of these formats joined by a + or - may follow the equal sign in a literal address.

Location To Be Filled. A double asterisk (**) indicates the address portion of this instruction is to be filled in by the object program at run time and is identical to an absolute address of 000.

The assembly program presumes the computer has a 15-bit address and, therefore, does not attempt to reduce the argument address to a 9-bit address. When the resulting object tape is loaded by the loader into memory starting at a location determined by the operator, these 15-bit addresses are modified as follows:

(a) If the argument address is located in MAP zero, the address is truncated to 9 bits and the MAP bit is set to zero.

(b) If the argument address is located in the same MAP as the instruction in which it is contained, the address is truncated to 9 bits and the MAP bit is set to one.

(c) Otherwise, truncate the 15-bit address to 14 bits and store the 14-bit address and its indirect and index bits automatically into a cell in MAP zero, set the 9-bit address of this cell in the instruction being loaded, and set the MAP bit to zero and the indirect bit to one.

Table 3-1 lists examples of address field entries.

Table 3-1. Example Address Field Entries

Field Entry	Description
	No Address
0, 1	Absolute Zero Address, Indexed
ALPH	Symbolic Address
ALPH, 1	Symbolic Address and Index
519	Absolute Decimal Address
'1067, 1	Absolute Octal Address, Indexed
NAME+4	Address Arithmetic
COMN-2, 1	Address Arithmetic, Indexed
ALPH-PHPA+2, 1	Symbolic Address Arithmetic Indexed
=100	Literal Decimal Constant
='41237	Literal Octal Constant
*	Current Location
*-3, 1	Near-by Address, Indexed
**	Address to be Filled
**, 1	Indexed Address to be Filled
$SQRT	External Symbolic Address

COMMENTS FIELD

The comments field starts immediately after the first space in the variable address field. This field has no effect on the assembler but is printed out on the symbolic listing if a listing is requested.

Any line which has an asterisk (*) in the first character position of that line will be considered a line of comments. (See line 1 in figure 3-1.)

Because of width limitations on the typewriter, comments appearing after column 50 will be output only on the line printer.

IDENTIFICATION FIELD

This field is not checked and is considered as part of the comments. It is provided as a programmers aid. For example, it may be used to identify a card or cards in a card deck or for sequencing the card deck. It is located in columns 73 thru 80.

MNEMONIC COMPUTER INSTRUCTIONS

The computer instructions listed in table 3-2 will be accepted by the SEL 810B assembly program. All permissible fields are shown in the Allowable Fields column with all required fields underlined. Any of the described symbolic notations in this manual may be used in the variable fields, providing they are defined.

Absolute notations for the variable fields are shown in table 3-3.

Table 3-2. SEL 810B Mnemonic Instructions

MNEMONIC Instruction	Allowable Fields	Description
AMA	AMA* Addr, 1	Add Memory to A-Accumulator
AMB	AMB* Addr, 1	Add Memory to B-Accumulator
SMA	SMA* Addr, 1	Subtract Memory from A-Accumulator
MPY	MPY* Addr, 1	Multiply B-Accumulator times Memory
DIV	DIV* Addr, 1	Divide A and B-Accumulator by Memory
RNA	RNA	Round A-Accumulator by MSB in B-Accumulator
OVS	OVS	Overflow Set
LAA	LAA* Addr, 1	Load A-Accumulator from Memory
LBA	LBA* Addr, 1	Load B-Accumulator from Memory
STA	STA* Addr, 1	Store Memory from A-Accumulator
STB	STB* Addr, 1	Store Memory from B-Accumulator
LIX	LIX DATA	Load Index (Immediate Mode)
		or
	LIX* DAC* Addr, 1	(Address Mode)
STX	STX DATA	Store Index (Immediate Mode)
		or
	STX* DAC* Addr, 1	(Address Mode)
LCS	LCS	Load Control Switches in A-Accumulator
BRU	BRU* Addr, 1	Unconditional Branch
SPB	SPB* Addr, 1	Store Place and Branch

Table 3-2. SEL 810B Mnemonic Instructions (Cont'd)

MNEMONIC Instruction	Allowable Fields	Description
SNS	SNS Switch no.	Skip if Console Switch Not Set
IMS	IMS* Addr, 1	Increment Memory and Skip
CMA	CMA*Addr, 1	Compare Memory and A-Accumulator (3 way) n+1(-), n+2 (0), n+3 (+)
IBS	IBS	Increment B-Accumulator (Index) and Skip
SAZ	SAZ	Skip if A-Accumulator is Zero
SAP	SAP	Skip if A-Accumulator is Positive
SAN	SAN	Skip if A-Accumulator is Negative
SOF	SOF	Skip NO Overflow
SAS	SAS	Skip on A-Accumulator sign (3 way) n+1 (-), n+2 (0), n+3 (+)
SNO	SNO	Skip if A-Accumulator is not Normalized
LOB	LOB	Long Branch
	EAC Addr, 1	
SXB	SXB	Skip if Index Pointer is Set to B-Accumulator
IXS	IXS	Increment Index and Skip if Positive
ABA	ABA	AND A-Accumulator and B-Accumulator
OBA	OBA	OR A-Accumulator and B-Accumulator
NEG	NEG	Negate A-Accumulator
ASC	ASC	Complement A-Accumulator Sign
CNS	CNS	Convert Number System
CLA	CLA	Clear A-Accumulator
TAB	TAB	Transfer A-Accumulator to B-Accumulator
IAB	IAB	Interchange A-Accumulator and B-Accumulator
CSB	CSB	Transfer B-Accumulator Sign to Carry and Clear B-Accumulator Sign to Positive
TBA	TBA	Transfer B-Accumulator to A-Accumulator
TAX	TAX	Transfer A-Accumulator to Hardware Index Register

Table 3-2. SEL 810B Mnemonic Instructions (Cont'd)

MNEMONIC Instruction	Allowable Fields	Description
TXA	TXA	Transfer Index Register to A-Accumulator
TAP	TAP	Transfer B-Accumulator to Protect Register
TPA	TPA	Transfer Protect Register to B-Accumulator
TBV	TBV	Transfer B-Accumulator to Variable Base Register (VBR)
TVB	TVB	Transfer Variable Base Register (VBR) to B-Accumulator
XPX	XPX	Set Index Pointer to X Index Register
XPB	XPB	Set Index Pointer to B-Accumulator
RSA	RSA Count	Right Shift A-Accumulator
LSA	LSA Count	Left Shift A-Accumulator
FRA	FRA Count	Right Shift A-Accumulator and B-Accumulator
FLA	FLA Count	Left Shift A-Accumulator and B-Accumulator
RSL	RSL Count	Right Logical Shift A-Accumulator
FRL	FRL Count	Logical Rotate A-Accumulator and B-Accumulator
LSL	LSL Count	Left Logical Shift A-Accumulator
FLL	FLL Count	Logical Left Shift A-Accumulator and B-Accumulator
HLT	HLT	Halt
NOP	NOP	No Operation
TOI	TOI	Turn Off Interrupt
PIE	PIE DATA Group & Level PIE* DAC* Addr, 1	Priority Interrupt Enable
PID	PID DATA Group & Level PID* DAC* Addr, 1	Priority Interrupt Disable
AOP	AOP Unit, Wait	A-Accumulator Out to Unit (n) (Without wait, skip on ready)
AIP	AIP Unit, Wait, Merge	A-Accumulator Input from Unit (n) (Without wait, skip on ready)

Table 3-2. SEL 810B Mnemonic Instructions (Cont'd)

MNEMONIC Instruction	Allowable Field	Description
MOP	MOP Unit, Wait DATA OR MOP* Unit, Wait, MAP DAC* Addr, 1	Memory Out to Unit (n) (Immediate Mode) OR (Address Mode)
MIP	MIP Unit, Wait, DATA OR MIP* Unit, Wait, MAP DAC* Addr, 1	Memory Input from Unit (n) (Without wait, skip on ready) (Immediate Mode) OR (Address Mode)
CEU	CEU Unit, Wait, DATA OR CEU* Unit, Wait MAP DAC* Addr, 1	Command External Unit (n) (Without wait, skip on ready) (Immediate Mode) OR (Address Mode)
TEU	TEU Unit, DATA OR TEU* Unit, MAP DAC* Addr, 1	Test External Unit (Immediate Mode) OR (Address Mode)

ABSOLUTE NOTATIONS FOR VARIABLE FIELDS

Table 3-3 lists the absolute notations for the variable fields.

Table 3-3. SEL 810B Absolute Notation Formats

Variable Field	Absolute Notation
Addr	5 octal digits ('00000-'77777) 5 decimal digits (00000-32767)
Count	2 octal digits ('00-'17) 2 decimal digits (00-15)
Switch No.	2 octal digits ('00-'17) 2 decimal digits (00-15)
Group	1 octal digit (0-7 = bits 1-3)
Level	4 octal digits (0001-7777= bits 4-15)
Unit	2 octal digits ('01-'77 representing units 1 to 63) or 2 decimal digits (1 to 63)

Table 3-3. SEL 810B Absolute Notation Formats (Cont'd)

Variable Field	Absolute Notation
Wait	1 binary bit (0, No Wait or 1, Wait = bit 9)
Merge	1 binary bit (0, No Merge or 1, Merge = bit 4)

PSEUDO-OPERATION INSTRUCTIONS

This group of instructions is used to instruct the SEL 810B Assembly Program and are not executed by the computer. A description of each pseudo-operation is given in the following paragraphs.

ABS Set the mode of the assembly program to ABSolute. When in this mode, all symbolic addresses will be assigned relative to location 00000 and output in a non-relocatable format.

REL Set the mode of the assembly program to

RELative. When in this mode, all symbolic addresses will be assigned relative to the start load address (assigned when loading the program into memory) and output in a relocatable format compatible with the loader. The assembly program is initialized to the absolute mode and will remain in this mode until changed by an REL pseudo-op).

ORG — The variable field specifies an address. When the assembly is in absolute mode, this address specifies the location of the next instruction. When the assembly is in relative mode, this address will be added to the start load address (assigned when loading the program into memory) in order to specify the location of the next instruction. In either case, all following instructions will be stored sequentially in memory until another ORG pseudo-op is given.

EQU — The symbol in the location field will be assigned the address or value specified in the variable field. Constant values may not exceed 15 bits.

DAC — This pseudo-op is used to generate Direct Address Constants used as argument addresses for subroutine calling sequences, or referred to by indirect instructions. The address in the variable field may be in any of the formats shown previously (variable field formats for instructions). The address will be truncated to 14 bits and will occupy bits 2 to 15 of the resulting word. The address may be indexed and the pseudo-op may be tagged indirect if required. (Setting bits 0 and 1.)

EAC — This pseudo-op is used to generate 15-bit Extended Address Constants used as arguments of Long Branch instructions. Any of the Formats shown previously are acceptable in the variable field, except that the instruction may not be indexed nor made indirect.

DATA — The variable field of this pseudo-op may contain any number and any mixture of the following data item formats. If the location field contains a symbol, it will be assigned the location of the first data item. If more than one data item is present (separated by commas), they will be assigned sequential storage locations.

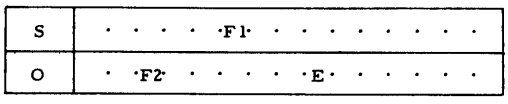
a. Octal Data Item - Format: An optional sign (+ or -), followed by an apostrophe character ('), followed by 0 to 6 octal digits (0 through 7). If less than 6 digits are present, the number will be right-justified with leading zeros added. If a minus sign is present, the number will be 2's complemented; a plus sign is ignored.

b. Decimal Integer - Format: An optional sign (+ or -) followed by 0 to 5 decimal digits (0 through 9). The number will be converted to binary and stored at a scale of B15. The number will be stored positively unless a minus sign is present. A minus sign will cause the 2's complement of the number to be stored.

c. Fixed-Point Single Precision Decimal Data - Format: An optional sign (+ or -), 0 to 5 decimal digits (0 through 9), mixed with an optional decimal point, the letter B, followed by a decimal number between +15 and -15. Example: -3.14157B6. A minus sign will cause the 2's complement of the number to be stored. One word will be generated.

d. Fixed-Point Double Precision Data - Format: An optional sign (+ or -), 0 to 10 digits mixed with an optional decimal point, the letter C, followed by a decimal number between +30 and -30. Example: 103.637942C10. A minus sign will cause the 2's complement of the number to be stored. Two words will be generated.

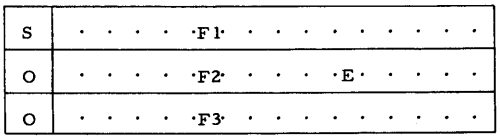
e. Floating Point Data - Format: An optional sign (+ or -), 0 to 7 decimal digits (0 through 9), mixed with an optional decimal point, and optionally followed by a decimal exponent consisting of the letter E, preceding a decimal number between +75 and -75. (Either the decimal point, the letter E, or the sign of the exponent must be present.) Two sequential memory cells are generated for each floating point data item using the following format:



Examples:

	0.1	=	+63146+31775
	5.0325E2	=	+76720+00011
	-1.E-1	=	-14631+47775
	-503.25	=	-01060+00011

f. Floating Point Double Precision Data - Format: An optional sign (+ or -), 0 to 11 digits mixed with an optional decimal point, the letter D, followed by a decimal number between +75 and -75. Three sequential memory cells are generated for each double-precision floating point item using the following format:



E = Characteristic (2's complement if negative).

F1, F2, F3 = Double-Precision Fraction (2's complement if negative).

g. Alphanumeric Data - Format: Two apostrophe characters (' ') followed by any number of characters (including blanks) until another pair of apostrophes is read. The characters within the apostrophe pairs are stored 2 per word (last character left-justified, if necessary).

Example: "ALPHA TEST" is to be stored into memory starting at location 2000.

2000	1	100	000	111	001	100	AL
2001	1	101	000	011	001	000	PH
2002	1	100	000	110	100	000	A-
2003	1	101	010	011	000	101	TE
2004	1	101	001	111	010	100	ST

The above example is in ASR-33 code (FULL ASCII code). This code will be used internally by the assembler to represent alphanumeric data. The I/O handling subroutines will translate from external to internal code and vice versa when necessary depending upon the I/O device in use.

h. Symbolic Address Data - Format: Any symbolic address optionally followed by address arithmetic. The effective address will be stored in memory as a 15-bit address (similar to that generated by the EAC pseudo-op). The address may not be tagged as indexed or indirect.

FORM This pseudo-op is used to setup the format for the FDAT pseudo-op. There is no data generated and no memory locations are used. This pseudo-op allows the programmer to define the bit assignments of 16-bit words generated by the FDAT statement. Up to 8 fields are allowed but the total number of bits must not exceed 16. All FDAT statements that follow a FORM will be in the same format until another FORM is encountered.

Example:

```
FORM 6, 4, 3, 1, 2
```

This assigns the FDAT bits as follows:

Field 1 - 6 bits (bits 0-5)
Field 2 - 4 bits (bits 6-9)
Field 3 - 3 bits (bits 10-12)
Field 4 - 1 bit (bit 13)
Field 5 - 2 bits (bits 14-15)

FDAT This pseudo-op is used to generate data in a format which has been previously defined by a FORM statement. The variable field for this instruction will accept decimal, octal, and alphanumeric data, but will mask off the most significant bits not defined by the previous FORM statement. Multiple FDAT statements may be placed on a card separated by slashes (/). If the location field contains a symbol, it will be assigned the location of the first data item. Example (using the FORM defined above):

```
FDAT    "A", 8, 7, 0, 1/'75, '13, 4, 1, 3
```

This will generate the following two consecutive octal words:

```
'003071
'173347
```

BSS Reserve a block of memory storage starting at the current location and extending for the number of words specified in the variable field. (If the variable field is symbolic, it must have been defined by a previous input line). The location field is optional but if a symbol is inserted in this field, it will refer to the first word in the block.

BES Same as BSS except that if the location field is occupied, it refers to the last+1 word in the block.

CALL This pseudo-op will generate the necessary coding and actions to call in a subroutine from a library tape into memory. The CALL pseudo-op is then replaced by a subroutine transfer instruction (SPB) to the subroutine. The variable field contains the subroutine name. The location field, when occupied, refers to the resulting SPB instruction. Logic is contained within the loader to assure that only one copy of a subroutine is called into memory from the library tape regardless of the number of CALL's for that subroutine. The subroutines name must start with a letter and may contain from 1 to 6 characters. An equally good way to call external subroutines would be with a leading dollar sign on the subroutine's name.

Examples: SPB $SQRT

or CALL SQRT

NAME When writing subroutines for inclusion into a library tape, the name by which the subroutine must be called as specified by the NAME pseudo-op. This must appear as the subroutine's first instruction line(s). The variable field consists of two symbolic names. The first is the name of the subroutine and is 1 to 6 characters long (FORTRAN compatible). The second name is the symbolic entry location for the subroutine and is 1 to 4 characters long, the first character being a letter. More than one NAME pseudo-op may be included in a subroutine if alternate names for the subroutine exist with either the same or different entry points. Also, external variables are defined by the NAME pseudo-op.

ZZZ The instruction bits (0 to 3) are set to 0000. The rest of the instruction is determined by the variable field and the presence of an indirect indicator (*) following the pseudo-op (ZZZ*).

*** Same as ZZZ but indicates the instruction will be filled at run time.

MOR This pseudo-op causes a pause in the assembly process useful when the source program is on more than one tape, and a pause is needed to change tapes.

END This pseudo-op must appear as the last instruction in any program or subroutine being assembled and tells the assembly program that assembly is complete. If the variable field is not blank, it should specify the starting location of the program just assembled.

LIST Set the mode of the symbolic output routine to list the output provided sense switch one is not ON. The assembler assumes the LIST mode until otherwise directed.

NOLS Set the mode of the symbolic output routine to suppress the listing of output unless an error is detected. This pseudo operation remains in control until a LIST pseudo-operation is encountered.

SUMMARY OF PSEUDO-OPERATION INSTRUCTIONS

Table 3-4 summarizes the SEL 810B pseudo-operation instructions and gives specific examples for their use.

Table 3-4. Summary of SEL 810B Pseudo-Operations

Symbolic Location	Pseudo-Operation	Variable Field Entry	Description
	ABS		Set Mode Absolute
	REL		Set Mode Relative
	MAP		Set Single Map Mode
	ORG	'1000	Set Origin of Program
ALPH	EQU	BETA+2	Set Symbol Equal to Symbol

Table 3-4. Summary of SEL 810B Pseudo-Operations (Cont'd)

Symbolic Location	Pseudo-Operation	Variable Field Entry	Description
IND	EQU	2	Set Symbol Equal to Number
OCT	DATA	'12734, -'21, +'6470	Octal Data
	DATA	9876, -3000, +24	Decimal Integer Data
MIX	DATA	23.456B10, -3B6, 12CO	Fixed Point Data
FLOT	DATA	22.3344EO, . 12345D2	Floating Point Data
ALPA	DATA	"HELP", "12-34, A.E2"	Alphanumeric Data
	DATA	X4, TEST+2, A-DLTA+1	Symbolic Address Data
	DATA	3,'77,1.23B4, -1233E-3,X4	Mixed Format Data
	FORM	6, 4, 3, 1, 2	Defines FDAT Format
	FDAT	"A",8,7,0,1/'75,'13,4,1,3	Mixed Formatted Data
TABL	BSS	100	Block Storage Skip (Front Label)
	BES	5	Block Storage Skip (End Label)
	CALL	SIN	Library Tape Call
	NAME	SIN, S21	Library Subroutine Name
	ZZZ	ALPHA	Instr. Bits = 0000
	***	**	Word to be filled at run Time
	MOR		Pause When Assembling
	END	STRT	End of Program
	END		End of Subroutine
AD2	DAC	TIME, 1	Direct Address Constant, Indexed
AD1	DAC*	LEVL	Indirect Address Constant
	EAC	MEM2	Extended Address Constant
AD3	EAC	MEM2, 1	Extended Address Constant, Indexed
	LIST		List
	NOLS		No List

MACRO SYSTEM

The Macro System generates in-line coding according to the respective prototype and parameter list assigned to a given Macro call name.

The general form of a Macro prototype is as follows:

Loc.	Oper.	Address, Index
NAME	MACR A SET OF DETAIL STATEMENTS EMAC	

Columns 1-4	The call name of the Macro which can be any combination of legal characters, blanks included.
Column 5	Blank
Columns 6-9	To denote the beginning of a prototype code MAC or MACR. To denote the end of a prototype code EMA or EMAC.
Column 10	Blank
Columns 11-72	Can be used as comments.

SPECIAL NOTE

Do not use an END or a $ end of job statement in a Macro prototype.

MACRO PROTOTYPE

The prototype is a set of detail statements which can contain elements to be supplied either internally or from a list of parameters. Elements are of three basic types as follows:

- Internal to a given Macro prototype
- Parameter supplied by user
- Fixed element name

The internal assignment applies only to labels and must be of the form @X where the at sign (ASCII 300) must be the first character of the label. The X is a decimal value from one through 16 and can be assigned in any order (not necessarily monotonically) per Macro. Leading zeros are suppressed, @009 is the same as @9. Each call of the same Macro which contains internal labels will generate a unique respective set increased by the last assembler assigned label plus one. The assembler will not allow more than 999 internal labels to be generated. All assignments in excess of 999 will be flagged as an error.

Example of internal label:

Loc.	Oper.	Address, Index
WAIT	MACR	NAME AND BEGIN PROTOTYPE
@1	NOP	INTERNAL LABEL FIXED OP
	NOP	FIXED OPERATION CODE
	NOP	FIXED OPERATION CODE
	BRU @1	FIXED OP CODE, INTERNAL LABEL
	EMAC	END OF WAIT PROTOTYPE

Every call to WAIT will generate @1 into a unique label for each wait loop.

The user supplied parameter can apply to any field of a valid assembler statement. The form of a user parameter is #X where the number sign (ASCII 243) may appear anywhere in a label or value to be specialized and must be immediately followed by a decimal value from one through 16 representing the correct parameter number to be concatenated into the generated element. Leading zeros are suppressed on parameter number assignments. Parameters which are requested but omitted from the list are replaced by a single blank character. Parameter numbers in excess of 16 will not be processed and will be flagged as an error.

Example of user supplied parameter:

Loc.	Oper.	Address, Index
FILL	MACR	NAME AND BEGIN PROTOTYPE
	LAA =#1	CHARACTER TO FILL AREA
	LBA =#2	SIZE OF AREA IN DECIMAL
	STA #3+#2, 1	AREA PLUS SIZE MINUS INDEX
	IBS	INCREMENT INDEX
	BRU *-2	LOOP TO FILL AREA
	EMAC	END OF MACRO PROTOTYPE

```
*  THE GENERAL FORM OF THE CALL
*  STATEMENT IS
*          MFILL  DATA, SIZE, AREA
*SAMPLE
           MFILL  '240, 80, TABL,
*FILL WITH A SPACE
*80 LOCATION TABLE
*LEFT ADDRESS IS CALLED TABL
```

To call a Macro prototype code a M in column 5 followed by the name of the Macro in columns 6-9 with the parameter list in columns 11-72.

Parameter elements in the main program call list are separated by a single level of delimiter which can be a comma, left parenthesis, or right parenthesis. The parameter list may be terminated by one of the three delimiters. Extra sets of parenthesis can be added for clarity but each must be counted when assigning values to the elements of a detail entry. Elements can be assigned in any order or any set of digits provided a parameter exist for the desired elements.

Example:

```
 Loc.        Oper.       Address, Index
 PAR         MACR        USE ONLY EVEN NUM-
                         BERED PARAMETERS
             DATA        #2, #4, #6, #8, #10
             EMAC
             .
             .
             .
            MPAR         (13) (6) ("A") (3B5) ('377)

* THE ABOVE IS NOT THE SAME AS THE
* FOLLOWING
             MPAR        13, 16, "A", 3B5, '377,
* THE ABOVE NEEDS A PROTOTYPE OF THE
* FORM

 PAC         MACR        USE SEQUENTIALLY NUM-
                         BERED PARAMETERS
             DATA        #1, #2, #3, #4, #5
             EMAC

* NOTE THAT A , () MAY NOT BE USED AS
* DATA IN A PARAMETER LIST, IF NEEDED,
* SUPPLY AS OCTAL DATA
```

The fixed element name is any field in which the detail statement supplies the value, operation code, operand or any portion of a statement.

Example:

```
 Loc.        Oper.       Address, Index
 LDB1        MACR        LOAD DOUBLE PRE-
                         CISION CONSTANT 1
             CLA         CLEAR THE MSB TO
                         ZERO
             LBA    =1   LOAD A 1 IN LSB
             EMAC        END OF MACRO PRO-
                         TOTYPE
* USAGE OF LDB1
             MLDB1       NOTE THAT NO
* PARAMETERS ARE NEEDED ALL DATA IS
* SUPPLIED BY THE PROTOTYPE
```

Comments may be entered in a prototype; however, only the asterisk and the next 24 positions will be retained when the prototype is specialized. If a detail line has comments as a continuation of a statement, they will not be processed at the time of specialization.

The MACRO storage area is normally 700_{10} words with a name table capacity of 30 names. The formula for computing the exact number of words needed to store a prototype is as follows:

Sum of words for each statement +1.

The words for each statement = 1+ (number of characters in location field + number of characters in op code field + number of characters in variable field) ÷ 2 + 0 if the remainder is 1, 1 if the remainder is 0. Count internal and parameter supplied labels as 2, that is, #003 + #02 + 6 is counted as 7 characters. A general safe rule-of-thumb would be to multiply the number of detail lines by 5 to obtain the storage requirements for a Macro prototype.

SECTION IV
INPUT / OUTPUT

GENERAL DESCRIPTION

The 810B Computer Input/Output (I/O) structure is designed particularly to meet the requirements of the on-line, real-time computer user. This computer application area imposes the most severe requirements on computer I/O capabilities due to the wide variety of peripheral devices required and the time-sharing mode of operation encountered. Many Systems Engineering Laboratories real-time systems not only have standard data processing peripherals such as card, paper tape, magnetic tape, disc and keyboard/printer devices, but also have a number of interface devices such as data acquisition systems, displays and control and communication units. Therefore, the I/O structure must enable connection of a large number of peripheral devices to the computer and must enable several devices to time-share communication with the computer.

The standard 810B Computer I/O structure consists of an Input/Output Processor (I/OP), which provides "party line" communication with peripheral devices or device controllers. Data is supplied over 64 direct information channels. Figure 4-1 shows how peripheral units are connected to the I/OP by means of the I/O Bus.

The standard I/OP alone is capable of meeting the I/O requirements of many systems. It is ideally suited to real-time applications in that each I/O instruction causes the device addressed by the instruction to be connected to the computer, the data transfer to be made, and then the device to be disconnected. Therefore, the successive transfer of data words to/from two or more different peripheral devices requires no intervening housekeeping operations such as channel and device testing and connection.

The time-sharing capability of the I/OP is further enhanced by the fact that all Systems Engineering Laboratories peripheral devices contain their own data buffers. Hence, the I/OP which contains no buffer, is never busy buffering data to be transferred to/from a device. As a result a two word data transfer instruction can be executed in 3.0 microseconds + wait and the I/OP released immediately for data transfer to/from a different device.

In addition to the I/OP, Block Transfer Control (BTC) units can be added to the 810B to provide a fully-buffered data transfer capability between computer memory and peripheral units. BTC channels enable a block of words up to 32,767 in length to be transferred to or from a peripheral device. One memory cycle is stolen per word transferred.

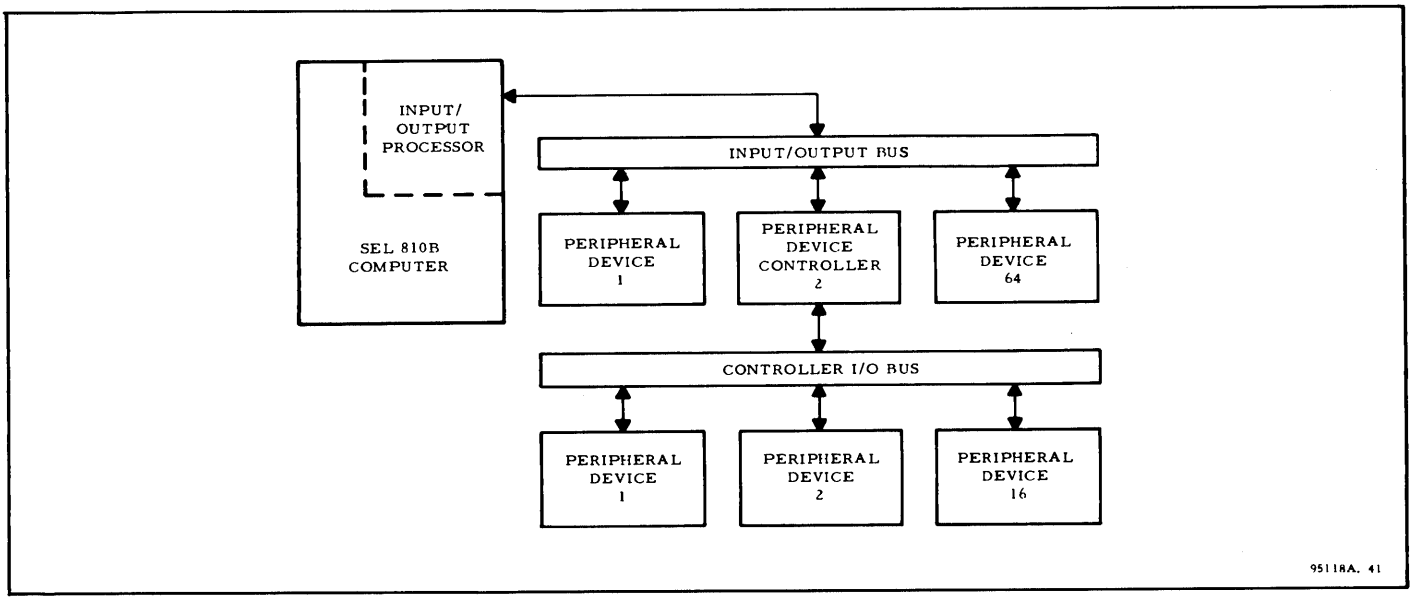


Figure 4-1. Connection of Peripheral Units to the Computer

The predominant reason for adding BTC units to the computer is to free the mainframe to perform internal processing functions while data is being transferred between memory and peripheral devices at high rates. For example, using two BTC units, a continuous stream of data words can be read into computer memory, blocked, and recorded on magnetic tape in gapped format, resulting in a loss of only slightly over two machine cycles (one for input, one for output) per word transferred. For a typical word rate of 20 KC, only an average of 1.50 microseconds of machine time is used per 50 microseconds of elapsed time to accomplish a single word input and output transfer function. The remainder of the time is available for performing such functions as scaling or limit checking of the data.

The optional Computer Graphics Processor (CGP) Model 84-235B is a high-speed data transferring control unit designed to satisfy the specialized needs of the SEL Computer Graphics Systems. The control unit is similar to the 810B optional Block Transfer Control (BTC) with the exception of its specialized operating characteristics and added control functions. A BTC, when outputting data, is unmindful of the nature of this data. However, the CGP examines each word as it comes from memory and either interprets the word as data and sends it to the Computer Graphics System or as an instruction and takes appropriate action.

The instructions allow the CGP to operate on its address counters, thereby freeing the 810B Computer from much of the control unit servicing, and allowing it more time to operate on the buffer areas of the system. This feature allows the use of subroutines to generate frequently used patterns.

The CGP is used in conjunction with the SEL 816A Computer Graphics System to provide the most efficient method of transferring data from the 810B Computer to the display unit. Using the CGP minimizes the amount of computer memory and transfer time required to support the display. The CGP also provides a high degree of flexibility in display format generation since the CGP contains the capability of executing the following instruction: (1) Branch Unconditionally, (2) Store Place and Branch, and (3) Stop. The first two instructions (which have the same execution capabilities as the corresponding computer instructions) enable the contents of non-contigous memory areas to be transferred automatically to the display. This capability enables display programs to be organized to provide maximum usage of closed subroutines that are stored in memory a single time and used as often as required in a given display format. The stop command enables the display unit to automatically control the refresh rate, and maintain a fixed rate regardless of the amount of data being displayed.

INPUT OUTPUT PROCESSOR

Figure 4-2 illustrates a more detailed block diagram of the SEL 810B I/O structure. It shows the connection of both the I/O structure to the computer and the peripheral units to the I/O Bus. Three of the five devices shown have additional connections to BTC units. However, all five devices shown and additional devices up to a total of 64 can be commanded by and can communicate with the computer under single-word program control. The additional, fully-buffered transfer capabilities of the units connected to the BTC units are described in the Block Transfer Control Unit section.

The I/OP provides a positive synchronization control for data flow between the computer and peripheral units. It can synchronize data transfer between a peripheral unit and either memory or the A-Accumulator. The data path for each word or character transferred is controlled by a program executing an input/output instruction.

The basic, automatic execution sequence for all I/O instructions consists of three steps:

1. Connect the device specified by the instruction to the I/O bus.

2. Execute the transfer directly between the device and the A-Accumulator or memory.

3. Disconnect the device from the I/O bus.

Three very significant features of this execution sequence are:

1. The device is always specified by the I/O instruction.

2. The device is always connected to and disconnected from the computer by the execution of the instruction.

3. Data transfers are always made directly between the specified device and the computer with no intermediate buffering.

The result of these three features is that the computer I/O structure is always available for use without testing. It is never "busy", except during the times that I/O instructions are being executed. No channel testing or selection is ever required. In addition, no unit selection instructions are required, since each I/O instruction causes the unit specified by the instruction to be selected for transfer.

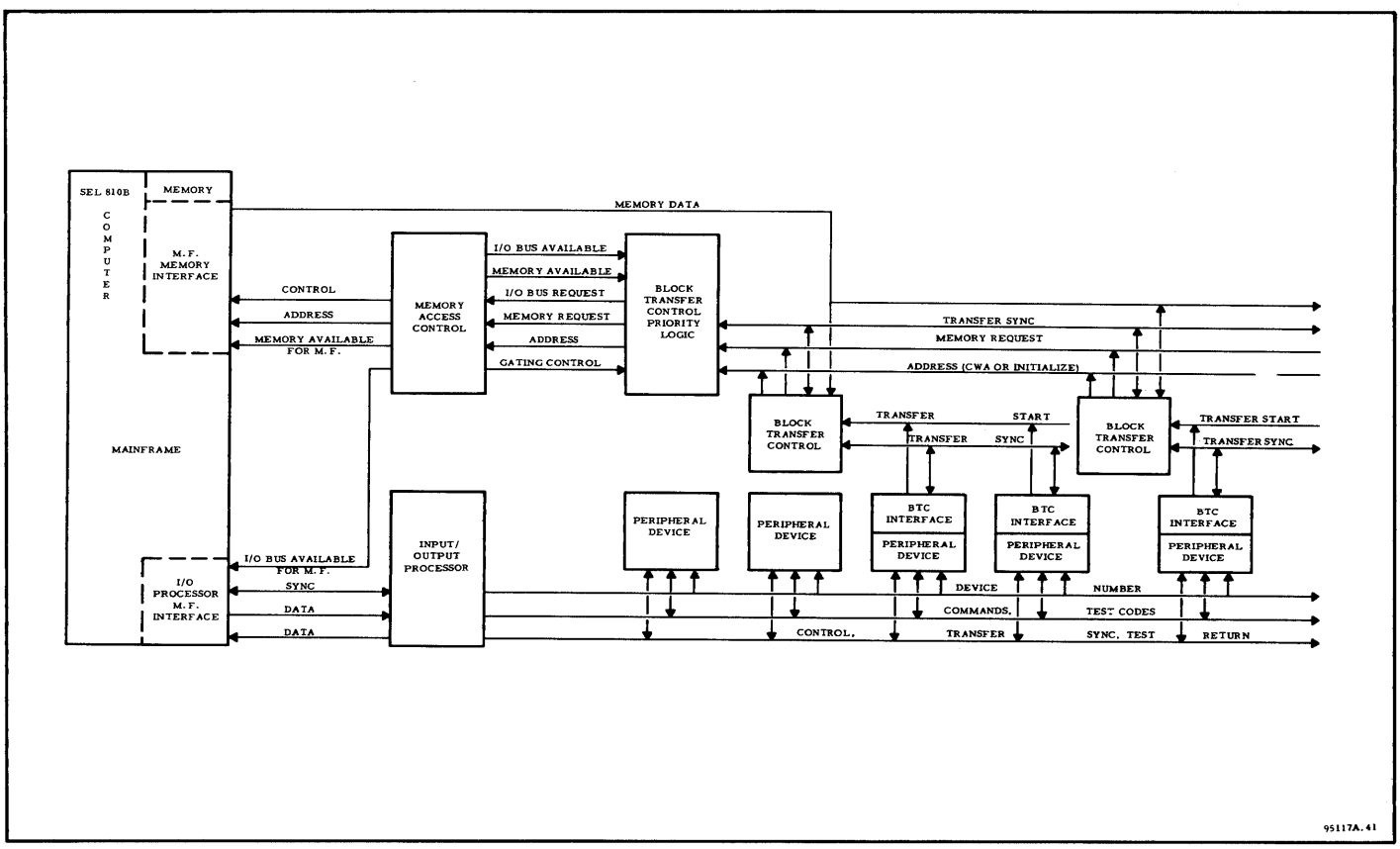


Figure 4-2. Input/Output Configuration and Computer Interface

The I/O Bus connects all peripheral devices to the I/OP in a "daisy-chained" manner, as shown in figure 4-3. The I/O Bus contains 16 data lines, six unit number lines, and numerous control lines.

The 16 data lines provide two-way communication paths. All data, CEU command words and TEU test words are transferred over these lines. Word-oriented units such as acquisition subsystems contain a full set of 16 cable drivers and terminators for the data lines. Character-oriented devices having character assembly buffers such as magnetic tape control devices also contain a full set of cable drivers and terminators. Character-oriented devices having character buffers such as paper tape punches and readers contain only eight to ten cable drivers and/or terminators. In this case, data commands and test codes are always received from the computer on the eight lines corresponding to computer bit positions 0-7. Some units also receive commands from bits 8 to 15. Single characters are always transferred to the computer on the data lines corresponding to bit positions 8-15. Characters having less than eight bits are right-justified in the eight-bit field. The data lines connected to each peripheral unit are defined in Section VI.

The six unit number lines connected to each unit permit up to 64 individual units to be addressed by the computer.

The control lines consist of the signals named in table 4-1.

These lines are used to enable I/O instructions to be executed in the following basic sequence (TEU differs).

(1) The computer initiates execution by sending out the device (unit) number contained in the instruction. The computer also sends out the Instruction Sync and instruction command (Data, Command, Test, Input/Output) signals.

(2) The addressed device responds by sending the Unit Sync Return and Unit Test Return signals to the computer.

(3) After recognizing the Unit Sync Return signal, the computer tests the Unit Test Return signal for the unit status ("Ready" to execute command or "Not Ready").

Table 4-1. I/O Control Signals

Signal	Computer Commands
Instruction Sync	ALL I/O INSTRUCTIONS
Data Instruction	AIP, AOP, MIP, MOP
Command Instruction	CEU
Test Instruction	TEU
Input/Output	AIP, AOP, MIP, MOP
Wait Flag	AIP, AOP, MIP, MOP, CEU
Unit Test Return	ALL I/O INSTRUCTIONS
Unit Sync Return	ALL I/O INSTRUCTIONS
Computer Data Here	AOP, MOP, CEU, TEU
Computer Data Accepted	AIP, MIP
Unit Data Accepted	AOP, MOP, CEU, TEU
Computer Clock	
Master Clear	

(4) If the device indicates "Ready", the data transfer is made. The Data Here and Data Accepted signals synchronize the transfer. For computer input transfer, the unit "Ready" signal also indicates "Data Here".

(5) After the transfer is completed, the computer tests the control lines from the device to insure that they have returned to the "Off" level. The next instruction is started in the following machine cycle.

The normal execution time for each immediate mode I/O instruction is four machine cycles. In addition, presence of the Wait Flag in an I/O instruction delays completion of instruction execution until the device indicates a "Ready". (The operation of the Wait Flag is described in Section II.)

The execution sequence is similar for all instructions except TEU. When a TEU instruction is executed, no "Ready" test is made before transfer of the test word. Transfer is made following recognition of the Unit Sync Return signal. The Test Return line is tested after the test word has been transferred to the device. The return signal is a particular unit status gated on the Test Return line by the value of the test word transferred to the device.

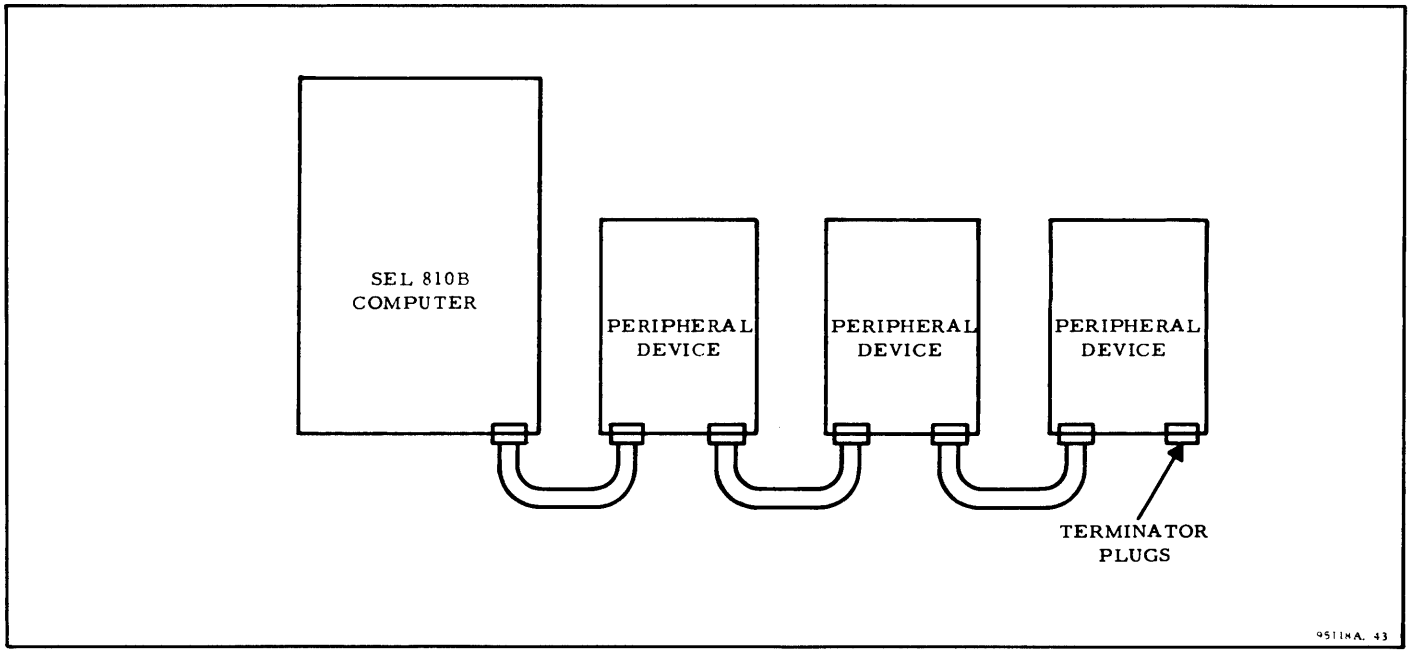


Figure 4-3. Peripheral Device Bus Connections

INPUT/OUTPUT BUS

The Basic 810B Computer is supplied with cable drivers and terminators which enable 16 units to be connected to the computer on one "daisy-chained" cable. Additional drivers, terminators and connectors are available if more than one cable chain is required.

BLOCK TRANSFER CONTROL UNIT

GENERAL CAPABILITIES

The SEL Block Transfer Control (BTC) unit is an optional computer input/output control unit which enables fully-buffered transfer of data between peripheral units and computer memory. The salient features of this unit are listed below:

Bits per Transfer	Full computer word
Maximum Words per Block	32,767
Maximum Transfer Rate	1,333,000 words per second
Memory Cycles Stolen per Transfer	One
Block Transfer Reinitialization	Automatic or program controlled
Maximum Number of BTC's per Computer	8
Maximum Number of CGP's per Computer	6
Maximum Number of Peripheral Devices per BTC	16

BTC OPERATION

The BTC contains two binary counters plus transfer initialization and synchronization logic. One of the counters stores the current word address (CWA) and the second stores the word count (WC). CWA defines the storage location for each word transferred to/from memory and WC defines the number of words to be transferred. The initial values for CWA and WC are obtained from two fixed locations in computer memory by the BTC each time a new block transfer is initiated (see table 5-2 for BTC memory location assignments). Each time a word is transferred between memory and the selected peripheral unit, CWA is incremented and WC is decremented. The block transfer is completed and an interrupt is generated when WC = 0. After a block transfer is completed, the BTC automatically initiates a new block transfer by obtaining a new initial set of CWA and WC values from the two dedicated memory locations. The block transfer sequence is ended by placing a terminate code in the WC word. The terminate code is a ONE in bit 0 (sign bit).

NOTE

The initial value of CWA is identified as the First Word Address (FWA). This allows the BTC starting address to be defined and distinguished from any other CWA.

The CWA value may be transferred from the BTC to the dedicated memory location by execution of a CEU command addressed to the device and containing a ONE in bit 13 of the command code. (See Appendix C.)

BTC Initialization and Data Flow

The BTC is initialized through the peripheral device to/from which the block transfer is to be made. Figure 4-2 shows the data and control path involved. The figure shows two peripheral devices connected to one BTC and a third peripheral device connected to a second BTC. These devices, as previously described, may communicate with the computer through execution of any of the I/O instructions. In addition, they may transfer data under BTC control, rather than under single-word program control.

Execution of the proper Command External Unit (CEU) instruction causes the device specified by the instruction to send an Initialize signal to the BTC to which it is cabled. In many peripheral devices, this instruction also causes the unit to initiate action to produce/accept data. When the BTC receives the Initialize signal from the device, it requests a memory cycle through the Memory Access Control (MAC). It also generates the address of the CWA memory location assigned to it. When the memory cycle is granted, the CWA value is transferred from the memory to the CWA counter in the BTC. A request for a second cycle is then made by the BTC and the address of the memory location containing WC is placed on the address lines by the BTC. When the second cycle occurs, WC is transferred from memory to the WC counter in the BTC. The terminate bit (bit 0) contained in the WC word is also tested and a latch is set if the terminate bit is a ONE, which signifies that no more block transfers are to be made after completion of the one being initialized. The maximum time for the entire initialization is four cycles for

the CEU execution, plus 2 cycles for the CWA and WC transfers, which occur immediately following CEU execution.

After BTC initialization, words are transferred between the selected peripheral device and memory over the I/O data lines under the joint control of the BTC, the BTC Priority Control and the MAC. A word transfer is initiated by the device which sends a Data Transfer Request line to the BTC. The Data Transfer Request signal causes the BTC to request a memory cycle through the MAC. When the MAC determines that the next cycle can be granted, a Memory Available signal is sent to the BTC. The BTC, in turn, sends a signal to the peripheral device which causes it to connect to the Unit I/O data lines, execute the data transfer, and then disconnect from the data lines. After completion of a word transfer, the CWA value is incremented and the WC value is decremented in the BTC counters. All words are transferred by repetition of this cycle, which is always initiated by the peripheral device.

When the value of WC is decremented to zero, the block transfer is terminated. If the terminate latch in the BTC had not been set by the terminate bit in the last WC word acquired from memory, a new block transfer is automatically initiated by the BTC. Re-initialization consists of acquiring new CWA and WC values from the memory locations assigned to the BTC. After re-initialization, an interrupt is generated which signifies that the transfer of the last block is completed and a new block transfer is initialized. The interrupt processing routine can then store in the dedicated locations the CWA and WC values for the next block transfer anytime prior to the completion of the current block transfer. This re-initialization technique reduces the problem of re-initializing block transfers under program control between the times of occurrence of two successive words in a continuous data stream.

If the terminate latch in the BTC had been set by the terminate bit in the last WC word acquired from memory, an interrupt is generated when the value of WC is decremented to zero and no new transfer is initialized by the BTC. In addition, the Data Transfer Request line from the peripheral signal is received. Hence, the BTC disconnects from the peripheral device.

BTC Priority and Timing

BTC's are granted memory cycle requests on a priority basis. The priority ordering function is performed by the BTC Priority Control. A unique priority is assigned to each BTC. The priority logic is structured similar to that of the interrupt priority logic, insuring that higher priority BTC's are always serviced before lower priority units. However, once a word transfer is initiated, it is not interrupted by a request from a higher priority BTC. In addition, BTC requests for memory cycles always take precedence over mainframe requests and can effectively "lock out" the mainframe if the peripheral transfer rate is high enough. BTC and P.I. Assignments are shown in table 5-3.

The maximum collective transfer rate for a BTC (or group of BTC's) is 1,333,000 words per second. Cycle stealing (or lockout) from the program is automatic and each BTC word transferred removes one cycle from the program. The BTC can gain access to the memory after a delay of one cycle except during the time of execution of the instructions listed in table 4-2 (the number of cycles refers to the number of consecutive cycles during which time the BTC cannot gain memory access). When these instructions are executed, the main program will hold out the BTC transfer for a maximum of the number of cycles indicated.

Table 4-2. Execution Times

Two Cycles	Four Cycles (or More)	
IMS	CEU	MOP
	AOP	MIP
	AIP	TEU

SECTION V
PRIORITY INTERRUPT SYSTEM

GENERAL DESCRIPTION

The SEL 810B Computer can have up to 98 individual levels of priority interrupts. Each level can be selectively enabled and disabled under program control except for two special interrupts (Power Fail Safe and Stall Alarm). Two standard levels plus the special power fail safe level are supplied with the basic computer. Additional, optional levels are available in groups of 12 levels each, except the first optional group contains ten levels.

Assignment of interrupts is highly flexible. Internal signals such as Overflow and Memory Parity can be connected to interrupt levels. BTC, End of Block signals and external signals from peripheral units and custom system components are connected to the levels which best fit the operation of each system.

Two special interrupts, Power Fail Safe/Auto Start and Stall Alarm, are provided. These levels are always enabled. Interrupt signals at these levels override all other computer functions, including Halt, I/O Wait and indirect address chaining. The Stall Alarm and Auto Start features are options described in Section VII.

A unique location in memory is assigned to each interrupt level. These locations are assigned in MAP 1 to keep the entire MAP 0 available for program usage. Location $1,002_8$ is assigned to the highest priority programmable level, location $1,003_8$ to the second highest level, etc. Tables 5-1 and 5-2 shows the assignment of interrupt locations and BTC locations.

When an interrupt signal is recognized by the mainframe, a wired-in instruction SPB*L (Store Place and Branch Indirect) is executed, where L is the address of the memory location assigned to the interrupt level. By storing the starting location of the interrupt processing routine in L, a linkage is provided to any point in memory. Since the address of the next instruction to have been executed in the interrupted program is stored in the interrupt routine entry point by the SPB instruction, a means for returning to the point of interrupt is provided. If the Program Protect and Instruction Trap option is included (and the computer is in the protected mode), the status of the Protect Latch is also stored in bit 0 of the interrupt routine entry point by the SPB instruction. When the program returns to the point of interrupt, the protect latch returns to the status present at the time of the interrupt.

The mainframe may be interrupted by a particular interrupt level provided that:

(a) the level has been previously enabled

(b) no higher level interrupt is active.

If a higher level interrupt is active when an interrupt signal occurs, the interrupt will be stored until the completion of execution of the higher level interrupt processing routine. The lower level routine will then be initiated. It will continue until completed or until interrupted by a higher level interrupt signal. In this case, the lower level routine will be completed after completion of the higher level routine. Program control will then be returned to the original point of interrupt. The priority logic enables any number of interrupt levels to be active at the same time. Routine execution is always performed in the order of priority of the active interrupts.

Table 5-1. Priority Interrupt Assignments

(Octal) Memory Location	Interrupt Assignment
1000	Power Fail Safe/Restore
1001	Stall Alarm
1002	PI Group, Level 0,1 (Highest)
1003	0,2
1004	0,3
1005	0,4
1006	0,5
1007	0,6
1010	0,7
1011	0,8
1012	0,9
1013	0,10
1014	0,11 (Std)
1015	0,12 (Std)
1022	1,1

Table 5-1. Priority Interrupt Assignments (Cont'd)

(Octal) Memory Location	Interrupt Assignment
1023	1, 2
1024	1, 3
1025	1, 4
1026	1, 5
1027	1, 6
1030	1, 7
1031	1, 8
1032	1, 9
1033	1, 10
1034	1, 11
1035	1, 12
1042	2, 1
1043	2, 2
1044	2, 3
1045	2, 4
1046	2, 5
1047	2, 6
1050	2, 7
1051	2, 8
1052	2, 9
1053	2, 10
1054	2, 11
1055	2, 12
1102	PI Group, Level 3, 1
1103	3, 2
1104	3, 3
1105	3, 4
1106	3, 5
1107	3, 6
1110	3, 7
1111	3, 8
1112	3, 9
1113	3, 10
1114	3, 11
1115	3, 12
1122	4, 1
1123	4, 2
1124	4, 3
1125	4, 4
1126	4, 5
1127	4, 6
1130	4, 7
1131	4, 8
1132	4, 9
1133	4, 10
1134	4, 11
1135	4, 12
1142	5, 1
1143	5, 2
1144	5, 3
1145	5, 4
1146	5, 5
1147	5, 6
1150	5, 7
1151	5, 8
1152	5, 9
1153	5, 10
1154	5, 11
1155	5, 12
1162	6, 1
1163	6, 2
1164	6, 3
1165	6, 4
1166	6, 5
1167	6, 6
1170	6, 7
1171	6, 8
1172	6, 9
1173	6, 10
1174	6, 11
1175	6, 12
1202	7, 1
1203	7, 2
1204	7, 3
1205	7, 4

Table 5-1. Priority Interrupt Assignments (Cont'd)

(Octal) Memory Location	Interrupt Assignment
1206	7, 5
1207	7, 6
1210	7, 7
1211	7, 8
1212	7, 9
1213	7, 10
1214	7, 11
1215	7, 12

Table 5-2. BTC Memory Location Assignments

Memory Location	BTC or Interrupt Assignment
1060	BTC 1 FWA
1061	1 WD CNT
1062	2 FWA
1063	2 WD CNT
1064	3 FWA
1065	3 WD CNT
1066	4 FWA
1067	4 WD CNT
1070	5 FWA
1071	5 WD CNT
1072	6 FWA
1073	6 WD CNT
1074	7 FWA
1075	7 WD CNT
1076	8 FWA
1077	8 WD CNT

DETAILED DESCRIPTION

The following paragraphs describe the priority interrupt system hardware and software in detail.

INTERRUPT CONNECTIONS

Two levels of interrupts are supplied with the basic computer. The priority levels of these interrupts are Group 0, Levels 11 and 12. The levels of the standard interrupts are placed at the bottom of Group 0 to make optional levels available of both higher and lower priorities.

The two standard levels are connected through the I/O cable to all peripheral units, as required. Interrupt signals in each unit are connected to one or the other of the two levels under program control.

Connection of a unit interrupt to a standard priority level is performed by execution of the CEU instruction. The format of the second word of the CEU instruction for the peripheral units contain three bits used to connect and disconnect the standard unit interrupts. The interpretation of the three bits is given in table 5-3.

Table 5-3. Standard Interrupt CEU Bit Functions

Bit	Function
1 = ONE	Connect Levels Designated in Bits 2 and 3
2 = ONE	Connect/Disconnect Level 11, Group 0
3 = ONE	Connect/Disconnect Level 12, Group 0
1 = ZERO	Disconnect Levels Designated in Bits 2 and 3
2 = ZERO	Leave Level 11, Group 0 in Present State
3 = ZERO	Leave Level 12, Group 0 in Present State
Refer to Appendix C	

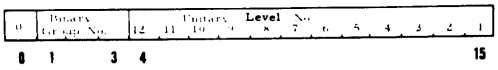
INTERRUPT ENABLING / DISABLING

Interrupt levels can be selectively enabled and disabled, one group (up to 12 levels) at a time, by the two instructions:

PIE (Priority Interrupt Enable)

PID (Priority Interrupt Disable)

The second word of these two-word instructions has a three-bit group field and a 12-bit level field which designate the group and the one to twelve levels within the group to be affected by the instruction. The bit assignment is shown below:



The group field is binarily coded, group 00 being the highest priority group. The level field is unitarily coded, a ONE in bit 15 is signifying the

highest priority level within a group (level 1). An instruction which will cause the five highest levels in group TWO to be enabled is written in assembly language as:

```
PIE
DATA      '20037
```

Execution of this instruction leaves the seven lower levels within group TWO (if present) unaffected. They remain either enabled or disabled.

INTERRUPT LEVEL LOGIC

The logic for each interrupt level consists of three latches and associated gates. The latches are designated Request (R), Enable (E) and Active (A). The R latch is set by the external interrupt signal. It provides storage for the signal until the level becomes active and the interrupt routine is executed. The R latch is reset by execution of the two instructions which normally terminate each interrupt routine:

```
TOI       Turn Off Interrupt
BRU*      Unconditional Branch Indirect
```

The R latch is also reset for all designated levels each time the PID instruction is executed.

The E latch is set by the proper group and level bits in the PIE instruction. It is reset by the same bit pattern in the PID instruction.

The logic expression for the signal (In) which interrupts the mainframe from interrupt level n is

$$In = Rn\ En\ \overline{(A_1 + A_2 + \text{---} + An\text{-}1)}$$

where A_1 in the "active" signal for the highest interrupt level. The mainframe tests for the presence of any I signal after fetching each instruction from memory, but prior to execution. If In is present, An is set and the instruction

$$SPB^* \ 1001_8 + n \ (n < 46),$$

is executed in two machine cycle times.

The A latch is reset by the signal generated by the execution of the TOI, BRU* instruction pair. Each time this instruction pair is executed, both the highest level A and R latches are reset, releasing the channel to accept a new interrupt signal.

A special interrupt card is available which provides an alternate means of effectively disabling interrupts. When this card is inserted at any interrupt level, the level can be made active under program control by execution of a PIE instruction specifying that level. Making the special level active inhibits any lower level from becoming active. Therefore, the lower levels are effectively disabled. Execution of the PID instruction addressing the special level causes the active (A) latch for that level to be reset.

If the special interrupt card is inserted at the highest level (Group 0, Level 1), a means is provided for keeping higher priority levels from becoming active while lower priority levels are being processed. An interrupt processing routine at any level can be executed without interruption by having a PIE 1,0 immediately after the entry location and a PID 1,0 immediately before the exit (BRU*) location.

INTERRUPT ROUTINE PROGRAMMING

The Assembly language coding for the typical interrupt routine is shown in table 5-4.

Table 5-4. Sample Assembler Interrupt Routine

Loc.	Operation	Address	Comments
INT4	ZZZ	***	Storage Location for Return Address
	(Interrupt Processing Routine)		
	TOI		Turn off Interrupt
	BRU*	INT4	Return to Point of Interrupt

Since an interrupt can occur at any time during the execution of the program, the interrupt routine has no method of determining which registers are being used by the interrupted program. Therefore, the interrupt routine must save and restore any registers which it uses, in order to allow the interrupted program to continue upon return: Example:

Loc.	Operation	Address	Comments
INT4	ZZZ	**	
	STA	ASAV	SAVE A-Acc
	STB	BSAV	SAVE B-Acc
	(Interrupt Processing Routine)		
	LAA	ASAV	Restore A-Acc
	LBA	BSAV	Restore B-Acc
	TOI		
	BRU*	INT4	

There are five instructions that do not allow an interrupt to be serviced until the next instruction is executed:

TOI	(Turn off Interrupt)
PID	(Priority Interrupt Disable)
PIE	(Priority Interrupt Enable)
SPB	(Store Place and Branch)
CSB	(Copy Sign of B-Acc.)

The TOI instruction inhibits servicing the interrupt for one instruction to allow the BRU* to be executed at the exit of the interrupt routines. This is to insure that the proper active latch (A) is reset.

The PID is included to allow the execution of a PIE immediately following the PID in order to reset any possible requests on the selected levels before enabling them. The inclusion of the PID also permits the disabling of higher level interrupts as the first instruction of an interrupt routine. However, this should be used with caution, because the PID instruction resets the request latches (R) on its second cycle. The duration of an interrupt signal must be greater than the longest non-interruptable instruction chain to guarantee that the interrupt is not erased when a PID instruction is executed. If this condition cannot be met, the special interrupt card (described later) is recommended.

When it is desired to use the same subroutine at more than one interrupt level, it must be allowed to complete the execution for the lower levels or be made re-entrant. The following example shows a method to allow the lower levels to complete.

```
SUBR    ZZZ     **
        PID             Disable all higher
        DATA    XYZ     levels

(Subroutine)

        PIE             Enable same levels
        DATA    XYZ
        BRU*    SUBR
```

The fact that the SPB instruction, used to enter the subroutine, and the PIE, used immediately preceding the BRU* are non-interruptable, allows the routine to complete before a possible re-entry. The SPB instruction being non-interruptable also allows the coding of re-entrant subroutines, in that the exit location can be saved before an interrupt (which can also use the routine) can occur.

The CSB instruction is non-interruptable to allow the execution of an AMA, SMA, or NEG immediately following the CSB. This is necessary because the CSB instruction must always be followed by an AMA, SMA, or NEG instruction because the carry latch is reset at the completion of execution of all instructions except CSB.

When multiple interrupt signals are connected to the same interrupt level, a certain amount of testing is necessary to determine the source of the interrupt. The following example in figure 5-1 shows the use of two typewriters on the same standard output interrupt level.

The first portion of the coding is executed at a lower priority level.

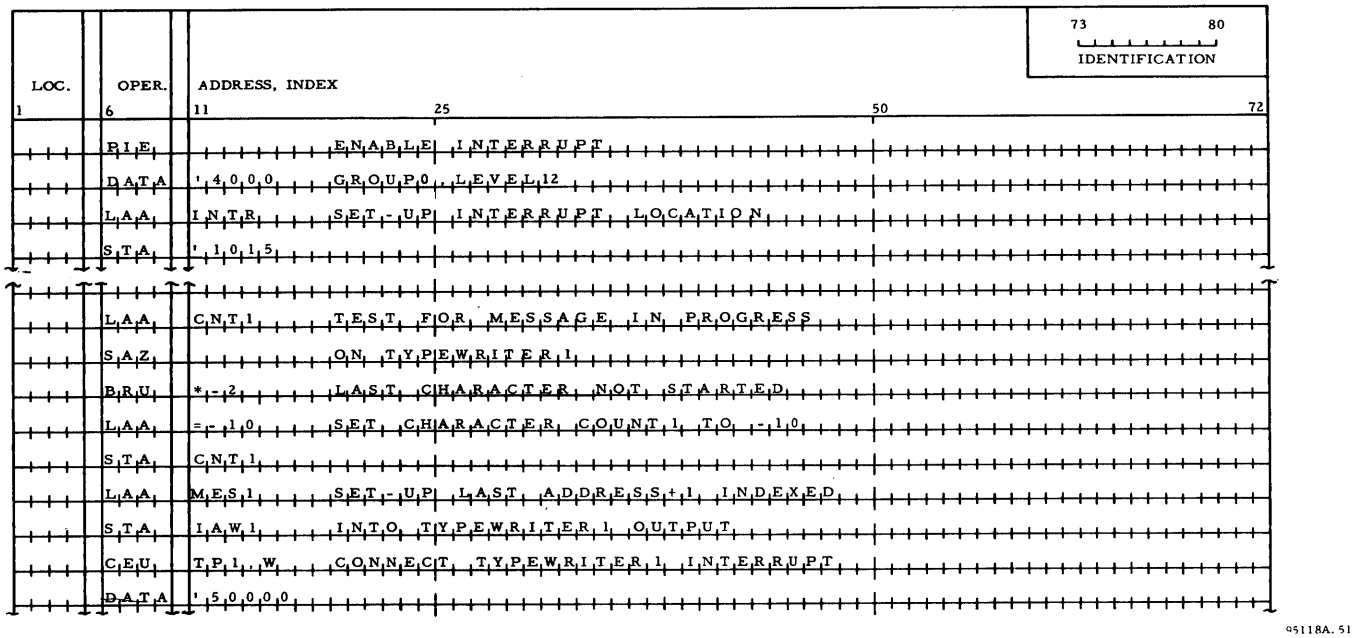


Figure 5-1. Sample Program for Two Typewriters on the Same Standard Output Interrupt Level

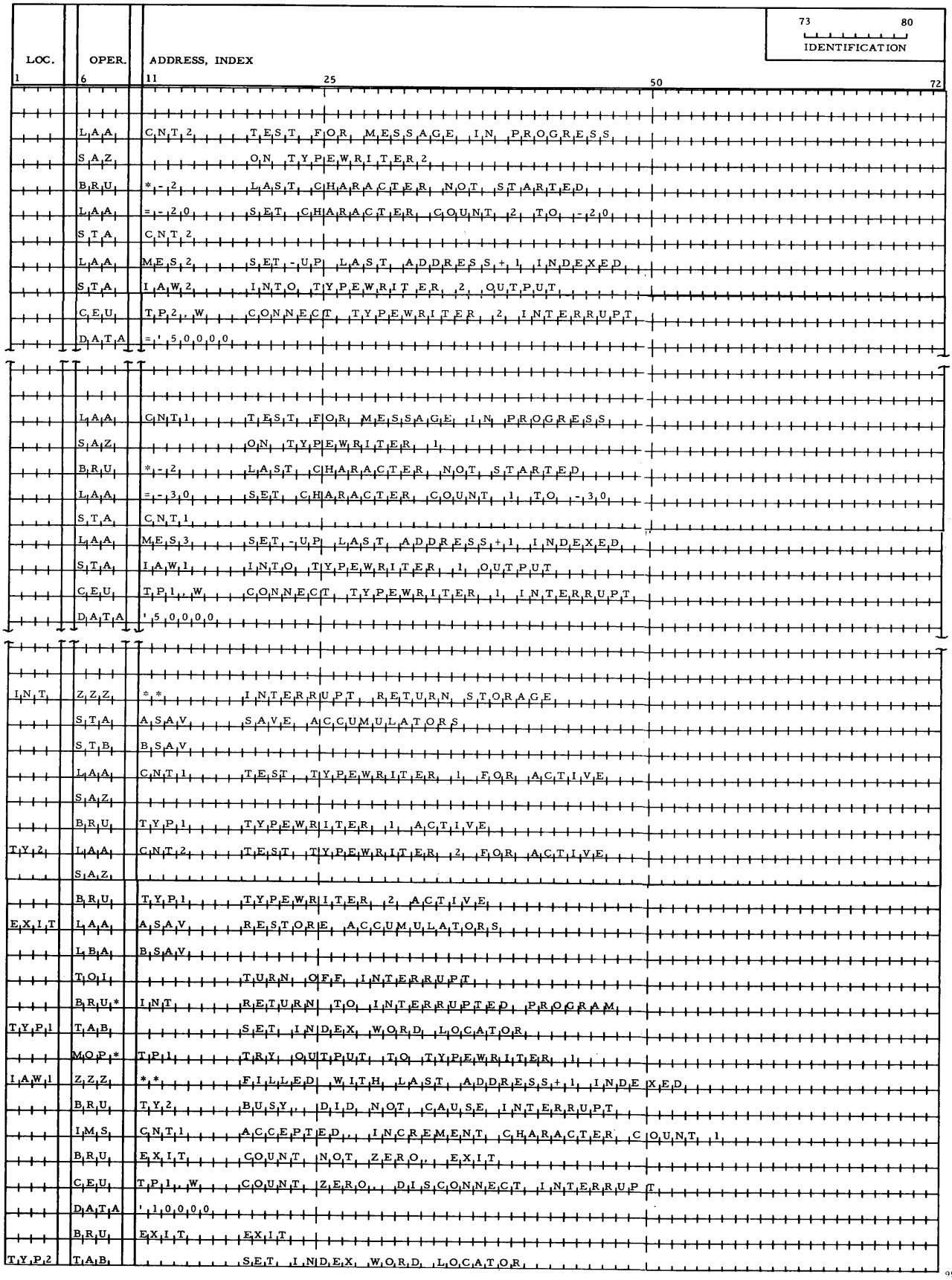
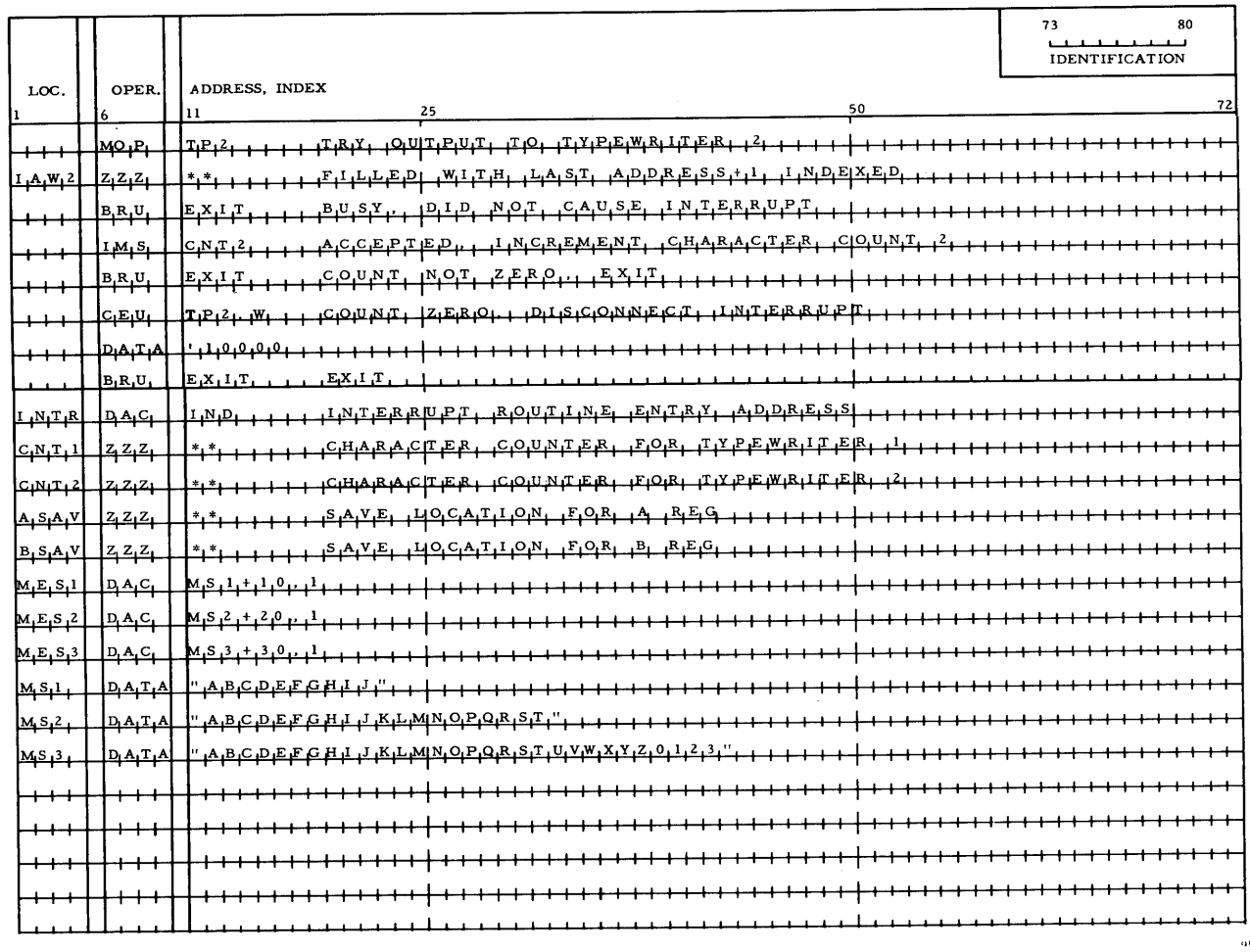


Figure 5-1. Sample Program for Two Typewriters on the Same Standard Output Interrupt Level (Cont'd)



05118A. 51

Figure 5-1. Sample Program For Two Typewriters on the Same Standard Output Interrupt Level (Cont'd)

SECTION VI
PERIPHERAL DEVICES

INTRODUCTION

This section contains brief descriptions of the standard peripheral devices available with the SEL 800 series of computers. Although part of each description consists of physical characteristics, the primary emphasis is on the programming aspects of the equipment.

CONSOLE TYPEWRITER (MODEL NO. 81-711-02A DEVICE NO. 1)

The standard console typewriter (Model 81-711-02A) provided with the 810B Computer is a modified Teletype Model ASR-33 send/receive typewriter. The modifications introduced by Systems Engineering Laboratories include the interface and control electronics that make the basic device more flexible when used with the computer.

The unit responds to the following commands:

a. Select Reader Mode

b. Select Keyboard Mode

c. Reader and Keyboard Off

The Model 81-711-02A consists of a typewriter keyboard and printer plus an eight-level paper tape punch and reader. It is supplied as standard equipment with 810B Computers and is mounted on a stand adjacent to the computer. A manual switch is provided to enable either on-line or off-line operation. Specifications for the Model 81-711-02A are defined in table 6-1.

Table 6-1. Model 81-711-02A Console Typewriter Specifications

Characteristics	Specification
Paper Tape Input Speed	20 characters per second
Output Speed	10 characters per second for print and punch
Code	ASCII

Table 6-1. Model 81-711-02A Console Typewriter Specifications (Cont'd)

Characteristics	Specification
Number of Printable Characters	63
Characters per Line	72
Paper Tape	Standard 1-inch roll
Dimensions	22 inches wide x 35 inches high x 18 inches deep including stand
Power	115 VAC, 60 cps, single phase. dc voltages required by the interface and control electronics are supplied by the computer power supplies.

ASR-33 PROGRAMMING

When the ASR-33 is operated on-line, the input devices (keyboard and reader) operate separately from the output devices (printer and punch). As a result, the paper tape reader can operate independently at the rate of 20 characters/second. If printing and punching of input characters are desired, each character read into the computer is transferred back to the output devices under program control. Both printing and punching occur unless the punch is turned off manually. Printing and punching can be performed at rates up to 10 characters/second.

The ASR-33 coupler contains two character buffers, one for input and one for output. The presence of two buffers plus the separation of the input and output devices enables both input and output data transfers to occur, both at the maximum rate. For example, a paper tape can be read at 20 characters/second while an independent set of characters is printed at 10 characters/second. The presence of the buffers in the ASR-33 also results in the computer and I/O bus being "busy" for only four machine cycles each time a character is transferred between the computer and the ASR-33.

The two standard peripheral unit interrupt levels can be connected in the associated buffer by program

control. Each time the input buffer, which stores characters from the keyboard and reader, receives a new character it generates an interrupt at Group 0, Level 11. Each time the output buffer, which stores characters for printing and punching, is emptied it generates an interrupt at Group 0, Level 12.

The CEU commands sent to the ASR-33 select the desired input mode and connect/disconnect the two interrupts. The single output mode is selected by the on-line switch and requires no CEU command execution. As a result, any time an AOP or MOP instruction having a unit 1 address is executed, a character will be printed. The character will also be punched if the punch is turned on.

Bit coding for the CEU second word function codes are shown in table 6-2. Bit numbering corresponds to the positions (0-15) in a computer word.

Table 6-2. Bit Coding for the CEU Command

Function	Bit (=1)
Priority Interrupt Connect	1
Computer Input Interrupt	2
Computer Output Interrupt	3
Reader Mode	4
Keyboard Mode	5
Mode Clear	6

Bits 1-3 permit the two ASR-33 interrupts to be selectively connected to the two standard peripheral unit interrupt levels. The input interrupt, used with keyboard and reader entry, is connected to PI Level 11, Group 0 by execution of the instruction:

```
CEU     1
DATA    '60000
```

It is disconnected by execution of the instruction:

```
CEU     1
DATA    '20000
```

The corresponding instructions which connect and disconnect the output mode interrupt to PI Level 12, Group 0 are:

```
Connect      CEU     1
             DATA    '50000

Disconnect   CEU     1
             DATA    '10000
```

Execution of this instruction causes an interrupt to be generated immediately so that the first as well as all succeeding characters can be transferred by the same interrupt processing routine. The connect commands must be preceded by a PIE instruction to enable the computer to be interrupted.

Execution of the input mode select commands, which contain bits 4, 5, or 6 cause the input buffer to be cleared as well as the specified input unit to be selected. In transferring between reader and keyboard modes, it is not necessary to mode clear. This mode clear signal allows turning both modes off.

Reader or keyboard operation is controlled by "select reader mode" or "select keyboard mode" bits in the CEU instruction. The reader control operates on an automatic character call-up philosophy. As a result of each AIP or MIP instruction, a data character is transferred to the computer and action is initiated to start calling up the next character on the tape. This feature simplifies the programming by turning the reader on and off (with CEU instructions) only once for each group of characters desired.

NOTE

Each time a character is read, the tape advances, and the character cannot be read again. The programmer must insure that a CEU to clear the reader mode ("select keyboard mode", or "select mode clear") and turn the reader off is generated within 10 ms after each desired group of characters is accepted by the computer. Otherwise, the next character on the tape will be read, stored in the input data buffer, and subsequently lost when the buffer is cleared or loaded with new data.

No test unit (TEU instruction) hardware is required on the ASR-33 coupler. A test for "busy" is built in each instruction. It can become desirable to perform this test without attempting to command the ASR-33 coupler. To accomplish this, an I/O instruction to unit one, with no bits set in the second word, will skip when not busy without changing the unit.

For the most part, the input section and the output section operate completely independently under one unit number. For reference, the busy times are approximately as follows:

Reader	50 milliseconds*
Keyboard	100 milliseconds
Output Devices	100 milliseconds

*First 10-12 milliseconds of each cycle left open to allow turn-off.

The reader control switch is not used for on-line operation and can be left in either the Start or the Neutral position while running.

Useful Keyboard Features are as follows:

a. Here is Key - Punches all zeros. Can be used to generate leader or spaces for off-line operation. Recommended way of pulling tape into punch. Will not input into computer.

b. Rub Out Key - Punches all ones.

c. Repeat Key - Repeats characters as long as a character key and the repeat key are depressed.

d. Line Feed, Carriage Return Keys - Self explanatory.

e. Control Key - Used to generate special functions (such as WRU, BELL and TAB) which appear on the top portion of the character keys. The control (CTRL) key must be held down to transmit these functions when depressing the character keys.

NOTE

The bell is a convenient, audible alarm that can be generated by the program.

The correspondence between computer bit positions and paper tape levels is shown in the data flow diagram (figure 6-1).

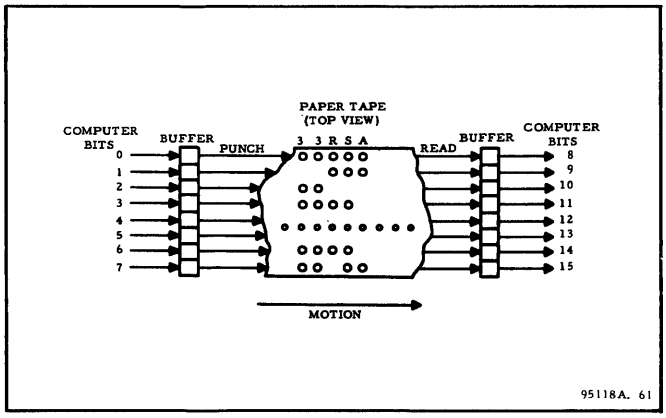


Figure 6-1. Paper Tape Data Flow Diagram

A programming routine for keyboard input and printer output is shown in table 6-3. This routine inputs, packs and stores a pair of characters. It also prints the characters as they are typed in.

Table 6-3. Programming Routine for Console Key-Board Input and Console Printer Output

Operation	Address	Comments
CEU	1	Select Keyboard
DATA	'2000	
BRU	*-2	Not Ready
AIP	1	Input Character
BRU	*-1	Not Ready
LSL	8	Shift to Output
AOP	1	Print Keyed Character
BRU	*-1	Not Ready
AIP	1,R	Input Character, Merge
BRU	*-1	Not Ready
STA	WORD	
LSL	8	
AOP	1	Print Keyed Character
BRU	*-1	Not Ready

NOTE

This routine is interruptable because no wait flags are used.

PAPER TAPE READER (MODEL NO. 81-510A, 300 CPS— DEVICE NO. 2)

The SEL Model 81-510A series paper tape reader reads eight-level one-inch paper or mylar tapes at synchronous speeds of up to three hundred characters per second in the forward direction. The paper tape reader responds to the following commands:

a. Reader Enable

b. Reader Disable

Specifications for the Model 81-510A Paper Tape Reader are defined in table 6-4.

Table 6-4. Model 81-510A Paper Tape Reader Specifications

Characteristic	Specifications
Speed	300 characters per second
Levels	8
Operation	Asynchronous start/stop
Size	19-inches wide x 7-inches high x 24-inches deep
Sensing	Photocell
Drive	Pinch roller
Power	115 volts, 60 cps ± 10%, 1.4 amp. nominal
Temperature	10° C to 35° C
Controls	Power on/off Run/load

PAPER TAPE PUNCH (MODEL NO. 81-520A, 110 CPS— DEVICE NO. 2)

The SEL Model 81-520A Paper Tape Punch punches eight-level paper or mylar tapes at speeds up to 110 tape characters per second. A sprocket hole is punched with each character. Punch power may be turned on or off under computer program control.

The unit responds to the following commands:

a. Turn punch, tape feed power on

b. Turn punch, tape feed power off

Specifications for the Model 81-520A Paper Tape Punch are defined in table 6-5.

Table 6-5. Model 81-520A Paper Tape Punch Specifications

Characteristic	Specifications
Speed	110 characters per second
Levels	8
Operation	Asynchronous start/stop

Table 6-5. Model 81-520A Paper Tape Punch Specifications (Continued)

Characteristic	Specifications
Size	19-inches wide x 14-inches high x 24-inches deep
Punches	High-carbon steel pins
Drive	Sprocket
Power	115 volts, 60 cps ± 10%, 2 amps. nominal
Temperature	10° C to 35° C
Controls	Power on/off and tape feed

Other features include built-in tape storage reel, tape cutter and transparent chad box.

PERFORATED PAPER TAPE SPOOLER (MODEL NO. 80-530A)

The SEL Model 80-530A Paper Tape Spooler supplies and spools 5, 6, 7 or 8 level paper tapes at rates up to 400 characters per second. The tape is rewound by a manually controlled switch at a 1000 characters per second rate.

No command or test functions are required from the computer. Specifications for the Model 80-530A paper tape spooler are defined in table 6-6.

Table 6-6. Model 80-530A Paper Tape Spooler Specifications

Characteristic	Specifications
Feed Speed	40 inches per second
Rewind Speed	100 inches per second
Tape Channels	5, 6, 7 or 8 levels, interchangeable
Reel Hubs	Standard NAB reel dimensions
Reel Diameter	8-inch O. D.
Reel Capacity	400 feet of 4-mil tape
Interlock	Tape break or no tape
Mounting	Any upright position
Size	19-inches wide x 10-1/2 inches high x 8-5/8-inchess deep

Table 6-6. Model 80-530A Paper Tape Spooler Specifications (Cont'd)

Characteristic	Specifications
Power	115VAC ± 10%, 60 cps, single phase
Temperature	5° C to 45° C
Controls	Power on/off Rewind

HIGH SPEED PAPER TAPE PUNCH/READER SYSTEM (MODEL NO. 81-525A—DEVICE NO. 2)

The paper tape system consists of a high-speed photoelectric reader and a high-speed punch. The photoelectric reader is capable of reading 6, 7, or 8-level paper tape at rates of 300 characters per second. The punch punches 8-level tape at a maximum rate of 110 characters per second.

Since both the punch and the reader are given the same unit number, they can be considered, from the software standpoint, as one unit with both input and output capabilities. Since they each have a separate buffer, they can be operated at maximum speed simultaneously.

There are two CEU commands that can be given the paper tape reader. These commands are "Reader Enable" and "Reader Disable". When the reader enable command is issued, the buffer will be filled with a character and the tape will advance one character position. This will occur only if the buffer is initially clear. While the reader is enabled, the tape will advance one position and the buffer will be filled each time an AIP or MIP is serviced by the reader. When the reader disable command has been issued, the reader will not advance when an AIP or MIP is serviced.

There are two CEU commands that can be given to the punch. They are "Punch Feed Power ON" and "Punch Feed Power OFF". The punch feed power on command must be given before transferring data to the punch. After punch tape feed power has been turned on, the punch will punch the tape and advance one character position each time an AOP or MOP is serviced.

There are two standard interrupts furnished with the Systems Engineering Laboratories high-speed paper tape system. One is the "Buffer Ready" interrupt from the punch and the other is the "Buffer Ready" interrupt from the reader. A routine for copying paper tape is given in table 6-7.

Table 6-7. Programming Routine for Copying Paper Tape

Operation	Address	Comments
CEU	2, W	
DATA	'5000	Punched Feed Power On/ Reader Enable
AIP	2, W	Input Character from Reader
LSL	8	
AOP	2, W	Output Character to Punch
BRU	*-3	

MAGNETIC TAPE (MODEL NO. 80-615 SERIES—DEVICE NO. 6 AND 7)

The SEL Model 80-615A magnetic tape transports use 1/2-inch mylar tape and have either 7 or 9 tracks. Specifications for the transports are given in table 6-8. The 7-track units are IBM 729 compatible and the 9-track units are IBM 2400 compatible. The tape control unit (unit 6 or 7) acts as a coupler for up to eight magnetic transports. Any standard SEL magnetic tape unit can be coupled to the computer by the tape control unit (TCU).

The control panel for the TCU contains the following indicators:

a. CRC Error (option on 9-track system only)

b. Ready

c. Parity Error (Lat/Long)

d. Read/Write Status

e. Busy

f. End-of-Tape

g. Load Point

h. Density (200/556/800)

i. Binary Mode

Table 6-8. Model 80-615 Magnetic Tape Transport Specifications

Model	Tape Speed	Start or Stop Time*	Minimum Gap Spanning Time *+	
			7 Track	9 Track
615-07	45 ips	9 msec.	25.4 msec.	21.7 msec.
615-09	75 ips	6 msec.	16.4 msec.	14.2 msec.
615-11	120 ips	3.8 msec.	10.2 msec.	8.9 msec.
615-12	150 ips	3.5 msec.	9.2 msec.	8.1 msec.

*Nominally +20%
+Time between the writing of the last character in one record to the writing of the first character in the next record.

j. BCD Mode

k. EOF

l. Characters/Word

m. Transport Number Selected

The control panel for the TCU also contains a three-position switch that allows the characters in the Write, Read or LRCC registers to be displayed.

MAGNETIC TAPE PROGRAMMING

CEU Instruction

In order to allow optimum programming of the magnetic tape system, a quasi-unitary bit assignment is used for the CEU data words. In order to represent all of the functions, two data word formats, called FORMAT 0 and FORMAT 1, are used. The desired format is selected by placing a 0 or a 1 in bit position 4 of the data word.

The quasi-unitary bit assignment allows more than one set-up or command function to be executed with one CEU instruction provided the selected bits are logical and do not contradict each other.

Format 0, CEU

The first tape operation command must be a CEU instruction using the format 0 data word which includes the set-up bits to select BCD or Binary tape density (200, 556, 800), transport number (0-7) and characters per word. These set-up bits must be entered in every CEU FORMAT 0 data word. Control bits are also provided to enable or disable one or both selectable interrupts, to rewind the tape, to erase four inches of tape, and to input the current word address of the BTC.

Table 6-9 lists the bit functions for the second word of the format 0, CEU command.

Table 6-9. CEU, Format 0, Second Word, Bit Functions

Bits	Function
0	Must be ZERO.
1 = ONE	Connects the interrupt selected with bits 2 and 3. If bits 2 and 3 are ZERO, bit 1 is ignored.
1 = ZERO	Disconnects the interrupt selected with bits 2 and 3. If bits 2 and 3 are ZERO, bit 1 is ignored.
2 = ONE	Selects word transfer ready interrupt for enable or disable.
3 = ONE	Selects end of record interrupt for enable or disable.
4 = ZERO	Selects the FORMAT 0 CEU data word.
*5 = ONE	Rewinds selected tape transport. Bits 10-12 must contain desired transport number.
*6 = ONE	Erases four inches of tape.
7 = ONE	Sets-up the TCU to transfer BCD data with even parity.
7 = ZERO	Sets-up the TCU to transfer Binary data with odd parity.

Table 6-9. CEU, Format 0, Second Word, Bit Functions (Cont'd)

Bits	Function
8 & 9	Selects tape density 00 $_2$ - 200 bpi 01 $_2$ - 556 bpi 10 $_2$ - 800 bpi
10-12	Selects tape transport (0-7)
13 = ONE	Inputs current word address of BTC into the corresponding dedicated locations.
14 & 15	Selects characters per word 01 $_2$ - 1 char/word 10 $_2$ - 2 char/word 11 $_2$ - 3 char/word 00 $_2$ - 4 char/word
*Bits 5 and 6 cannot both be ONE in the same CEU Format 0 instruction (only logical bit combinations should be selected).	

Table 6-10. CEU, Format 1, Second Word, Bit Functions

Bits	Function
0 = ONE	Initializes Block Transfer. (Must not be used if Bit 13 is ONE).
1 = ONE	(Same as Format 0)
1 = ZERO	(Same as Format 0)
2 = ONE	(Same as Format 0)
3 = ONE	(Same as Format 0)
4 = ONE	Selects the Format 1 CEU data word.
**5 = ONE	Write Record - the tape will move forward as the information being sent from the computer replaces the previous information on the tape. When this information ceases to come from the computer, a longitudinal check character is written and a record gap is generated. In the case of a 9 track system, the CRC character is written 4 character times previous to the LRCC character and follows the last data character by 4 character times.
**6 = ONE	Approximately 3-1/2 inches of tape are erased and an EOF mark is written.

Tape Format

Figure 6-2 shows the magnetic tape format for a format 0 data word.

Format 1, CEU

The Format 1 CEU data word is completely unitary. Care must be taken to assure that only logical bit combinations are selected. Table 6-10 lists the bit functions for the second word of the format 1, CEU command.

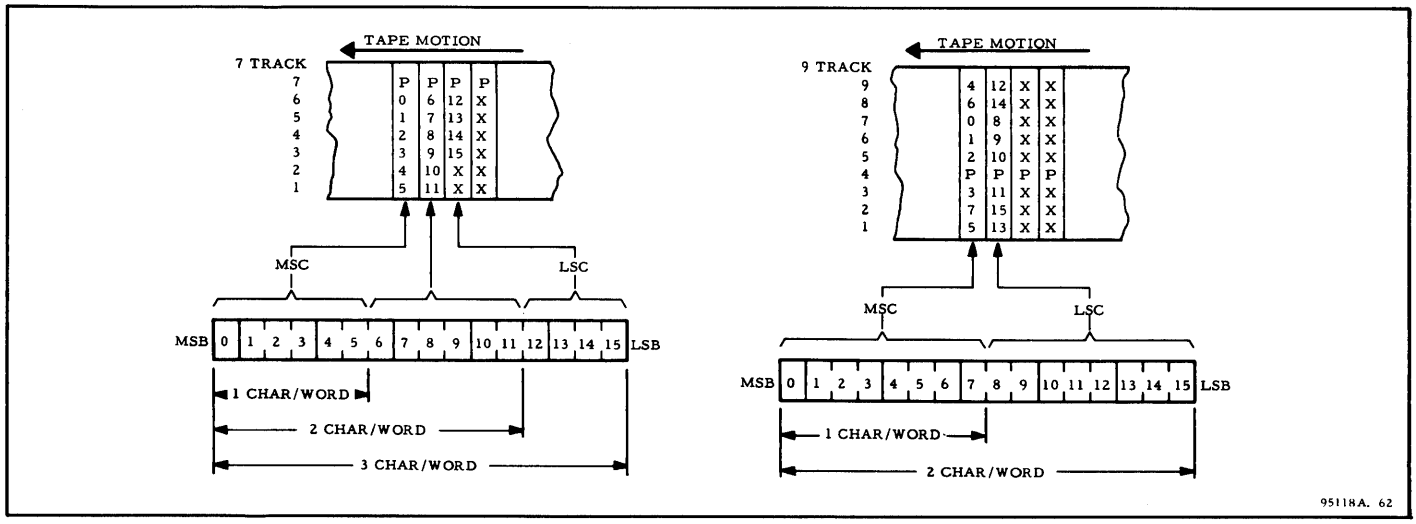


Figure 6-2. Magnetic Tape Format 0 Data Word

Table 6-10. CEU, Format 1, Second Word, Bit Functions (Cont'd)

Bits	Function
**7 = ONE	Read Record - the tape will move forward to the next record gap leaving the data on the tape undisturbed. When the tape is in motion, the data is transferred to the computer as it is encountered on the tape. This transfer will continue to take place until the next record gap is reached or until the computer stops requesting the data. If the computer stops requesting the data before the next record gap is encountered, a data overflow condition exists. This condition can be tested. A read operation should not follow a write, erase four inches of tape or write EOF operation. *
**8 = ONE	Advances tape forward one record, leaving the read/write heads in the middle of the next record gap. The data on the tape is undisturbed.
**9 = ONE	Advances tape one file, leaving the read/write heads in the middle of the record gap following the next end of file mark. The data on the tape is undisturbed. *
**10 = ONE	Backspaces one record, leaving the read/write heads in the middle of the previous record gap. The data on the tape is undisturbed.
**11 = ONE	Backspaces one file, leaving the read/write heads in the gap preceding EOF mark. The data on the tape is undisturbed.
12	Not Used
13 = ONE	Inputs current word address of BTC into the corresponding dedicated location (Must not be used if bit 0 is at ONE).
14 & 15	Unused

*These commands should not follow an erase four inches of tape, write record or write end of file operation. This is not a hardware restriction, but if one of these commands is executed and there is no more data on the tape, the transport will run all of the tape off of the reel unless it is manually halted.

**Only one motion command, represented by bits 5 through 11 in Format 1 can be used in a single CEU data word. Two or more motion commands in the same instruction are contradictory.

At the termination of an "erase four inches of tape", "write record", or "write end-of-file" operation the TCU is left in write status. All other operations will leave the TCU in read status.

TEU Instruction

This instruction is used to query the status of a given unit. With the magnetic tape system all status conditions are reset by every motion command issued to the TCU. The transport tested is always the one that is set up at the time the TEU instruction is executed. The TEU codes are set up to skip on the "GO" condition.

The TEU data word is completely unitary, in that each status condition has a single bit assignment. If more than one status condition is queried with a single TEU instruction, any one status condition being a "NO GO" will inhibit the instruction from skipping. The tests that may be initiated by the TEU instruction are listed in table 6-11.

Table 6-11. TEU, Second Word, Bit Functions

Bits	Function
0 = ONE	Skip On Not Busy - When the TCU is capable of executing a motion command, the next instruction is skipped. If the TCU is not capable of executing a motion command, the next instruction is executed.
1 = ONE	Skip On No End-Of-File - When no EOF status is present, the next instruction is skipped. When an EOF

Table 6-11. TEU, Second Word, Bit Functions (Cont'd)

Bits	Function
	status is present the next instruction is executed.
2 = ONE	Skip On No Overflow - When no overflow status is present, the next instruction is skipped. When an overflow status is present, the next instruction is executed. An overflow occurs when the data request from the computer is dropped before an end-of-record gap is reached on the tape. (Occurs only when reading magnetic tape.)
3 = ONE	Skip On Load Point - If the read/write heads are positioned at load point the next instruction is skipped. If the read/write heads are not positioned at load point, the next instruction is executed.
4 = ONE	Skip On End-Of-Record Interrupt - If the magnetic tape end-of-record causes an interrupt, the next instruction is skipped. If the magnetic tape end-of-record did not cause an interrupt the next instruction is executed.
5 = ONE	Skip On No Parity Error - When no parity error status is present the next instruction is skipped. When a parity error status is present, the next instruction is executed.
6 = ONE	Skip On Write Ring In - When the write ring (file protect ring) is in the reel, the next instruction is skipped; when the write ring is not in the reel, the next instruction is executed. The absence of the write ring prevents the destruction of data on tapes, such as library tapes. The write ring must be in the reel in order to write on the tape.
7 = ONE	Skip On No End-Of-Tape - If the end of tape marker has not been sensed, the next instruction is executed. The tape transports will not stop when the end-of-tape is sensed.

Table 6-11. TEU, Second Word, Bit Functions (Cont'd)

Bits	Function
8 = ONE	Skip On Rewinding - If the transport that is selected is mechanically rewinding, the next instruction is skipped. If it is not rewinding, the next instruction is executed. (When a rewind command is issued to the TCU, the rewind status does not exit until the mechanical motion has started. Therefore, before disconnecting a tape transport which has been given a rewind command, it is necessary to test for the rewind status. Once the mechanical motion has started, the tape transport can be deselected and the rewinding will continue while another tape transport is being commanded.)
9 = ONE (9-Track only)	Skip On No CRC Error - If no cycle Redundancy Check error exists the next instruction is skipped. If a Cycle Redundancy Check error exists the next instruction is executed. The 9-track tape system, when writing tape, generates a cycle redundancy character (CRC) and writes it on the tape after each record. When the tape is being read back into the computer, the cycle redundancy character is generated again and is compared with the one written on the tape. If the two characters do not compare, a CRC error exists and can be tested for with a TEU instruction.
10-15	Unused

BTC With Magnetic Tape

The automatic block transfer control (BTC) unit is an optional computer input/output control unit which enables a fully-buffered transfer of data between peripheral units and computer memories. All magnetic tape control units are designed to operate with a BTC unit and over the standard I/O bus without BTC. However, a BTC unit is normally used with all tape control units, except in some low usage or low transfer rate applications. The I/O handlers in the standard software assume the presence of a BTC unit connected to the TCU.

Word Transfer (No BTC) With Magnetic Tape

This method of transfer can be used with any magnetic tape unit available with the SEL 810B. When

using this method of transfer, the data is output by means of an AOP or MOP instruction or input by means of an AIP or MIP. In this mode, end-of-record is generated when the data flow to the tape unit ceases. When reading, EOR is sensed in the normal manner.

Care must be taken when outputting to the TCU without the BTC, to insure that the data is available to the TCU in the time required by the speed of the tape transport and the recording density. The same considerations must be made when reading tape or data will be lost.

Interrupts With Magnetic Tape

There are two standard interrupts available with SEL tape transports. One interrupt is an "end-of-record" interrupt that occurs when an EOR is written or sensed. The second standard interrupt available is the "Word" interrupt which is used if no BTC is available.

In the condition of output, or writing on the tape, the TCU will interrupt (starting with the second word) anytime its word buffer is ready to receive data. In the condition of input, or reading from the tape, the TCU will interrupt (starting with the first word) anytime its word buffer is ready to send data.

Other interrupts such as EOF interrupt, parity error interrupt, end-of-tape interrupt, and information overflow interrupts are optionally available.

Tape Transport Selector Switches

The tape transport selector switches should not be changed on any of the tape transports unless the transport is in the local mode.

Sample Magnetic Tape Program for BTC

Table 6-12 gives a routine that will read one record from tape and will terminate the BTC with the completion of that transfer since the terminate bit was set in the block length word.

Table 6-12. Programming Routine for Magnetic Tape With BTC

Loc.	Oper.	Address	Comments
READ	ZZZ	**	
	STA	SAVA	Save A-Accumulator

Table 6-12. Programming Routine for Magnetic Tape With BTC (Cont'd)

Loc.	Oper.	Address	Comments
	LAA	LOC	
	STA	FWA	Set up First Word Address
	LAA	SIZE	
	STA	BL	Set up Block Length
	CEU	6	Set up Tape Deck 1
	DATA	'113	Read/556/Binary/ 3 Char. Per Word
	BRU	*-2	
	CEU	6	Initialize BTC/ Enable EOR Intr/ Start Tape Motion
	DATA	'156000	
	BRU	*-2	
	LAA	SAVA	Restore A-Accumulator
	BRU*	READ	Exit
SAVA	ZZZ	**	
LOC	DAC	BLOCK	Address of Buffer
SIZE	DATA	'101750	Buffer Size, 1000 Words

HIGH SPEED PRINTER (MODEL NO. 80-700 SERIES– DEVICE NO. 5)

Three models of line printers are available which differ in printing speeds and type printer used.

Model 81-731A - 600 LPM

Model 81-732A - 1000 LPM

Model 81-733A - 300 LPM (Shuttle)

The printers use plain or pre-printed standard perforated multi-part, fan-folded paper stock. Horizontal formatting is computer controlled while vertical format can be determined by either computer commands or a vertical control loop.

The printer responds to the CEU and TEU command and test functions listed in table 6-13.

Table 6-13. CEU and TEU Second Word Bit Format for High Speed Printer

CEU Commands

Bits	Function
9	Fill Buffer or Format Tape Channel # if Bit 4 is a one
8	Clear Buffer
7	Print
5	Paper Advance One Line
4	Paper Advance to Loop Channel N
6	Paper Advance to Top of Form

TEU Commands

Bits	Function
9	Test Printer Inoperable
8	Test Bottom of Form
6	Test Parity Error
4	Test Busy

Specifications for the high-speed line printers are defined in table 6-14.

Table 6-14. Model 80-700 Series, High Speed Printer Specifications

Characteristic	Specification
Speed	300, 600 or 1000 lines per minute
Characters per Line	Up to 120
Paper Width	Up to 20 inches
Print Area	Horizontal - 10 Characters/inch; Vertical - 6 lines/inch

Table 6-14. Model 80-700 Series, High Speed Printer Specifications (Cont'd)

Characteristic	Specification
Type Face	Open Gothic
Number of Characters	64
Code Wheel Sensing	Photocell
Vertical Form Control	8 Channel
Power Requirements	115V, 60 cps, single phase
Size	56-inches wide x 34-inches high x 30-inches deep
Controls	a. Power on/off b. Master clear c. Single space d. Top of form e. Index tractors f. Format control g. Parity error

HIGH SPEED LINE PRINTER PROGRAMMING

Systems Engineering Laboratories line printers have a 120-character buffer. This buffer must be filled in the order that the characters are to appear in the printed line. If the entire buffer is not to be filled, it must be cleared prior to filling. When a function is issued to clear the buffer, spaces are loaded into the buffer by the printer logic. Since the buffer holds only 120 characters, the printer logic will not reply to an attempt to send the 121st character, and the program will "hang up" if a wait flag is used with the AOP or MOP.

The Systems Engineering Laboratories printer logic is designed to accept simultaneous commands as long as they are not in mechanical or logical conflict. For example, no two mechanical commands such as "Print" and "Advance Paper One Line" can be given at the same time. Also the buffer cannot be commanded to clear and fill at the same time. The third type of conflict that must be avoided is trying to combine the command "Advance to Format N" with anything else, since another command code would make the function ambiguous.

There are three modes for advancing paper on the line printer. One mode is to advance the paper only one line. Another is to advance the paper to the top of form, which is the logical top of page (where the printing is to begin). The third mode is

to advance to format N. This can be only one line or as much as a full page, depending on where the next punch is in the channel specified by N. N = 0 corresponds to top of form.

An example of programming the line printer is shown in table 6-15. This routine prints one line. It assumes that the data has been stored one character per word.

Table 6-15. Programming Routine for High Speed Printer

Loc.	Oper.	Address	Comments
	SPB	LINE	
	DAC	FWA	First Word Address
	DATA	WC	Character Count
LINE	ZZZ	**	Enter
	STA	SAVA	
	STB	SAVB	Save Registers
	LAA*	LINE	
	STA	ADRS	Setup First Word Address
	IMS	LINE	
	LAA*	LINE	
	NEG		
	TAB		Setup Negative Character Count
	IMS	LINE	Setup Exit Address
	CEU	5, W	
	DATA	'2200	Clear Buff/Advance one Line
	CEU	5, W	
	DATA	'100	Fill Buff
	LAA*	ADRS	
	AOP	5, W	Output Characters

Table 6-15. Programming Routine for High Speed Printer (Cont'd)

Loc.	Oper.	Address	Comments
	IMS	ADRS	
	IBS		
	BRU	*-4	Character Count not Zero
	CEU	5, W	
	DATA	'400	Print Line
	LAA	SAVA	
	LBA	SAVB	Restore Registers
	BRU*	LINE	Exit
SAVA	ZZZ	**	
SAVB	ZZZ	**	
ADRS	ZZZ	**	

PUNCHED CARD READER (MODEL NO. 81-450A 400 CPM—DEVICE NO. 4)

The SEL Model 81-450A Card Reader reads standard 80-column cards at a maximum rate of 400 cards per minute. Reading is column by column. The binary values of all 12 rows are transferred to the computer for each column read. Specifications for the punched card reader are defined in table 6-16.

Table 6-16. Model No. 81-450A Punched Card Reader Specifications

Characteristic	Specification
Speed	400 cards/minute
Read Mechanism	Photoelectric
Hopper Capacity	1000 cards
Dimension	22-inches wide x 52-inches high x 30-inches deep
Power	115 VAC ±10%, 60 cps, ±3 cps

PUNCHED CARD READER PROGRAMMING

The unit responds to a Feed a Card CEU command. This command also functions as a test. If a unit is ready and no Wait flag is used, the program counter will skip to the next sequential address. If it is not ready, the program executes the next sequential address. If a Wait flag is used, the computer will wait until the unit is ready before it executes the CEU.

Table 6-17 gives a "CARD INPUT" subroutine for reading a card.

Table 6-17. Programming Routine for Punched Card Reader

Loc.	Oper.	Address	Comments
CARD	ZZZ	**	Entry
	LAA*	CARD	
	STA	IADD	Store IBUF Address
	LBA	=-80	
	CEU	4, W	Feed A Card
	DATA	'4000	
WDIN	AIP	4, W	Input Character
	STA*	IADD	Store in Input Buffer
	IBS		Increment B (Index) and Skip
	BRU	WDIN	Not Last Word
	IMS	CARD	Setup Exit
	BRU*	CARD	Exit to L + 2
IADD	ZZZ	**	

This routine is called by the sequence:

```
L     SPB     CARD
      DAC     IBUF
```

It stores 80 characters, right-justified into 80 consecutive memory locations starting at location IBUF. Control is returned to L + 2.

X-Y INCREMENTAL PLOTTER (MODEL NO. 81-810A AND 81-812A—DEVICE NO. 11)

The SEL Models 81-810A and 81-812A incremental plotters are high-speed digital, two-axis plotters. The actual plot is produced by the movement of a pen over the surface of the chart paper.

The units responds to the following command and test instructions:

a. -Y (Drive carriage right)

b. +Y (Drive carriage left)

c. -X (Rotate drum up)

d. +X (Rotate drum down)

e. -X-Y (Rotate drum up and drive carriage right)

f. +X-Y (Rotate drum down and carriage right)

g. -X+Y (Rotate drum up and drive carriage left)

h. +X-Y (Rotate drum down and drive carriage left)

i. Pen up

j. Pen down

Specifications for the X-Y Incremental Plotters are defined in table 6-18.

Table 6-18. Model No. 81-810A and 81-812A X-Y Plotter Specifications

Characteristic	Model 81-810A Specifications	Model 81-812A Specifications
Chart Width	12 inches	31 inches
Speed (X, Y direction)	18,000 steps/minute 0.005 inches. 12,000 steps/minute 0.01 inches.	12,000 steps/minute
Pen Up/Down	600 movements per minute	600 movements per minute

Table 6-18. Model No. 81-810A and 81-812A X-Y Plotter Specifications (Cont'd)

Characteristic	Model 81-810A Specifications	Model 81-812A Specifications
Resolution	0.005 inches (81-810-01A) or 0.01inches (81-810-02A)	0.01 inches
Plot Width	11 inches	29.5 or 11 inches
Chart Drive	Sprocket	Sprocket
Power	115 volts,±10%, 60 cps, 1.6 amps nominal	Same as 81-810A
Temperature	10°C to 35°C	10° to 35°C
Manual Controls	Drum forward/ reverse Carriage right/ left Pen up/down Single step and continuous modes Power on/off	

The Y-axis plot is produced by lateral movements of the pen carriage and the X-axis plot by rotary motion of the chart drum. Provisions for Z-axis modulation are also incorporated. The plotter responds to the CEU command functions listed in table 6-19.

The basic movements on each axis (X or Y) are at 0 degrees and 90 degrees, and the X, Y combinations yield 45-degree angles. The remaining two movements are pen up and pen down.

Table 6-19. CEU Second Word Bit Format for X-Y Plotter

Bit	Function
4	Pen Down
5	Pen Up
6	Drum Down
7	Drum Up
8	Carriage Left
9	Carriage Right

The Plotter can be made to move manually (off-line) by means of knobs on a continuous or on a single step basis.

Under computer control (on-line) only step movements are provided.

X-Y PLOTTER PROGRAMMING

Plotter addressing is handled by means of the CEU instruction. Six bits of the second word of the CEU command are used as control bits. The list of all the possible commands and their bit configurations is shown in table 6-20.

Table 6-20. X-Y Plotter Commands and Bit Configuration in CEU Second Word

Command	Bit Configuration	Command	Bit Configuration
Pen Up	('2000)	X+ Y+	('1200)
Pen Down	('4000)	X+ Y-	('1100)
X+	('1000)	X-Y-	('500)
Y+	('200)		
X-	('400)	X-Y+	('600)
Y-	('100)		

Program examples:

```
CEU     '11,W     Unit '11 Wait
DATA    '2000     Pen Up

CEU     '11       Unit '11
DATA    '2000     Pen Up

BRU     BUSY      Calcomp Busy
```

Table 6-21 shows a Calcomp visual diagnostic routine. It plots a polygon using all the possible movements including pen up and pen down at a predetermined scale factor. The calling sequence is: SPB CDIA. The A-Accumulator is loaded with the desired scale factor. This program lowers the pen, draws an octagon using the eight movements and raises the pen before halting.

Table 6-21. Diagnostic Routine for X-Y Plotter

Loc.	Oper.	Address	Comments
CDIA	ZZZ	**	Calcomp Visual Diagnostic

Table 6-21. Diagnostic Routine for X-Y Plotter (Cont'd)

Loc.	Oper.	Address	Comments
	NEG		Set Scale Factor Counter
	STA	HOWN	Save It
	STA	TIMS	
	LBA	MEIG	(-8) No. of Different Movements
	CEU	'11, W	Pen Down, Wait
	DATA	'4000	
GO	CEU*	'11, W	
	DAC	MEIG +9, 1	
	IMS	TIMS	Completed Scale Factor
	BRU	GO	No, Output Same Command
	IBS		Yes, Test for all Movements Done
	BRU	*+4	No, Do Next One
	CEU	'11, W	Yes, Bring Pen Up
	DATA	'2000	
	BRU*	CDIA	Exit
	LAA	HOWM	
	STA	TIMS	
	BRU	GO	
HOWM	ZZZ	**	
TIMS	ZZZ	**	
MEIG	DATA	-8	
	DATA	'200	
	DATA	'1200	
	DATA	'1000	
	DATA	'1100	

Table 6-21. Diagnostic Routine for X-Y Plotter (Cont'd)

Loc.	Oper.	Address	Comments
	DATA	'100	
	DATA	'500	
	DATA	'400	
	DATA	'600	

MOVABLE HEAD DISC STORAGE (MODEL NO. 81-653A— DEVICE NO. 13)

The SEL Model 81-653A Disc Storage System (DSS) consists of a Disc Control Unit (DCU) and a Disc Storage Drive.

The DSS is a random access, bulk storage device with a Storage capability of 1,536,000, 16-bit words. The disc is subdivided into tracks, surfaces and sectors. Each recording surface of the disc is accessed by a moveable head. The head can be moved to any of 100 tracks. Each track contains 16 sectors. Figure 6-3 shows the track and sector layout of a recording surface.

The ten recording surfaces of the disc pack are addressed by the moveable head assembly. Each surface is read/written by an individual head. Figures 6-4 and 6-5 show the head arrangement in relation to the recording surfaces.

Each sector will store 96 words, thus each track of a recording surface will store 1,536 words and an entire recording surface will store 153,600 words.

Considering a disc pack as 100 cylinders, 15,360 words can be written/read in each cylinder without moving the head assembly.

The disc rotates at 2400 RPM. This gives a maximum latency time of 25 milliseconds. Figure 6-6 shows the time required to move the head "n" positions. The data transfer rate of the disc system is 78.125 KHz, or one word every 12.8 microseconds.

MOVEABLE HEAD DISC STORAGE PROGRAMMING

The CEU instruction is used to command the DCU. There are two types of CEU second word formats, DISC SEEK and DISC DATA. The disc seek command is used to position the head assembly to the required track. Refer to Appendix C for the CEU second word formats for the disc.

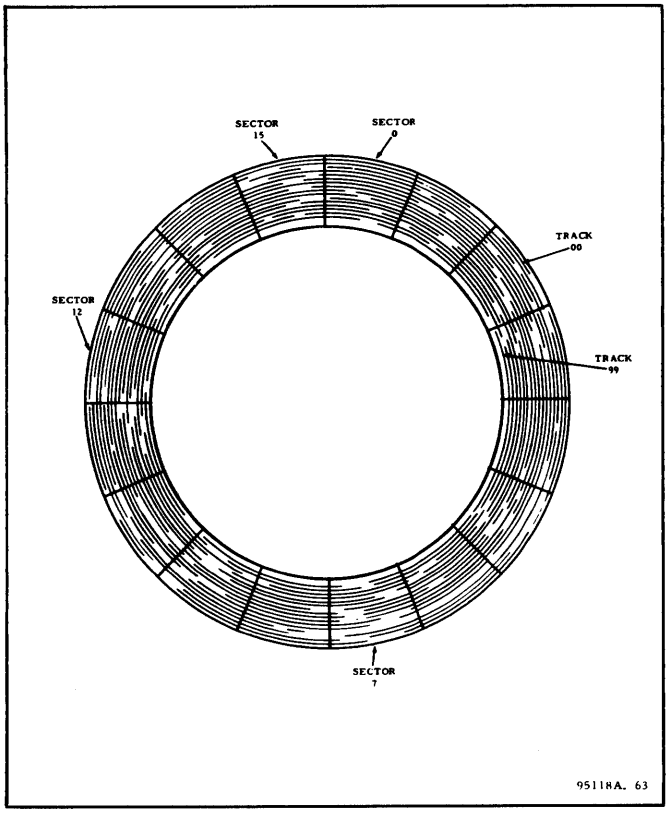


Figure 6-3. Track and Sector Layout

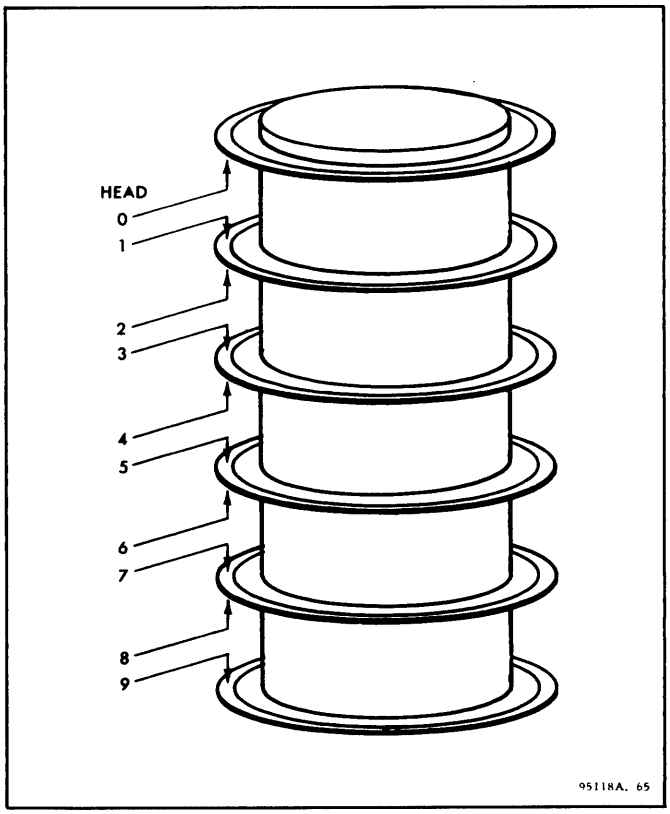


Figure 6-5. Head Position

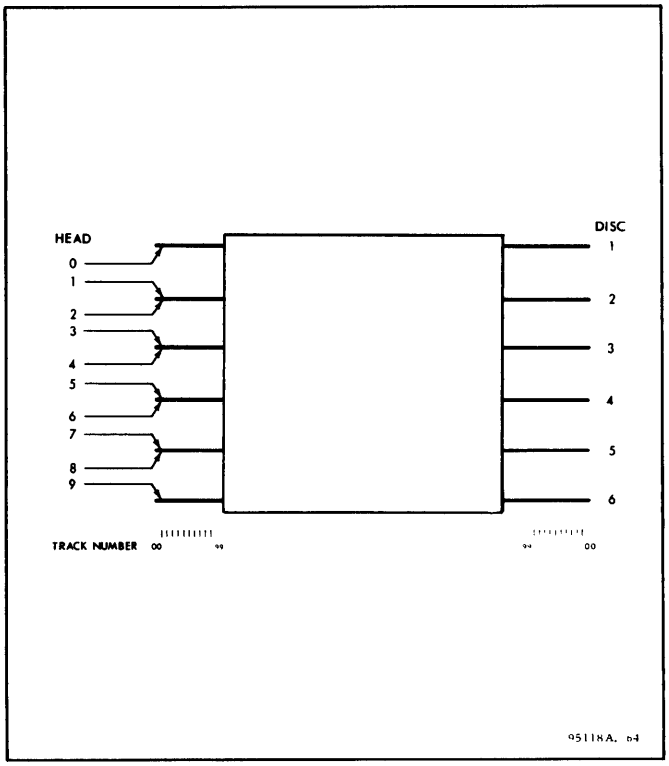


Figure 6-4. Movable Head Arrangement - Recording Surface

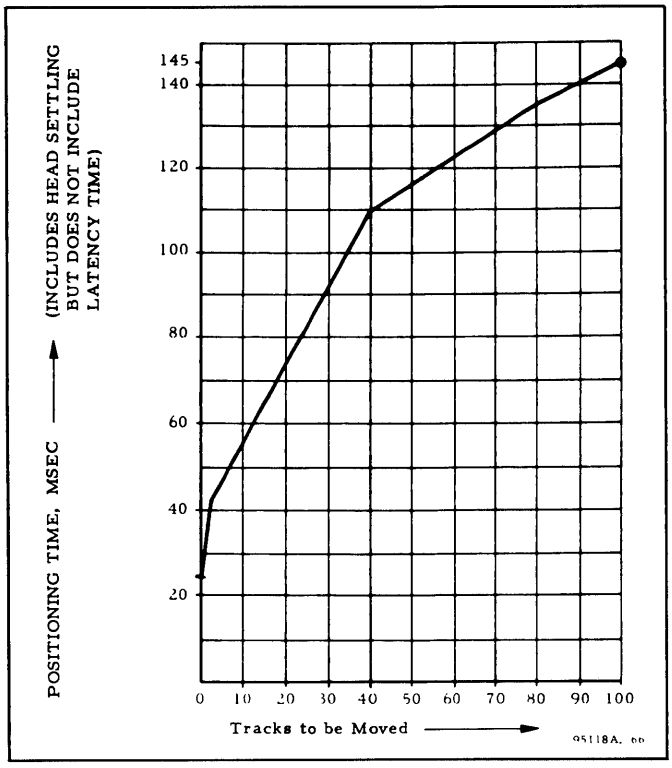


Figure 6-6. Typical Head Positioning Time Chart

The Disc Control Unit accepts a total of five commands from the computer which define all permissable disc operations. These commands are:

a. Seek Track Zero
b. Seek N Tracks Forward } Disc Seek Mode
c. Seek N Tracks Reverse
d. Write Sector I, Head J } Disc Data Mode
e. Read Sector I, Head J

The three seek commands enable the ten physically connected heads to be positioned to any desired track number. The program can keep track of the current head assembly position and command the head assembly to be moved a specified number of tracks in either direction to position the heads to a new track number. The positioning mechanism is quite reliable, but an absolute verification of the new head position can be obtained by recording track and sector identification in one or all sectors per track and reading a sector containing I. D., each time the heads are repositioned.

An alternate method of track accessing consists of sending the heads to track zero after each disc transfer is completed. Use of this technique enables absolute rather than relative track addressing but it does increase the minimum time between successive disc operations.

The read and write commands enable any sector on the ten tracks currently under the disc heads to be written or read.

To seek track 00 (when the current track is unknown) the following CEU is executed:

```
CEU  '13        CEU '13, W
DATA '10        DATA '10
BRU*-2
```

Once the head is positioned at any track, motion commands specify the number of tracks to be moved, and the direction of movement (forward or reverse).

For example, assume if the head assembly is at track 50 and the new positions are to be, successively, track 55, track 71, track 38 and track 43. The instructions listed in table 6-22 must be executed to move the head assembly to the required tracks.

Table 6-22. Moveable Head Disc Storage, Movement Commands

Command	Movement
CEU '13, W DATA '132 . . .	Forward, 5 tracks
CEU '13, W DATA '412 . . .	Forward, 16 tracks
CEU '13, W DATA '1031 . . .	Reverse, 33 tracks
CEU '13, W DATA '132	Forward, 5 tracks

Note that in a disc seek command bit 12 of the CEU second word is always a ONE.

Head and sector selection are performed by executing the CEU with the disc data second word format.

For example, to read sector 12, head 5 the CEU instruction would be;

```
CEU '13, W
DATA '6121
```

and to write sector 7, head 7 the CEU instruction would be;

```
CEU '13, W
DATA '3562.
```

The two standard interrupts can be connected/disconnected by the execution of the CEU instruction with the appropriate combination of bits 1, 2 and 3 of the CEU second word. The seek error interrupt occurs when a motion command occurs that cannot be executed; for example, the heads are at track 70 and a "forward 70 tracks" command is given. The seek complete interrupt occurs when the heads are at the selected track.

The TEU instruction can be used to test for seek complete, seek error, disc pack on line, read overflow, write overflow, checksum error, DCU ready and unit busy.

Data is transferred between the disc and the computer one word at a time, in not more than 96 words (1 sector) blocks.

When using the Block Transfer Control Unit with the disc, the BTC is initialized with bit 0 of the Disc Data CEU. If more than one sector of data is to be read or written, the terminate bit should be set in the word count location so that the interrupt processing routine can handle the bookkeeping functions (sector and head modification, first word address of buffers, etc.).

For non-BTC operation, a data word must be presented to the disc each 12.8 microseconds. Otherwise data will be dropped during the transfer. This transfer rate restricts other operations that can be performed concurrently in the computer. Therefore, all transfers are normally made between disc and computer via BTC units. The I/O disc handler routines assume BTC operation.

FIXED HEAD DISC STORAGE (MODEL NO. 81-654A— DEVICE NO. 13)

The Fixed Head Disc Storage Unit (DSU) provides random-access bulk storage of output data from any SEL Series 800 computer. Storage capacity, when used with a SEL 810B Computer, is up to 909,312 16-bit words. Units with from one to eight recording surfaces are available. There are 64 fixed recording heads per surface (refer to figure 6-7). Each surface contains 64 recording tracks with each track divided into 16 sectors (refer to figure 6-8). Each sector provides storage for 111, sixteen bit data words. Average access time for data recording or retrieval is 8.3 milliseconds. Maximum access time is 17 milliseconds. Word transfer rate to or from the storage unit is 112.5 KHz for 16-bit word storage. The data format is completely under program control and can take any required form. The Disc Control Unit (DCU) regulates data transfer between the DSU and the computer. The DCU also performs all track and sector selections required to store and retrieve data from specified disc locations. A checksum is generated and written at the end of each recorded sector. A checksum is also generated from the data read from a sector and compared to the checksum written at the end of that recorded sector. Additional specifications are listed in tables 6-23 and 6-24.

Table 6-23. Model 81-654A Fixed Head Disc Storage Specifications

Characteristic	Specification
Speed	3600 RPM, 16.7 milliseconds/revolution ±S%*
Capacity	Bits - 3,637,248 16-Bit Words - 227,328
Word Rate	112.5 KHz
Bit Rate	1.8 MHz
Number of Tracks	64 per surface 128 per disc
Access Time	Average - 8.3 milliseconds Maximum 16.7+S* milliseconds
Time Between Sectors	Approximately 37.2 microseconds +S*
Error Detection	Checksum
Interrupts (2)	1. Program Error or Checksum Error 2. Read and Write Overflow
Recording Density	1000 BPI
Disc Coating	Nickel-Colbalt Magnetic Plating
Prerequisitor	Connection to Computer BTC unit.
Options	Expansion of storage capacity to 14,548,992 bits by increasing number of discs to four and number of heads to 512.

*S=Induction Motor Slippage, approximately 3 to 4%

Table 6-24 lists the fixed head disc capacity.

Table 6-24. Model 81-654-128A Disc Storage Capacity Specifications (One Disc)

	Bits	Words	Sector	Track	Surfaces
Per Word	16	-	-	-	-
Per Sector	1776	111	-	-	-
Per Track	28,416	1776	16	-	-

Table 6-24. Model 81-654-128A Disc Storage Capacity Specifications (One Disc) (Cont'd)

	Bits	Words	Sector	Track	Surfaces
Per Surface	1,818,624	113,664	1024	64	-
Total	3,637,248	227,328	2048	128	2

FIXED HEAD DISC STORAGE PROGRAMMING

TEU and CEU word formats for the 810B are included in tables 6-25 and 6-26. During on-line operation, commands for testing and transferring data to the DCU from the computer are generated by instructions within the computer program. Test function are performed by TEU (Test External Unit) instruction. Data transfer is accomplished by AOP (Accumulator Out to Peripheral), MOP (Memory out to Peripheral), AIP (Accumulator In from Peripheral) or MIP (Memory in from Peripheral).

The DCU accepts a total of five command instructions (CEU).

The DCU accepts a total of seven test instructions (TEU) used to verify the status of the disc storage system. Each of these instructions test circuits in the DCU which have two conditions, set or reset. If the reset state is detected when the particular test is performed, a program skip is initiated. Conversely, if the set state is detected, no skip is initiated.

PRIORITY INTERRUPTS FOR FIXED HEAD DISC

The DCU is equipped with two priority interrupts designated #1 and #2. The interrupt circuits are enabled or disabled by CEU command instructions in the computer program. A single CEU command instruction can enable or disable one or both interrupts.

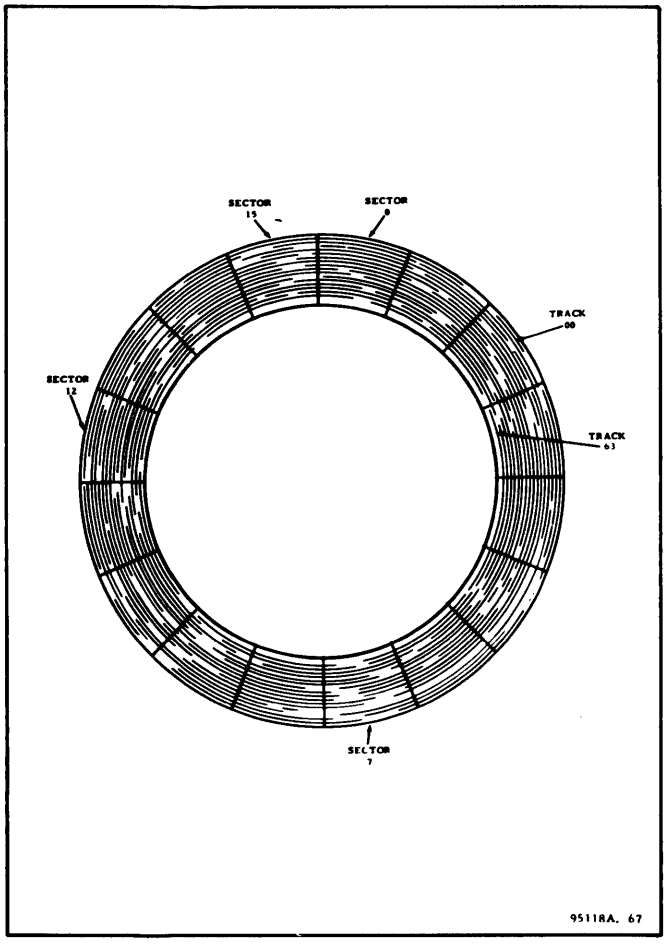


Figure 6-7. Fixed Head Track and Sector Layout

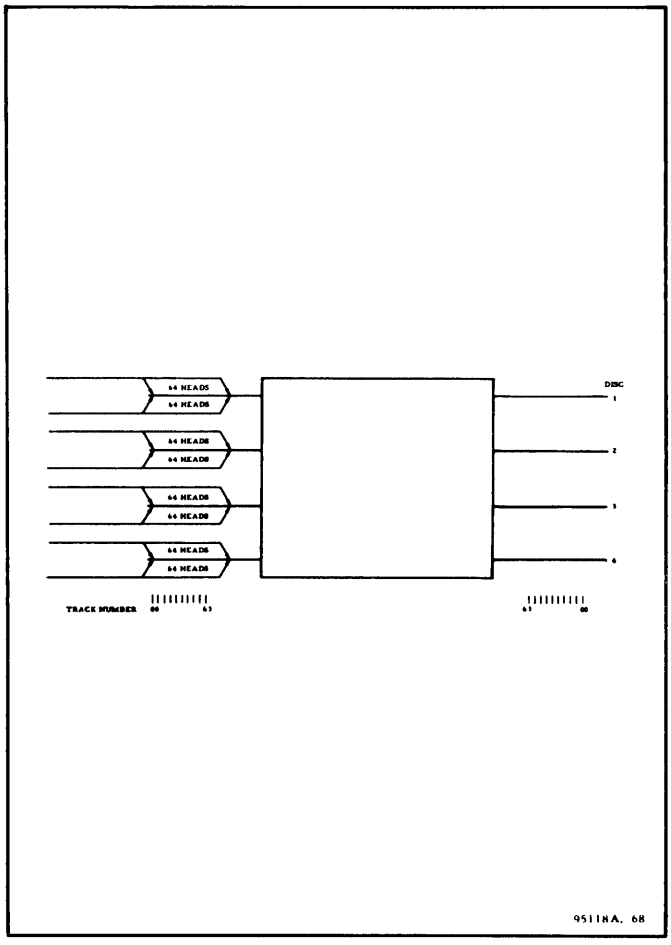


Figure 6-8. Fixed Head Arrangement - Recording Surface

Table 6-25. Fixed Head Disc TEU Second Word Format

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Skip On No Program Error	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Skip On Disc On Line	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Skip On No Disc Read Overflow	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Skip On No Disc Write Overflow	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Skip On No Parity Error	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Skip On No Disc File Area Protected	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Skip On Disc Controller Not Busy	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 6-26. Fixed Head Disc CEU Second Word Format

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Select Track	BTC	P/I Enable Disable	P/I No. 1	P/I No. 2	← Track Number →									BTC	1	1
					256	128	64	32	16	8	4	2	1			
Write Sector N	BTC	P/I Enable Disable	P/I No. 1	P/I No. 2	Not Used					Sector				BTC	Write 1	0
										8	4	2	1			
Read Sector N	BTC	P/I Enable Disable	P/I No. 1	P/I No. 2	Not Used					Sector				BTC	0	Read 1
										8	4	2	1			
Write Starting At Sector N Sequential	BTC	P/I Enable Disable	P/I No. 1	P/I No. 2	Seq. 1	Not Used				Starting Sector				BTC	Write 1	0
										8	4	2	1			
Read Starting At Sector N Sequential	BTC	P/I Enable Disable	P/I No. 1	P/I No. 2	Seq. 1	Not Used				Starting Sector				BTC	0	Read 1
										8	4	2	1			

SECTION VII
OPTIONS

PROGRAM PROTECT AND INSTRUCTION TRAP (MODEL 81-080B)

When the program protect option is included in an 810B Computer, the memory is divided into 16 areas of 1024 (-1B option) or 2048 (-2B option) words each. A 16-bit protect register is included in the computer which stores the protect status of each memory area. The protect logic causes an interrupt to be generated if an instruction attempts to write into a protected memory area when the computer is operating in the "Unpriviledged" state. Instructions are provided for loading and storing the contents of the protect register.

The status of the program protect mode ("Priviledged" or "Unpriviledged") is maintained by the "protect latch" (PL).

The protect latch operates as follows:

a. The protect latch is set ON ("Priviledged" state) when the mode key switch is turned from disable to enable. Thereafter, it is turned ON each time a priority interrupt occurs.

b. When the protect latch is ON, any instruction in a protected memory area can be executed.

c. Any instruction in an unprotected memory area can also be executed provided that no attempt is made to write into a protected memory area. However, execution of any legal instruction in an unprotected area causes the protect latch to be turned OFF.

d. When the protect latch is OFF, any attempt to execute an instruction which attempts to write into a protected area will generate a protect violation interrupt, regardless of whether the instruction is stored in a protected or unprotected area. This feature prevents unprotected programs from randomly entering protect programs.

e. Any priority interrupt will turn ON the protect latch and any instruction within the interrupt subroutine which is not in a protected area will turn OFF the protect latch. To insure that the protect status that was present at the time of the interrupt is returned after the interrupt subroutine is completed, the protect latch status is stored as follows: after the interrupt has occurred, when the wired SPB instruction is executed, the status of the protect latch is stored in bit 0 of the effective address defined to store the program counter contents. When the TOI and BRU indirect (or LOB) instructions are executed following the interrupt subroutine, the protect latch is returned to the status present at the time the interrupt occurred.

There are two control panel indicators associated with the program protect feature. One indicator displays the status of the protect latch, and the other indicator (located adjacent to the A-Accumulator display indicators) is lit when the protect register is selected for display on the row of indicators that normally displays the A-Accumulator contents.

VARIABLE BASE REGISTER, (MODEL 81-042B)

The variable base register is a 6-bit register which allows any MAP to be used as a reference or base MAP. Whenever the MAP and index bits of an instruction are set to logical zero, the contents of the VBR are treated as the most significant bits of, and appended to, the nine-bit operand address. If the MAP bit is set to a logical ONE, the most significant six bits of the operand address are defined by the most significant six bits of the program counter. If the VBR is set to zero, the 810B operates just as if no VBR were present.

The contents of the VBR are not appended to memory reference instructions having a ZERO MAP bit when indexing is specified in the instruction (Index Bit = 1). This feature permits relative addressing by indexing to be performed independently of the VBR contents. For example, execution of the instructions,

```
LBA = '1000
LAA   0, 1
```

causes the contents of '1000 to be loaded into A-Accumulator regardless of the VBR contents.

STALL ALARM (MODEL 81-043B)

The stall alarm is designed to correct, or inform the computer operator of, the stalling or "hanging up" of computer operations. If after 32 machine cycles (0:79 microseconds per cycle), the computer program counter has not advanced, an "Override" interrupt is generated. This override interrupt is capable of interrupting an indirect chain, an I/O instruction, or even a Halt condition. The interrupt routine assigned to this level can take the most suitable corrective action.

AUTO START (MODEL 81-041B

The auto start feature provides the capability of the computer to return to a "run" condition automatically in the event that, after being "lost", power is restored. An "override" interrupt is provided which allows return to the regular program or to a recovery routine.

TABLE TOP (MODEL 81-057B)

This option provides a table-height writing surface mounted on the front of the computer cabinet for the convenience of the operator.

INPUT/OUTPUT PARITY (MODEL 81-210B)

This option is made available for use with special computer interfaces such as serial communication links in which a parity bit is transmitted with each word. This option consists of a means to transfer the parity bit stored in memory with each computer word output transfer, and to check the parity bit accompanying each computer input word transfer.

The operation of the I/O Parity unit is controlled by the external interface unit which must send a request for parity checking/transfer with each input/output word request. If a parity error is detected for an input transfer, the error signal is transferred to the external unit.

INDEX REGISTER (MODEL 81-006B)

The optional index register can be used to perform the same indexing functions as those performed using the B-Accumulator. This additional, programmable register is made available to make the SEL 810B Computer even more powerful in applications involving extensive operand and address manipulation. Two single precision, or one double precision operand can be manipulated in the A- and B-Accumulators and the optional index register can be used for address manipulation without disturbing the contents of the B-Accumulator.

A program controlled flip-flop is supplied with the optional index register which acts as an index register pointer. When this flip-flop is set to one state, the execution of an instruction containing the Index Flag causes the contents of the B-Accumulator to be added to the operand address. When the flip-flop is set to the other state, the presence of an Index Flag in an instruction causes the contents of the optional index register to be added to the operand address.

Instructions are provided to change and to test the state of the index pointer flip-flop. Therefore, either the B-Accumulator or the optional index register can be used for indexing in a program, or both can be used in the same program for double indexing operations.

The eight instructions provided with the optional index registers are: LIX, STX, SXB, IXS, TAX, TXA, XPX, and XPB. Refer to the SEL 810B Instruction List Summary and to Section II of this manual for description of these instructions.

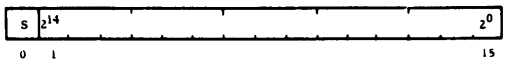
60Hz REAL-TIME CLOCK (MODEL 81-031B)

This option provides interrupt signals at the frequency of the ac input power supplied to the computer.

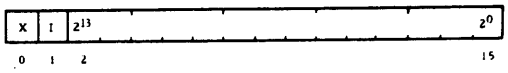
APPENDIX A

SEL 810B COMPUTER WORD FORMATS

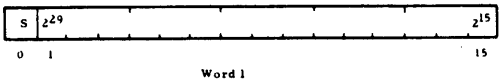
Integer Data

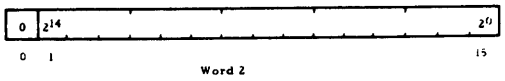


Indirect Address Word

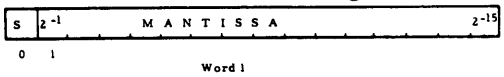


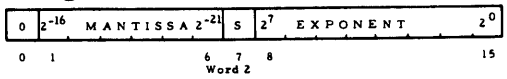
Double-Precision Fixed Point Data



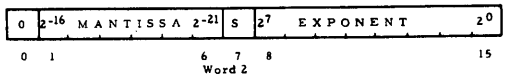


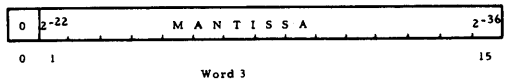
Single-Precision Floating Point Data



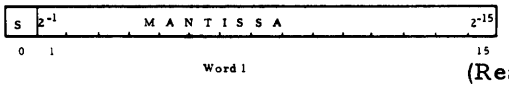


Double-Precision Floating Point Data

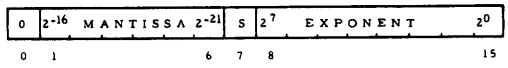


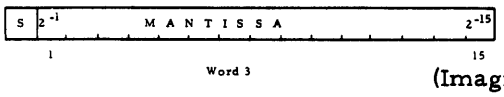


Complex Floating Point Data

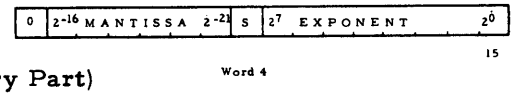


(Real Part)

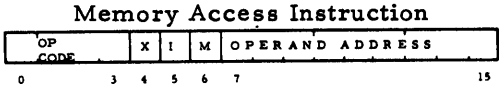




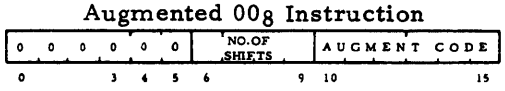
(Imaginary Part)



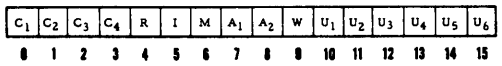
Memory Access Instruction



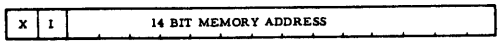
Augmented 00_8 Instruction



Input/Output Instructions

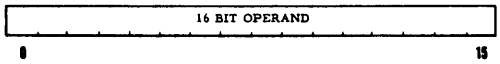


First Word



Second Word (Address Mode)

OR



OR

Second Word (Immediate Mode)

95118A A1

APPENDIX B

SEL PERIPHERAL DEVICE OCTAL CHARACTER CODES

ALPHABETIC CHARACTERS

Character	Teletype ASR-33 & ASR-35	Line Printer (Truncated ASCII)	IBM/BCD	Hollerith Card Code	
				Octal Code	Card Rows
A	301	01	61	4400	12-1
B	302	02	62	4200	12-2
C	303	03	63	4100	12-3
D	304	04	64	4040	12-4
E	305	05	65	4020	12-5
F	306	06	66	4010	12-6
G	307	07	67	4004	12-7
H	310	10	70	4002	12-8
I	311	11	71	4001	12-9
J	312	12	41	2400	11-1
K	313	13	42	2200	11-2
L	314	14	43	2100	11-3
M	315	15	44	2040	11-4
N	316	16	45	2020	11-5
O	317	17	46	2010	11-6
P	320	20	47	2004	11-7
Q	321	21	50	2002	11-8
R	322	22	51	2001	11-9
S	323	23	22	1200	0-2
T	324	24	23	1100	0-3
U	325	25	24	1040	0-4
V	326	26	25	1020	0-5
W	327	27	26	1010	0-6
X	330	30	27	1004	0-7
Y	331	31	30	1002	0-8
Z	332	32	31	1001	0-9

APPENDIX B (CONT'D)

SEL PERIPHERAL DEVICE OCTAL CHARACTER CODES

NUMERIC CHARACTERS

Character	Teletype ASR-33 & ASR-35	Line Printer (Truncated ASCII)	IBM/BCD	Hollerith Card Code	
				Octal Code	Card Rows
0	260	60	12	1000	0
1	261	61	01	0400	1
2	262	62	02	0200	2
3	263	63	03	0100	3
4	264	64	04	0040	4
5	265	65	05	0020	5
6	266	66	06	0010	6
7	267	67	07	0004	7
8	270	70	10	0002	8
9	271	71	11	0001	9

SPECIAL SYMBOLS OR FUNCTIONS

Symbol Or Function	Teletype ASR-33 & ASR-35	Line Printer (Truncated ASCII)	IBM/BCD	Hollerith Card Code	
				Octal Code	Card Rows
@	300	00	57	2006	11-8-7
[	333	33	75	4022	12-8-5
\	334	34	36	1012	0-8-6
]	335	35	55	2024	11-8-5
↑	336	36	32	1202	0-8-2
←	337	37	77	4006	12-8-7
Space	240	40	20	—	—
!	241	41	52	3000	11-0
"	242	42	37	1006	0-8-7
#	243	43	35	1022	0-8-5
$	244	44	53	2102	11-8-3
%	245	45	—	—	—
&	246	46	—	—	—

95118A. B2

SEL PERIPHERAL DEVICE OCTAL CHARACTER CODES

SPECIAL SYMBOLS OR FUNCTIONS

Symbol Or Function	Teletype ASR-33 & ASR-35	Line Printer (Truncated ASCII)	IBM/BCD	Hollerith Card Code	
				Octal Code	Card Rows
'	247	47	14	0042	8-4
(	250	50	34	1042	0-8-4
)	251	51	74	4042	12-8-4
*	252	52	54	2042	11-8-4
+	253	53	60	4000	12
,	254	54	33	1102	0-8-3
—	255	55	40	2000	11
.	256	56	73	4102	12-8-3
/	257	57	21	1400	0-1
:	272	72	15	0022	8-5
;	273	73	56	2012	11-8-6
<	274	74	76	4012	12-8-6
=	275	75	13	0102	8-3
>	276	76	16	0012	8-6
?	277	77	72	5000	12-0
Carriage Return	215				
Line Feed	212				
Bell	207				
Delete	377				

95118A. B3

APPENDIX C

SEL 810 PERIPHERAL DEVICE COMMAND AND TEST CODE FORMATS

CEU SECOND WORD FORMAT

	0	1	2*	3*	4	5	6	7	8	9	10	11	12	13	14	15
Magnetic Tape Format 0	0	P.I. Connect = 1 Disconn = 0	Word Transfer Ready Interrupt	End of Record Interrupt	0	Rewind	Erase 4 Inches of Tape	BCD = 1 Binary = 0	Density**		Tape Transport			Current Word Address In	Characters Per Word	
Magnetic Tape Format 1	BTC Initialize		Word Transfer Ready Interrupt	End of Record Interrupt	1	Write Record	Write End of File	Read Record	Advance Record	Advance/ End of File	Backspace Record	Backspace End of File		Current Word Address In		
ASR-33/35			In	Out	Reader Mode	Key Mode	Clear									
Paper Tape Reader and Punch			In	Out	Feed		Reader Enable	Reader Disable								
Card Reader/ Punch			In	Out	Feed Card Reader		Read Stacker Offset	Feed Card Punch	Eject Card (Punch)	Punch Stacker Offset						
X-Y Plotter			Process Complete		Pen Down	Pen Up	Drum Down	Drum Up	Carriage Left	Carriage Right						
Line Printer			End of Print	Buffer Not Busy	Advance Paper To Format Tape Channel***	Advance 1 Line	Top of Form	Print	Clear Buffer	Fill Buffer						
								Or Format "n" if Bit 4								
Moveable Head Disc Seek			Seek Error	Seek Complete				Number of Tracks to be Moved					1		FWD ****	REV ****
						64	32	16	8	4	2	1				
Moveable Head Disc Data			Seek Error	Seek Complete	Sector Number					Head Number			0		Write	Read
					8	4	2	1	8	4	2	1				
Fixed Head Disc Select Track			Checksum Error or Program Error	Read Overflow or Program Error				Track Number							1	1
					256	128	64	32	16	8	4	2	1			
Fixed Head Disc Read			Checksum Error or Program Error	Read Overflow or Write Overflow	Read Sequential						Starting Sector				0	Read 1
Fixed Head Disc Write			Checksum Error or Program Error	Read Overflow or Write Overflow	Write Sequential						Starting Sector				Write 1	0
CRT			Overflow	Stop	Display On	Display Off										

*Interrupt Levels:
Bit 2 = Group 1, Level 1
Bit 3 = Group 1, Level 2

**Magnetic Tape Density:

Bits 8	Bits 9	Density	Bits 14	Bits 15	Characters Per Word
0	0	200 BPI	0	1	1
0	1	556 BPI	1	0	2
1	0	800 BPI	1	1	3
			0	0	4

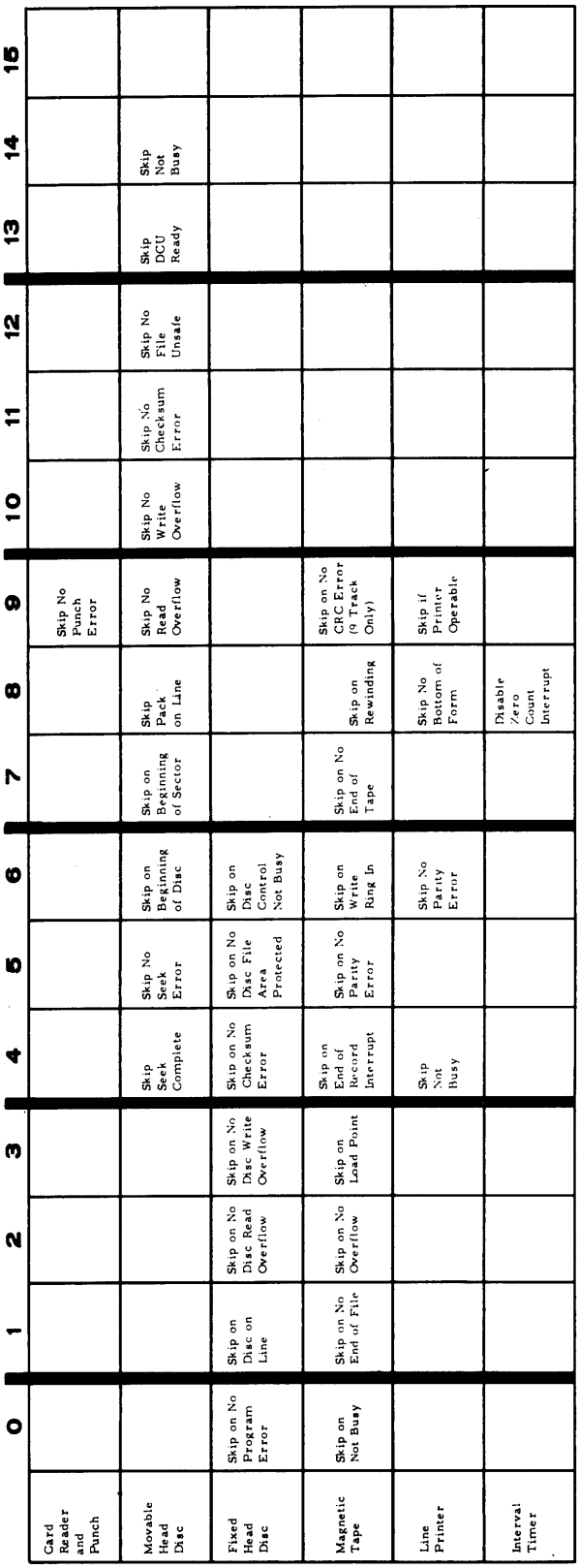
***When a one is present in bit position 4, advance to the format tape channel number (expressed in octal) represented by the bits present in positions 7, 8 and 9.

****To seek track 00 both bits must be zero.

APPENDIX C (CONT'D)

SEL 810 PERIPHERAL DEVICE COMMAND AND TEST CODE FORMATS

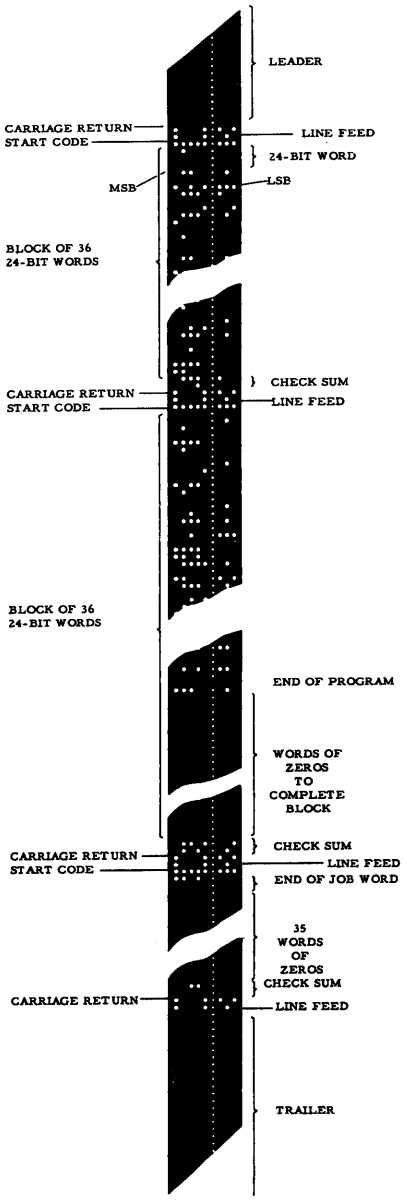
TEU SECOND WORD FORMAT



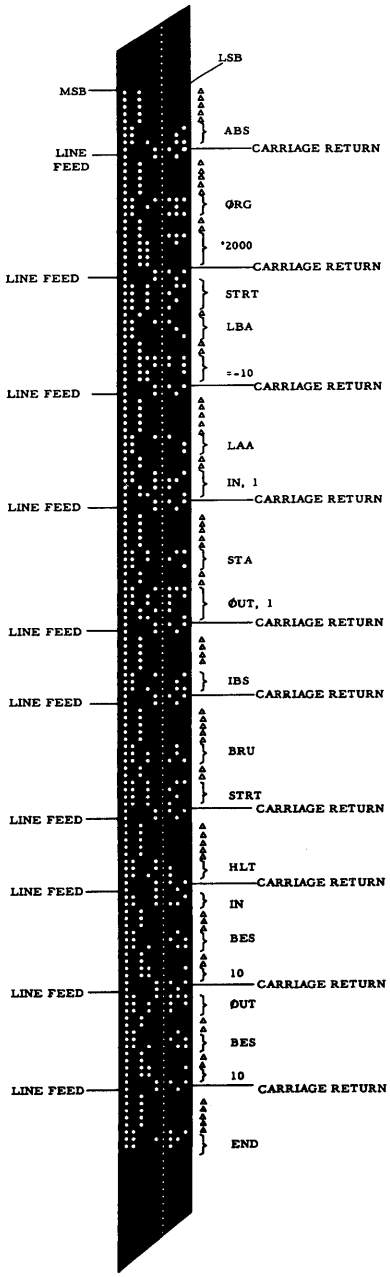
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Card Reader and Punch										Skip No Punch Error						
Movable Head Disc					Skip Seek Complete	Skip No Seek Error	Skip on Beginning of Disc	Skip on Beginning of Sector	Skip Pack on Line	Skip No Read Overflow	Skip No Write Overflow	Skip No Checksum Error	Skip No File Unsafe	Skip DCU Ready	Skip Not Busy	
Fixed Head Disc	Skip on No Program Error	Skip on Disc on Line	Skip on No Disc Read Overflow	Skip on No Disc Write Overflow	Skip on No Checksum Error	Skip on No Disc File Area Protected	Skip on Disc Control Not Busy									
Magnetic Tape	Skip on Not Busy	Skip on No End of File	Skip on No Overflow	Skip on Load Point	Skip on End of Record Interrupt	Skip on No Parity Error	Skip on Write Ring In	Skip on No End of Tape	Skip on Rewinding	Skip on No CRC Error (9 Track Only)						
Line Printer					Skip Not Busy		Skip No Parity Error		Skip No Bottom of Form	Skip if Printer Operable						
Interval Timer									Disable Zero Count Interrupt							

95118A C2

APPENDIX D
SEL 810 PAPER TAPE FORMATS



SEL 810B ASSEMBLER AND COMPILIER
OBJECT PROGRAM OUTPUT
TAPE. MUST BE LOADED
WITH THE SEL MNEMBLER
LOADER PROGRAM.

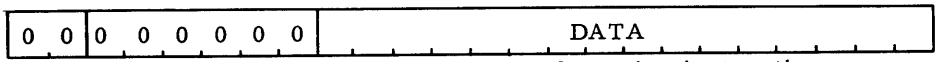


ASC II CODE ASSEMBLER
SOURCE INPUT IN CARD
FORMAT IMAGE. MUST
BE LOADED BY ASSEMBLER.

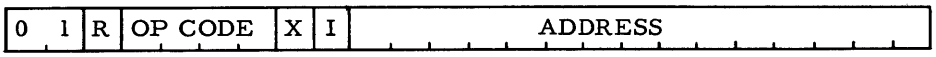
95118A. DI

APPENDIX E

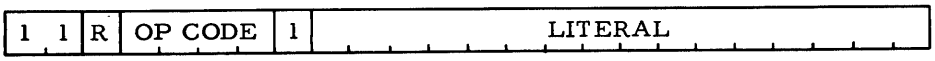
SEL 810 ASSEMBLER OUTPUT FORMATS



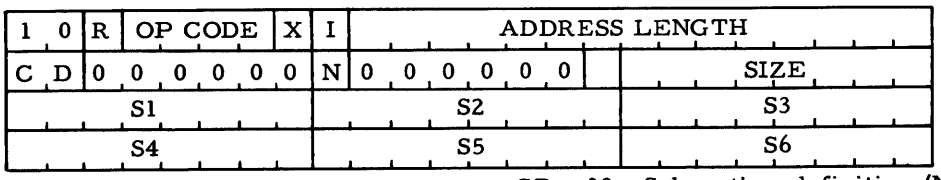
DIRECT LOAD: Data or Non-memory-referencing instructions.



MEMORY REFERENCING INSTRUCTIONS: R = Relocation flag
(DAC) OP = '13, 14-bit address constant
(EAC) OP = '17, 15-bit address constant



LITERAL REFERENCING INSTRUCTIONS:

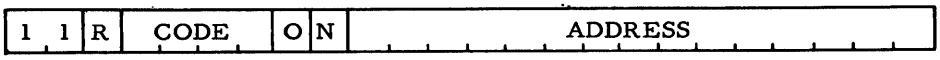


SUBROUTINE OR COMMON:
CD = 10: Common defn.
Address = length

= 11: Common request
Address = rel. to block.
N = negation flag

CD = 00: Subroutine definition (NAME)
Address = relative entry point.

= 01: Subroutine call (CALL)
Address = 0



SPECIAL ACTION:
Code = 00, Establish Load Point
= 01, END Jump
= 02, STRING
= 03, 9-Bit ADD-TO
= 04, 14-Bit ADD-TO (DAC)
= 05, 15-Bit ADD-TO (EAC)
= 06, Turn on CHAIN flag
= 07, Turn on Load flag
= 10, END-OF-JOB

APPENDIX F

NUMERICAL INFORMATION

NUMERICAL OCTAL TO DECIMAL CONVERSION

OCTAL MULTIPLICATION

m•n or n•m

m \ n	1	2	3	4	5	6	7	10
1	1	2	3	4	5	6	7	10
2	2	4	6	10	12	14	16	20
3	3	6	11	14	17	22	25	30
4	4	10	14	20	24	30	34	40
5	5	12	17	24	31	36	43	50
6	6	14	22	30	36	44	52	60
7	7	16	25	34	43	52	61	70
10	10	20	30	40	50	60	70	100

OCTAL ADDITION

m+n or n+m

m \ n	1	2	3	4	5	6	7	10
1	2	3	4	5	6	7	10	11
2	3	4	5	6	7	10	11	12
3	4	5	6	7	10	11	12	13
4	5	6	7	10	11	12	13	14
5	6	7	10	11	12	13	14	15
6	7	10	11	12	13	14	15	16
7	10	11	12	13	14	15	16	17
10	11	12	13	14	15	16	17	20

$8^0 = 1$
$8^1 = 8$
$8^2 = 64$
$8^3 = 512$
$8^4 = 4096$
$8^5 = 32768$

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

95118A. F3

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 (Octal)	1024 (Decimal)
to	to
2777 (Octal)	1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 (Octal)	1536 (Decimal)
to	to
3777 (Octal)	2047 (Decimal)

05118A. F-4

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

OCTAL-DECIMAL INTEGER CONVERSION TABLE

4000 to 4777 (Octal)	2048 to 2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 to 5777 (Octal)	2560 to 3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

95118A.F5

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 (Octal)	3072 (Decimal)
to	to
6777 (Octal)	3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	1022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 (Octal)	3584 (Decimal)
to	to
7777 (Octal)	4095 (Decimal)

95118A. F-

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

APPENDIX F (CONT'D)

NUMERICAL INFORMATION

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

NUMERICAL INFORMATION

MATHEMATICAL INFORMATION

POWERS OF TEN ($10^{\pm n}$) IN OCTAL

10^{n}	n	10^{-n}
1	0	1. 000 000 000 000 000 000 00
12	1	0. 063 146 314 631 463 146 31
144	2	0. 005 075 341 217 270 243 66
1 750	3	0. 000 406 111 564 570 651 77
23 420	4	0. 000 032 155 613 530 704 15
303 240	5	0. 000 002 476 132 610 706 64
3 641 100	6	0. 000 000 206 157 364 055 37
46 113 200	7	0. 000 000 015 327 745 152 75
575 360 400	8	0. 000 000 001 257 143 561 06
7 346 545 000	9	0. 000 000 000 104 560 276 41
112 402 762 000	10	0. 000 000 000 006 676 337 66
1 351 035 564 000	11	0. 000 000 000 000 537 657 77
16 432 451 210 000	12	0. 000 000 000 000 043 136 32
221 411 634 520 000	13	0. 000 000 000 000 003 411 35
2 657 142 036 440 000	14	0. 000 000 000 000 000 264 11
34 327 724 461 500 000	15	0. 000 000 000 000 000 022 01
434 157 115 760 200 000	16	0. 000 000 000 000 000 001 63
5 432 127 413 542 400 000	17	0. 000 000 000 000 000 000 014
67 405 553 164 731 000 000	18	0. 000 000 000 000 000 000 001

VARIOUS CONSTANTS IN OCTAL NOTATION

π = 3.11037552421

$\pi/2$ = 1.04417665210

$1/\pi$ = 0.24276301556

$\sqrt{\pi}$ = 1.61337611067

$\ln \pi$ = 1.11206404435

$\sqrt{10}$ = 3.12305407267

e = 2.55760521305

$1/e$ = 0.27426530661

$\sqrt{e}$ = 1.51411230704

$\log_{10} e$ = 0.33626754251

$\sqrt{2}$ = 1.32404746320

ln 10 = 2.23273067355

APPENDIX G

SEL 810B INSTRUCTION LIST SUMMARY

CLASS	MNEMONIC	OP CODE	CYCLES 750 NANOSECONDS	FUNCTION	PAGE
ARITHMETIC:	AMA	05	2	Add Memory to A	2-6
	AMB	16	2	Add Memory to B	2-6
	SMA	06	2	Subtract Memory from A	2-6
	MPY	07	6 Sec.	Multiply B times Memory	2-7
	DIV	10	11 Sec.	Divide A and B by Memory	2-7
	RNA'	00-01	1	Round A by MSB in B	2-8
	OVS'	00-37	1	Set Overflow	2-8
LOAD/STORE:	LAA	01	2	Load A from Memory	2-8
	LBA	02	2	Load B from Memory	2-8
	STA	03	2	Store Memory from A	2-9
	STB	04	2	Store Memory from B	2-9
	LCS'	00-31	1	Load Control Switches in A	2-9
	LIX'	00-45	2	Load Hardware Index Register	2-9
	STX'	00-44	2	Store Hardware Index Register	2-9
BRANCH/SKIP:	BRU	11	1	Unconditional Branch	2-10
	SPB	12	2	Store Place and Branch	2-10
	SNS	13-4	1	Skip if Control Switch Not Set	2-10
	IMS	14	3	Increment Memory and Skip if 0	2-10
	CMA	15	3	Compare Memory and A (3 Way) n+1 if (A)-(M) 0 n+2 if (A)=(M) n+3 if (A)-(m) 0	2-10
	IBS'	00-26	1	Increment B (Index) and Skip if 0	2-11
	SAZ'	00-22	1	Skip if A is Zero	2-11
	SAP'	00-24	1	Skip if A is Positive	2-12
	SAN'	00-23	1	Skip if A is Negative	2-11
	SOF'	00-25	1	Skip No Overflow	2-12
	SAS'	00-21	1	Skip on A Sign (3 Way) n+1(-), n+2(0), n+3(+)	2-11
	SNO'	00-32	1	Skip if A is Normalized	2-12
	LOB'	00-36	2	Long branch	2-12
	SXB'	00-50	1	Skip if Index Pointer is Set to B	2-13
	IXS'	00-51	1	Increment Index and Skip if ≥ 0	2-13
LOGICAL:	ABA'	00-27	1	AND A and B	2-13
	OBA'	00-30	1	OR A and B	2-13
	NEG'	00-02	1	Negate A	2-14
	ASC'	00-20	1	Complement A Sign	2-14
	CNS'	00-34	1	Convert Number System	2-14
REGISTER CHANGE:	CLA'	00-03	1	Clear A	2-14
	TAB'	00-05	1	Transfer A to B	2-15
	IAB'	00-06	1	Interchange A and B	2-15
	CSB'	00-07	1	Transfer B sign to Carry and Clear B Sign to Positive	2-15
	TBA'	00-04	1	Transfer B to A	2-14
	TBP	00-40	1	Transfer B-Accumulator to Protect Register	2-15
	TPB	00-41	1	Transfer Protect Register to B-Accumulator	2-15
	TAX'	00-52	1	Transfer A-Accumulator to Hardware Index Register	2-16
	TXA'	00-53	1	Transfer Hardware Index Register to A-Accumulator	2-16
	XPX'	00-46	1	Set Index Pointer to Index Register	2-16
	XPB'	00-47	1	Set Index Pointer to B-Accumulator	2-16
	TBV'	00-42	1	Transfer B-Acc. to VBR	2-16
	TVB'	00-43	1	Transfer VBR to B-Acc.	2-16
SHIFT:	RSA'	00-10	Time for Shifts vary as follows:	Right Shift A	2-17
	LSA'	00-11		Left Shift A	2-17
	FRA'	00-12		Right Shift A and B	2-17
	FLA'	00-17		Left Shift A and B	2-18
	RSL'	00-15	Shifts Time	Right Logical Shift A	2-17
	FRL'	00-14	1-4 2	Full Rotate Logical A and B	2-18
	LSL'	00-16	5-8 3	Left Logical Shift A	2-18
	FLL'	00-13	9-12 4	Left Logical Shift A and B	2-18
			13-15 5		
CONTROL:	HLT'	00-00	1	Halt	2-19
	NOP'	00-33	1	No Operation	2-19
	TOI'	00-35	1	Turn off Interrupt	2-19
	PIE'	130600	2	Enable Interrupt	2-19
	PID'	130601	2	Disable Interrupt	2-19
INPUT/OUTPUT	CEU'	13.0IM.0	4 + wait	Command External Unit	2-22
	TEU'	13.0IM.2	4 + wait	Test External Unit	2-23
	AOP'	1700	4 + wait	Accumulator Word Out to Unit	2-23
	AIP'	1702	4 + wait	Accumulator Word In from Unit	2-23
	MOP'	17.0IM.4	4 + wait	Memory Word Out to Unit	2-24
	MIP'	17.0IM.6	4 + wait	Memory Word In from Unit	2-24

Notes:

a. Underlined instructions require optional hardware.

b. ' = Augmented.

98118A.BC