2/14/72

DR 2126

K-1829

MASTER

A REAL-TIME MULTI-PROGRAMMING SYSTEM FOR THE SEL 810A

AUTHOR:

D. I. Dunthorn

UNION CARBIDE CORPORATION
NUCLEAR DIVISION
OAK RIDGE GASEOUS DIFFUSION PLANT

operated for the ATOMIC ENERGY COMMISSION *under* U. S. GOVERNMENT Contract W-7405 eng 26

UNION CARBIDE

OAK RIDGE GASEOUS DIFFUSION PLANT
P. O. Box P
Oak Ridge, Tennessee 37830

DISCLAIMER

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.95

Date of Issue: May 1, 1972

Report Number: K-1829

Subject Category: MATHEMATICS AND COMPUTERS

A REAL-TIME MULTI-PROGRAMMING SYSTEM FOR THE SEL 810A

D. I. Dunthorn
Materials and Systems Development Department
Gaseous Diffusion Development Division

UNION CARBIDE CORPORATION
NUCLEAR DIVISION
Oak Ridge Gaseous Diffusion Plant
Oak Ridge, Tennessee

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

ABSTRACT

This report describes and serves as a manual for an operating system designed to handle real-time applications using a multi-programming approach. The system is core resident on an SEL 810A computer with a small equipment configuration, including process inputs and outputs. Requiring about 4K of memory, the system consists of a time-sharing program scheduler and interrupt handler coupled with a spectrum of re-entrant utility routines which handle input-output unit declaration and waits, various program scheduling options, character string processing, floating point arithmetic, and system power failures. A conversational executive program and a desk calculator simulator, closely associated with the operating system, are also described.

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

TABLE OF CONTENTS

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

TABLE OF CONTENTS (Contd.)

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

LIST OF TABLES

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

A REAL-TIME MULTI-PROGRAMMING SYSTEM FOR THE SEL 810A

INTRODUCTION

The operating system described in this manual provides the utilities and control necessary for the programming of conversational real-time applications. Designed primarily for areas requiring response times on the order of a second or longer, the system is core resident on an SEL 810A with 8K of 16-bit words, Teletype and high-speed paper tape input-output, a 60 Hz clock, a 128-channel analog-to-digital system, and several words of digital (switch) input and output. The system requires about 4K of core storage and consists of a time-sharing program scheduler and interrupt handler coupled with a spectrum of re-entrant utility routines which handle input-output unit declaration and waits, various program scheduling options, character string processing, floating point arithmetic, and system power failures.

In order to illustrate how the system may be used, consider a simple application in monitoring a chemical process having four basic requirements:

1. Every 15 minutes, the status of thirty process variables must be recorded on paper tape.

2. Every 5 minutes, a short status report must be printed on the Teletype, giving the results of a process material balance.

3. Every 10 seconds, the status of five critical parameters must be checked and an alarm sounded if any are out of range.

4. If an out-of-range parameter is found, it must be rechecked once per second, and if it remains out of range for more than 15 seconds, a control switch must be set and a detailed report output to the Teletype.

This application, of course, presents the general requirements of computing results and communicating with peripheral devices in proper format, but more important is the attention required for timing problems. To start with, the Teletype material balance report is likely to take more than 10 seconds and certainly more than one second, and perhaps even the calculation involved may require more than this length of time. Thus, some limit checking must take place during the reporting. Additionally, there is the chance that an out-of-range Teletype report will conflict with a material balance report, so that care must be taken to avoid a jumble of characters from merging of the two reports. Under the operating system described here, however, the four requirements would be programmed as four (or perhaps three) separate programs, allowing the time-sharing system to handle the timing and interaction problems. Likewise, the system routines substantially reduce the effort required to carry out calculations and communications with peripherals.

Naturally, applications of greater complexity than the above example may be treated, also with a minimum of attention to timing details in most cases. Where required, however, provisions have been made to allow tuning the system to handle unusual situations.

SYSTEM CONCEPTS AND CONVENTIONS

The operating system makes use of a number of specialized concepts and definitions which may or may not be familiar to the reader. This section briefly describes the method of implementation of these concepts and is intended to allow the reader to delve into the system to whatever degree his requirements dictate.

Re-Entrant Routines

One of the more basic requirements for a time-shared system is that any routine which may be used by more than one time-shared program must be "re-entrant". Consider, for example, a case in which a program has been interrupted in the middle of the system routine for finding an exponential; a point at which the routine has already determined several necessary partial results in the exponential calculation. Under time-sharing operation, another program might then be activated which also required the system routine for exponentials. If normal methods of programming were used, when the second program had finished and the first program was reactivated in the middle of the exponential calculation, it would find that the previously calculated partial results had been changed by the second program, and an incorrect exponential value would result. Such system routines must therefore be made "re-entrant", which means that no data or instruction within the routine can be altered during its execution, and it thus cannot be affected by interruption during its operation. Intermediate storage is still required for most such routines, but if it is arranged that storage areas are directly associated with the calling program rather than with the system routine, then a switch between programs can simply involve a switch between storage areas to solve the problem.

Re-Entrant Routine Storage

In this operating system, locations octal 700 - 776 are used as a Program Definition Area, as described in more detail later. Part of this area is saved in a corresponding storage block associated with each program whenever a switch is made between programs to be restored whenever the program is reactivated. Provision is also made for separate, programmer defined, re-entrant storage tables. If MSTØ in the Program Definition Area of a particular program is a positive non-zero quantity, it is taken as the address of a block of three words specifying an additional re-entrant storage area:

first word - The address of the first word in the active, shared, re-entrant storage table. Usually located in map 0, this table serves the same type of function as locations 700 - 776, and is usually shared by several programs.

second word - The address of the first word in the dormant, nonshared, re-entrant storage table. This is the corresponding table that is specific to the one program alone.

third word - Negative of the number of words in the table.

These tables will, like Program Definition Areas, be switched upon program activation and deactivation.

All system utility routines may be used in a re-entrant manner, although portions of some routines are not truly re-entrant. The system prevents two programs from simultaneously using Teletype output, for example, so that making the entire Teletype output routine re-entrant would be pointless. The only portion of the routine which is re-entrant, in fact, is that which determines whether the calling program currently has control over Teletype output.

For a reasonably efficient implementation of the re-entrant system routines on the SEL 810A computer, it was found necessary to destroy the contents of at least one of the hardware registers (A or B), and conventions vary among the routines as to which of the register contents (if either) is safe over the routine. The description of each routine notes the particular convention used.

Program Definition Area

As mentioned above, a Program Definition Area is required for each time-shared program. This block of 63 words is used both to provide re-entrant routine storage and storage for program-specific system constants. The scheduler must have the indexed address of the first word of a Program Definition Area in its program table for each active program (an action which is usually performed using the ESTABLISH command of the executive system). Each time the program is activated, the contents of this area are transferred to locations octal 700-776, and when the program is later deactivated, the contents of locations 700-776 are transferred back to the Program Definition Area.

The Program Definition Area is organized as shown in table I. More detailed information on the function of the various program-specific constants is given in the sections of this report describing those areas in which they are used.

Teletype and High-Speed Paper Tape Conventions

Two pairs of system routines are used with the Teletype unit and the high-speed paper tape unit, as shown in table II. Depending upon context, the system routine might either be called directly or by using the corresponding unit number. Each of the routines operates on a

TABLE I

ORGANIZATION OF PROGRAM DEFINITION AREA

Location (octal)	Name	Function
0	PSW	Program Status Word
1	PSD1	Program Status Definition 1
2	PSD2	Program Status Definition 2
3	ESCM	Panic Action
4	SLØC	Starting Location
5	ELØC	Panic Location
6	DCV	Input-Output Unit Disuse Counter Value
7	RLSV	Saved Location Counter Setting
10	RØSV	Saved Overflow Indicator
11	RASV	Saved A Register Contents
12	RBSV	Saved B Register Contents
13	MSTØ	Secondary Re-Entrant Storage Indicator
14	PNUM	Program Number
15	TSTI	Teletype Input Status Word
16	TSTØ	Teletype Output Status Word
17	HSTI	High Speed Paper Tape Input Status Word
20	HSTØ	High Speed Paper Tape Output Status Word
21	RSTR	Restart Indicator
22	IQUA	Time Quanta
23-26		Not Currently Used
27	ITMP	Saved Index for System Routines
30-32	FA	Floating Accumulator
33-35	TR	Floating Transfer Register
36-76	U	Storage for System Routines

TABLE II

UNIT DEFINITIONS

Unit Number	System Routine	Function
0	TCI	Teletype Input
1	TCØ	Teletype Output
2	PCI	High Speed Paper Tape Input
3	PCØ	High Speed Paper Tape Output

character-by-character basis, using the rightmost eight bits of the "A" register as the ASCII character code. In the normal mode of operation, some pretreatment of input and output is carried out, as indicated in table III, and all noncontrol characters input to the Teletype are echoed back to be printed by the Teletype. Control characters other than carriage return, line feed, and control G (G^C or bell) are not echoed, and it should be noted that echoing is done, currently, *when a program calls for an input character*. This allows "type-ahead" even while the Teletype is printing something else without producing jumbled copy. Finally, the input characters are "or-ed" with octal 200, allowing the reading of tapes prepared on other than the system Teletype.

All input-output is buffered, and although the buffers are not large, they are of sufficient length to allow continuous operation of the devices except under heavy system loads. Naturally, the operating system takes care of turning input-output units on and off, dismissing programs on buffer overflows and underflows, and other similar functions.

The normal input-output conventions are convenient for use in most programming applications. For special cases, however, these conventions may be overridden by setting the appropriate status bit, as shown in table IV. "Eight-channel mode" indicates that characters are to be input and output exactly as received, with no special conventions used. "Stop after carriage return" is useful in dealing with paper tape. It causes the input unit to stop the reading of tape as soon as a carriage return is received, without reading additional characters into the buffer. Since carriage return usually indicates the end of a line of text, this allows the programmer to stop reading in the middle of a tape and continue at a later time. In this mode, a request for the next character after a carriage return will restart the reading of the tape, allowing reasonable continuity.

Automatic Unit Declaration

Whenever two or more programs use the same peripheral device on a timesharing system, an obvious conflict arises concerning which program is allowed to use the unit at any given time. One solution is to allow programs to request a unit, wait until it is available, claim it, and release it when finished. This is the method used by this operating system, although it is done in a manner which usually requires no specific action by the programmer. Whenever an input-output request is made, a check is made to determine whether the requesting program has already claimed the device in question. If it has, the request is handled; if not, the program claims the device if it is not in use. If the device is in use, the program is automatically dismissed until the device becomes available. When a program claims a device, the input-output disuse counter value (DCV) from the Program Definition Area of that program is placed in the disuse counter associated with claimed device, which functions as follows: Whenever an input-output request is fulfilled, the counter is started at its maximum (negative) value, and in between input-output requests on the particular device, the counter is incremented approximately once for each scheduler time quanta (1/20 second under

TABLE III

INPUT-OUTPUT CHARACTER CONVENTIONS--NORMAL MODE

Character	Teletype Convention	High Speed Paper Tape Convention
Input		
Cr	Carriage return is ignored if the last accepted character was a line feed	Same as Teletype
Lf	Line feed is ignored if the last accepted character was a carriage return	Same as Teletype
P^{SC}	Shift-Control P (Null) is ignored	Same as Teletype
Rubout	Rubout is ignored	Same as Teletype
L^C	Control L puts the Teletype into the tape read mode and is ignored	Ignored
D^C	Control D puts the Teletype into key mode	Control D Stops the Tape Reader
A^C	Control A echoes "↑"	No Special Treatment
Q^C	Control Q echoes "←" followed by carriage return and line feed	No Special Treatment
ALTMODE	ALTMODE causes a "panic". No special treatment if in tape reading mode	No Special Treatment
Output		
Cr	Outputting a carriage return will cause a Cr-Lf pair to be output	Same as Teletype
Lf	Outputting a line feed will cause the sequence Lf-Cr-Null to actually be output. (The Null provides timing to prevent the Teletype from misprinting the first character of the following line.)	Same as Teletype

TABLE IV

INPUT-OUTPUT STATUS WORDS

Bit Set (sign bit is 0)	Meaning	Comments
1	Eight-Channel Mode	No echoes on Teletype
2	Echo Off	Applies only to Teletype input
5	Stop After Cr	Applies only to paper tape input from either unit. Not effective in 8-channel mode
Remaining Bits		Not currently used.

standard system conventions). When the counter is incremented to zero, the program loses its claim on the device, and other programs are then free to use it.

The standard value of the disuse counter is - 200, providing about 10 seconds' pause between contiguous input-output requests. This may, of course, be reset to any desired time interval by setting DCV in the Program Definition Area, if conflicts are observed under the standard conditions. Setting DCV to a positive value has the effect of permanently declaring a unit for a program the first time it is used; however, care must be taken to reset DCV to a reasonable value before calling any other units, since permanent declaration would otherwise result in all cases.

The fact that the disuse counter is incremented only in the interval between one request being fulfilled and another request being initiated is of particular importance since, for example, a program which requests Teletype input will wait, claiming that unit, until some input is given. Thus, a program waiting in this mode effectively ties up the unit.

Multiplexer and Digital Input-Output

These methods are not used with either the analog multiplexer input or the digital input-output. For the analog system, requests for input are made in blocks, and a complete block for a requesting program is treated before requests are honored for any other program. In this case, the requesting program is dismissed while the readings are being made and reactivated in normal sequence after the complete block has been treated. Digital input and output are carried out on an immediate basis.

Program Scheduling

The time-sharing program scheduler routine is a software simulation of an interrupt service routine operating at the lowest possible priority, so that its return information always refers to an active program and never to another priority interrupt routine. Future versions of the operating system may actually utilize a hardware interrupt, but the current software simulation was dictated by the initial hardware setup of the object machine. The scheduler is typically triggered by the 60 Hz clock interrupt, and may also be triggered by other interrupts or directly from a time-sharing program which is to be dismissed for any reason. A table of the locations of the Program Definition Areas for all programs is scanned in sequence by the scheduler until a program is found which may be activated. Control is then transferred to this program for a period of time determined by the entry for IQUA in the Program Definition table. If IQUA is negative, it is used as a count of 1/60 second clock ticks to determine the quanta of time assigned to the program. The system standard value of three clock ticks is used if IQUA is positive or zero.

The scheduler also attends to such details as counting down the disuse counters for input-output units even while the system is "idling" by searching for activatable programs. Two exceptions exist to the sequential scan of the program table normally used by the scheduler. First, the scheduler always checks the fast activation location, octal 677, and if it contains a program number, that program is immediately activated, whether or not the current program's time quanta has expired. Since this particular check is made on every interrupt, it provides a method of fast response to an interrupt request. This feature is discussed further under the Set Clock Trap (SCT) system routine description. Second, if a program has been dismissed on a wait for Teletype input, and if the Teletype is operating in keyboard mode, that program will ordinarily be immediately activated when an input character is received so that the echoing action will proceed smoothly. Manual typing is ordinarily erratic enough so that this procedure does not disrupt the smoothness of function of the operating system. The echoing scheme used in this operating system was chosen in spite of the fact that it imposes this requirement because attaining its desirable attributes by other means was found to appreciably increase the amount of code required in the operating system.

Table V lists the possible program status word values and their definitions which dictate the particular test the scheduler must perform in determining whether it can activate a program. Conditions zero through five may be set under program control using the EXIT or EXIA routines; the remaining conditions are set by internal operating system functions.

Interrupt Routine Conventions

While writing special purpose interrupt routines, it is necessary to follow the system conventions which lead to proper triggering of the scheduler interrupt. Interrupt routines must be programmed as follows:

```
ENT     ***     **      Interrupt Entry Point
        IMS     ICTR    (= '571) Reset Interrupt Level Count
        STA     ASAV    Save Registers (and Overflow if it May
        STB     BSAV    be Altered)
      <Interrupt Coding>
        LBA     ICTR    Interrupt Level
        AMB     M1      Minus One
        LAA*    REIU    (= '556) Check Scheduler
        SAZ             In Use
        BRU     E2      No, Try to Use It
E1      STB     ICTR    Cannot be Activated
        LAA     ASAV    Recover Registers
        LBA     BSAV    (Overflow Also if Required)
        TØI
        BRU*    ENT     Return
```

TABLE V

DEFINITIONS OF PROGRAM STATUS WORD VALUES

Number in PSW	Reason for Program Wait
0	Immediately activatable
1	Wait for time period
2	Wait for software flag
3	Wait for sense switch
4	Wait for digital input bit set
5	Program halt
10	Wait for Teletype input
11	Wait for Teletype output
12	Wait for high-speed paper tape input
13	Wait for high-speed paper tape output
14	Wait for Teletype input unit available
15	Wait for Teletype output unit available
16	Wait for high-speed paper tape input unit available
17	Wait for high-speed paper tape output unit available
18	Not currently used
19	Wait for analog multiplexer unit available
20	Wait for analog multiplexer task completion

```
E2      TBA                   Check If At Lowest Level
        SAZ
        BRU      E1           No, Don't Activate
        LAA      ASAV         Recover Registers
        LBA      BSAV         (Overflow Also If Required)
        SPB*     RESC         (= '557) Call Scheduler
        DAC      ENT          Return Link
```

It should be stressed that the above sequence must be used for all interrupts, and that if there is any chance that the overflow might be altered during the interrupt, it must also be saved. This sequence does not ensure that rescheduling will take place, however. Placing a program number in FAC (octal 677) can be used to cause the scheduler to immediately activate that program, or putting a non-zero quantity in location octal 572 will cause rescheduling.

Great care should be given to coding special purpose interrupts, observing all system conventions in order to preserve smooth operation. In particular, with very few exceptions, no use should be made of system routines; if such is required, control should be transferred to a regular, numbered program, with the scheduler locked out if necessary.

Program Panics

A simple but effective system of program panics is provided which can be activated either by using the "ALTMODE" key on the Teletype or through option number six of the system EXIT routine. Whenever the Teletype is in the normal mode and the ALTMODE key is depressed once, the system immediately sets location octal 527 to a non-zero value. This location may be checked by a program and some sort of first order programmer defined panic action taken. If the ALTMODE key is depressed twice in a row, or if the sixth exit option is taken, the system panic routine is initiated, and ESCM in each Program Definition Area is checked to determine the action to be taken for each program. Three actions are possible:

1. ESCM = 0; the program is halted. This is especially useful when debugging the program and provides a means of stopping it.

2. ESCM = -1; the program is activated at the location specified by ELØC in the Program Definition Area. This is especially useful in connection with the above, since a program being debugged may be stopped and a program such as a debugging system may be activated at the same time.

3. ESCM = 1; panics have no effect on the program. Thus, work may progress on programs without altering productive jobs that may be running at the same time.

Restarts

The system has the ability to restart from a power failure. In general, restarting is only of interest over short failures, since timing and sequencing will be seriously in error over longer failures, and in general, the process being monitored will also suffer from the failure. In restarting after a failure, the following occurs:

1. All I/Ø units are reinitialized.

2. All programs which were active (i.e., not waiting under an EXIT option 1-5) are given EXIT option 5 (halt).

3. Location RSTR in the Program Definition Area is examined for each program that is halted (EXIT option 5). If RSTR is zero, the program remains halted, but if not, the program is restarted at the location given in SLØC in the Program Definition Area.

In general, programs which are restarted should contain logic to determine when the failure occurred and how to proceed from that point.

Initialization

The operating system, executive, and desk calculator system are arranged on paper tape to be loaded by the SEL loader. The system is started at location octal 10000, which is the same procedure used in restarting from a power failure.

The Primary Program

In order to have any communication with the system, there must be at least one active program. The system makes special provision for a primary program (numbered minus one) with a Program Definition Area starting at location octal 10200. Most system programs, such as the executive. are written to be primary programs and are usually overlayed whenever another systems program is loaded, using a system procedure which is initiated by branching indirectly to location octal 672. The Teletype prints "OK?" and waits for a confirming response, allowing the new system to be loaded in the high-speed paper tape reader. After the confirming response, the tape is read and control is passed through the starting location (SLØC) of the primary Program Definition Area. System tapes should have all program information first, followed by new values of SLØC and ELØC in the Program Definition Area, which since the program is "active", will be found at octal 700 rather than octal 10200 during the reading. Proper caution should be exercised in changing any of the other variables in the Program Definition Area. Tape reading errors or panics during tape reading will cause the message "RELOAD TAPE" to be typed out, followed by "OK?" and another confirmation wait.

Floating Point Arithmetic

Floating point numbers each require three words of storage. The first word is used as a two's complement binary exponent, and the second and third words are used as a normalized double word two's complement binary fraction. (The sign bit of the third word must be zero and is maintained as zero by the floating point routines. Use of quantities which have the sign of the third word set or which are not normalized may cause a variety of erroneous results if used as arguments for the system routines.) This particular three-word format was chosen as a compromise between speed and precision: A two-word format with part of one word used as an exponent would have only about six decimal digits of precision while if a three-word format were chosen using part of a word as an exponent, the actual arithmetic would have to be performed on a fraction extending over three words, and would be much slower than arithmetic on a two-word fraction. The chosen format preserves slightly more than nine decimal digits of precision, and as a side effect, allows representation of numbers with absolute values in the range of about $10^{\pm 9864}$. Excessive use of numbers pressing the limits of this range, of course, is neither to be expected nor encouraged. Binary to decimal conversion times for such huge quantities, for example, approach a full second, using system routines,and full nine-digit accuracy is not maintained.

A "floating point accumulator" is maintained in octal locations 730-732, and using system routines, this accumulator is loaded, stored, added to, etc., in much the same way as a true hardware accumulator would be used. System routine calls which specify the location of a second operand may use indirect addressing and indexing on that operand, and if indexing is used, the index (B register) value is multiplied by three before use in obtaining the final argument address to account for the three-word floating point format. Any intermediate indexing used to obtain an address, however, does not cause this multiplication by three, so that indexed tables of addresses may also be used.

Floating point overflows and other errors are normally handled by zeroing the floating point accumulator and proceeding, and no well-defined distinction is made among the various possible sources of error. Provision has been made, however, for programming to detect the presence of such errors in that each causes an indirect branch to one of four locations in Map 0, as shown in table VI. Each of the four locations has a corresponding location in which the return address for the routine in error is stored. The corresponding value of the index (B) register will be found in location octal 727. For error detection, the vector locations may be redefined to point to programmer supplied routines, although care should be exercised to preserve the re-entrant qualities of the system.

String Processing System

Several special definitions are used in this system:

Character: Any eight-bit quantity, such as an ASCII code representation.

TABLE VI

FLOATING POINT ERROR TRANSFER VECTORS

Vector Location	Return Address Location	Possible Sources of Error
544	736	FAD,FSB,FSBI,FMP,FDV,FDVI
545	737	NØRM,FNEG,FABS
546	743	EXP,SQRT,LN
547	757	FPW,FPWI

Character Address: Characters are stored two per 16-bit word, and the address of a character is thus two times the word address for the left-hand character in a word and two times the word address plus one for the right-hand character.

String: A string is any group of characters stored in sequential character addresses.

String Pointer: A string pointer is a pair of character addresses. The first address is that of the character immediately <u>preceding</u> the first character in the string, and the second address is that of the last character in the string.

Null String: Any string for which the two character addresses in the string pointer are equal.

Use of the above definitions in a set of routines provides a system in which character addresses are usually manipulated rather than the characters themselves. A single string, for example, may easily have several string pointers referring to it in the same program, each pointer operating on a different portion. Routines are provided for string-numeric conversion and string input-output, as well as string creation and testing so that elaborate formating of both input and output puts little strain on the programmer.

Timekeeping

The 60 Hz clock is used to maintain a double word seconds counter in locations octal 576 and 577. This word pair may be referenced at any time, and if properly initialized, will give the seconds elapsed from the previous midnight. The counter is not re-zeroed at midnight, but is allowed to keep running in order to provide operating continuity. The system time conversion routine, however, operates modulo the 24-hour day, and thus gives the appearance that the counter has been zeroed. Location octal 575 contains the address of the 60 Hz counter. This counter is started at minus 60 and incremented until zero each second. It may be referenced along with the seconds counter to more accurately determine or set time intervals.

SYSTEM ROUTINE DESCRIPTIONS

The following descriptions provide information necessary to use the system utility routines in assembly language coding and, additionally, information concerning intersystem calls and re-entrant storage usage for documentation purposes. For use with the SEL assembler, it is expedient to enter definitions of the system transfer vectors which are to be used; i.e.,

```
LFA     EQU     '600
SFA     EQU     '601
```

The system assembler, however, has system definitions built in, usually in the form of additional operation codes. Parameter names used in the descriptions refer to locations in the Program Definition Area.

FLOATING POINT ARITHMETIC ROUTINES

Forms which permit indexing multiply the contents of the B register by three prior to use if the indexing occurs in determining the final argument address. Indexing of intermediate indirect addresses uses the unmodified content of the B register. Naturally, indirectness and indexing need not be specified if not desired. Timing figures for these routines are based upon a series of actual tests and are, of course, variable.

LFA - Load Floating Point Accumulator

TRANSFER VECTOR LOCATION: 600

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	LFA	LFA*	ADDR,1
DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The three-word quantity starting at the referenced address is copied into the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, ITMP

SYSTEM ROUTINES USED: FSET

LOCATIONS INDIRECTLY ALTERED: UO, Ul

TIMING: 0.10 ms

SFA - Store Floating Point Accumulator

TRANSFER VECTOR LOCATION: 601

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	SFA	SFA*	ADDR,1
DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The contents of the floating point accumulator are stored into three words starting at the referenced address.

LOCATIONS DIRECTLY ALTERED: ITMP

SYSTEM ROUTINES USED: FSET

LOCATIONS INDIRECTLY ALTERED: UO, Ul

TIMING: 0.10 ms

FAD - Floating Point Add

TRANSFER VECTOR LOCATION: 602

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FAD	FAD*	ADDR,1
	DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The floating point quantity referenced is added to the floating point accumulator, the sum replacing the contents of the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, TR, ITMP, U1-U3

SYSTEM ROUTINES CALLED: FSET, LTR, IATR, NØRM

LOCATIONS INDIRECTLY ALTERED: U0, U32

ERROR CONDITIONS POSSIBLE: Zero returned for floating point addition overflow.

TIMING: 0.30 ms typical

FSB - Floating Point Subtract

TRANSFER VECTOR LOCATION: 603

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FSB	FSB*	ADDR,1
	DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The referenced floating point quantity is subtracted from the floating point accumulator, the difference replacing the contents of the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, TR, ITMP, U1-U3

SYSTEM ROUTINES CALLED: FSET, LTR, IATR, NØRM, FNEG

LOCATIONS INDIRECTLY ALTERED: U0, U32

ERROR CONDITIONS POSSIBLE: Zero returned for floating point subtraction overflow.

TIMING: 0.40 ms typical

FSBI - Floating Point Subtract Inverse

TRANSFER VECTOR LOCATION: 604

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FSBI	FSBI*	ADDR,1
	DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The floating point accumulator is subtracted from the referenced quantity, the difference replacing the contents of the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, TR, ITMP, U1-U3

SYSTEM ROUTINES CALLED: FSET, LTR, IATR, NØRM, FNEG

LOCATIONS INDIRECTLY ALTERED: U0, U32

ERROR CONDITIONS POSSIBLE: Zero returned for floating point subtraction overflow.

TIMING: 0.40 ms typical

FMP - Floating Point Multiply

TRANSFER VECTOR LOCATION: 605

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FMP	FMP*	ADDR,1
	DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The contents of the floating point accumulator are multiplied by the referenced quantity, the product replacing the contents of the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, TR, ITMP, U1-U4

SYSTEM ROUTINES CALLED: FSET, LTR, FNEG, NØRM

LOCATIONS INDIRECTLY ALTERED: U0, U32

ERROR CONDITIONS POSSIBLE: Zero returned for floating point multiply overflow.

TIMING: 0.45 ms typical

FDV - Floating Point Divide

TRANSFER VECTOR LOCATION: 606

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FDV	FDV*	ADDR,1
	DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The contents of the floating point accumulator are divided by the contents of the referenced location, the quotient replacing the contents of the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, TR, ITMP, U1, U3, U4

SYSTEM ROUTINES CALLED: FSET, LTR, FNEG, NØRM

LOCATIONS INDIRECTLY ALTERED: UO, U2, U32

ERROR CONDITIONS POSSIBLE: Zero returned for floating point divide overflow.

TIMING: 0.50 ms typical

FDVI - Floating Point Inverse Divide

TRANSFER VECTOR LOCATION: 607

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FDVI	FDVI*	ADDR,1
	DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The contents of the referenced address are divided by the floating point accumulator, the quotient replacing the contents of the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, TR, ITMP, U1, U3, U4

SYSTEM ROUTINES CALLED: FSET, LTR, FNEG, NØRM, IATR

LOCATIONS INDIRECTLY ALTERED: UO, U2, U32

ERROR CONDITIONS POSSIBLE: Zero returned for floating point divide overflow.

TIMING: 0.50 ms typical

FPW - Floating Point Power Function

TRANSFER VECTOR LOCATION: 614

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FPW	FPW*	ADDR,1
	DAC*	ADDR,1		

REGISTER TREATMENT: A destroyed, B saved

FUNCTION: The contents of the floating point accumulator are raised to the power contained as a floating point quantity at the referenced address, the result replacing the contents of the floating point accumulator. This operation may take place by multiplication for integer valued powers less than 16, or through the exponential-log transform otherwise.

LOCATIONS DIRECTLY ALTERED: ITMP, U17-U21

SYSTEM ROUTINES CALLED: FSET, LTR, IATR, SFA, FIX, FLØT, FNEG, FABS, LN, EXP, LFA, FAD, FSB, FSBI, FMP, FDV, FDVI

LOCATIONS INDIRECTLY ALTERED: FA, TR, U0-U16, U32

ERROR CONDITIONS POSSIBLE: Zero is returned for power function error.
For A ↑ B, the error conditions are:
A = 0, B < 0
A < 0, B not integer valued
Notes: 0 ↑ 0 is taken to be 1.0, and for negative numbers raised to odd integer powers, the result is negative.

TIMING: 1.0 - 6.0 ms for integer powers, 14 ms typical if exponential and logarithm required.

FPWI - Floating Point Power Function

TRANSFER VECTOR LOCATION: 615

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER	
	SPB*	FPWI	FPWI*	ADDR,1
	DAC*	ADDR,1		

FUNCTION: The referenced argument is raised to the power contained in the floating point accumulator, the result replacing the contents of the floating point accumulator. Note: The notations for FPW apply to FPWI.

FLØT - Float an Integer Quantity

TRANSFER VECTOR LOCATION: 610

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
LAA <integer>	LAA <integer>
SPB* FLØT	FLØT

REGISTER TREATMENT: A and B registers both destroyed

FUNCTION: The single word integer quantity in the A register is converted to a floating point quantity in the floating point accumulator.

LOCATIONS DIRECTLY ALTERED: FA, UO

SYSTEM ROUTINES CALLED: NØRM

LOCATIONS INDIRECTLY ALTERED: U1, U2

TIMING: 0.10 ms typical

FIX - Integer and Fraction from Floating Point Quantity

TRANSFER VECTOR LOCATION: 611

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
SPB* FIX	FIX

REGISTER TREATMENT: A and B registers destroyed, A returns the resultant integer.

FUNCTION: The contents of the floating point accumulator are reduced to two quantities: an integer returned in the A register and a fraction replacing the contents of the floating point accumulator. The integer returned is the signed least significant 15 bits of the integer part of the original floating point quantity. A few examples illustrate the mode of operation:

Original Quantity	Integer	Fraction
+ 62.35	+ 62	+ 0.35
- 62.35	- 62	- 0.35
- 0.062	+ 0	- 0.062
100000.78	+ 1696	0.78

LOCATIONS DIRECTLY ALTERED: FA, UO-U4

SYSTEM ROUTINES CALLED: FABS, FNEG, NØRM

LOCATIONS INDIRECTLY ALTERED: None

TIMING: 0.24 ms typical

FNEG - Negate Floating Accumulator

TRANSFER VECTOR LOCATION: '612

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER
SPB*	FNEG	FNEG

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The contents of the floating accumulator are replaced by the same value with opposite sign.

LOCATIONS DIRECTLY ALTERED: U1, U2, FA, U32

ERROR CONDITIONS POSSIBLE: A zero is returned upon exponent overflow during conversion.

TIMING: 0.08 ms typical

FABS - Absolute Value of Floating Accumulator

TRANSFER VECTOR LOCATION: '613

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER
SPB*	FABS	FABS

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The contents of the floating point accumulator are made positive.

LOCATIONS DIRECTLY ALTERED: U1, U2, FA, U32

ERROR CONDITIONS POSSIBLE: A zero is returned upon exponent overflow during conversion.

TIMING: 0.02 ms positive, 0.08 ms negative typical

EXP - Exponential Function of Floating Accumulator

TRANSFER VECTOR LOCATION: '616

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER
SPB*	EXP	EXP

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The exponential of the contents of the floating point accumulator replaces the contents of the floating point accumulator. Calculation of the exponential is done using a reduced form of the continued fraction expansion of exp(x).

LOCATIONS DIRECTLY ALTERED: FA, U5-U16

SYSTEM ROUTINES CALLED: FABS, LFA, SFA, FAD, FSB, FSBI, FMP, FDV, FDVI, FIX, FSET, LTR, IATR, FNEG

LOCATIONS INDIRECTLY ALTERED: TR, UO-U4, U32

ERROR CONDITIONS POSSIBLE: Zero is returned if the initial value of the floating point accumulator is greater than 22712.

TIMING: 5.0 ms typical

LN - Natural Logarithm of Floating Point Accumulator

TRANSFER VECTOR LOCATION: 617

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER
SPB*	LN	LN

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The natural logarithm of the contents of the floating point accumulator replaces the contents of the floating point accumulator. Calculation of the logarithm is done using a reduced form of the continued fraction expansion for $\ell n[(1 + x)/(1 - x)]$

LOCATIONS DIRECTLY ALTERED: FA, U5-U12

SYSTEM ROUTINES CALLED: LFA, SFA, FAD, FSB, FMP, FDV, FDVI, FLØT, FSET, LTR, IATR, FNEG

LOCATIONS INDIRECTLY ALTERED: TR, UO-U4, U32

ERROR CONDITIONS POSSIBLE: Zero is returned if the initial value of the floating point accumulator is zero or negative.

TIMING: 7.0 ms typical

SQRT - Square Root of Floating Point Accumulator

TRANSFER VECTOR LOCATION: 620

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER
SPB*	SQRT	SQRT

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The square root of the contents of the floating point accumulator replaces the contents of the floating point accumulator. Calculation of the square root involves an initial fractional approximation followed by three refinement passes through a Newton's method iteration.

LOCATIONS DIRECTLY ALTERED: FA, U5-U13

SYSTEM ROUTINES CALLED: SFA, FAD, FSBI, FDVI, FSET, LTR, IATR, FNEG

LOCATIONS INDIRECTLY ALTERED: TR, UO-U4, U32

ERROR CONDITIONS POSSIBLE: Zero is returned if the initial value of the floating point accumulator is negative.

TIMING: 3.1 ms typical

FLC - Three-Way Floating Point Comparison

TRANSFER VECTOR LOCATION: 626

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	FLC	FLC*	ADDR, 1
DAC*	ADDR, 1	BRU	<LESS>
BRU	<LESS>	BRU	<EQUAL>
BRU	<EQUAL>	BRU	<GREATER>
BRU	<GREATER>		

REGISTER TREATMENT: A register destroyed, B register saved

FUNCTION: The contents of the referenced location are compared with the contents of the floating point accumulator. If greater, two locations are skipped; if equal, one location is skipped, and if less, no locations are skipped. This provides a means for faster comparison without destroying the contents of the floating accumulator.

LOCATIONS DIRECTLY ALTERED: TR, ITMP, UO

SYSTEM ROUTINES CALLED: FSET, LTR

LOCATIONS INDIRECTLY ALTERED: UO-U2

TIMING: 0.16 ms typical

FLCZ - Three Way Floating Point Compare with Zero

TRANSFER VECTOR LOCATION: 627

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	FLCZ	FLCZ	
BRU	<NEGATIVE>	BRU	<NEGATIVE>
BRU	<ZERO>	BRU	<ZERO>
BRU	<POSITIVE>	BRU	<POSITIVE>

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: If the contents of the floating point accumulator are positive, two locations are skipped; if zero, one location is skipped; and if negative, no locations are skipped.

LOCATIONS DIRECTLY ALTERED: None

SYSTEM ROUTINES CALLED: None

TIMING: 0.02 ms

DFIX - Fix Double Word Quantity in Floating Point Accumulator

TRANSFER VECTOR LOCATION: 650

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER
SPB*	DFIX	DFIX

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The floating point accumulator value is converted to a double word integer which replaces the second and third words of the floating point accumulator. This routine is specifically designed to aid in dealing with double integer time values. Unlike FIX, all quantities are converted to the largest integer value not greater than the original value. Thus, 1.2 is converted to 1, while minus 1.2 is converted to minus 2.

LOCATIONS DIRECTLY ALTERED: FA, UO, U1

FDBL - Float Double Word Integer in Floating Point Accumulator

TRANSFER VECTOR LOCATION: 647

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER
SPB*	FDBL	FDBL

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The double word integer contained in the second and third words of the floating point accumulator is converted to a floating point quantity replacing the contents of the floating point accumulator. Like DFIX, this routine is primarily intended for dealing with double integer time values.

LOCATIONS DIRECTLY ALTERED: FA, UO

SYSTEM ROUTINES CALLED: NØRM

LOCATIONS INDIRECTLY ALTERED: U1, U2

LDI - Load Double Integer

TRANSFER VECTOR LOCATION: 667

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	LDI	LDI*	A
DAC*	A		

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The double word integer quantity located in the two words starting at the specified address is loaded into the floating point accumulator as a floating point quantity.

LOCATIONS DIRECTLY ALTERED: FA

SYSTEM ROUTINES CALLED: SET, FDBL

LOCATIONS INDIRECTLY ALTERED: UO-U2

SDI - Store Double Integer

TRANSFER VECTOR LOCATION: 670

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	SDI	SDI*	A
DAC*	A		

REGISTER TREATMENT: A and B destroyed; FA preserved

FUNCTION: The contents of the floating point accumulator are converted to a double-word integer and stored in the two words starting at the specified address. The contents of the floating accumulator are restored.

LOCATIONS DIRECTLY ALTERED: U2-U5

SYSTEM ROUTINES CALLED: SFA, SET, DFIX, LFA

LOCATIONS INDIRECTLY ALTERED: UO, U1

FSET - System Indexed Argument Fetch

TRANSFER VECTOR LOCATION: 564

CALL SEQUENCE:

```
STB     ITMP
LAA     <address of argument>
SPB*    FSET
```

FUNCTION: System routine to obtain the location of a floating point argument. It is entered with the address, in the A register, of a location which should contain the address of the argument. FSET handles any indirectness or indexing which may be present in this address, returning the true argument address in the B register. Additionally, the initial contents of the A register plus one are stored into UO, normally to serve as the return pointer for the calling program.

LOCATIONS DIRECTLY ALTERED: UO, U1

SET - System Argument Fetch

TRANSFER VECTOR LOCATION: 555

CALL SEQUENCE: LBA <address of argument address>
SPB* SET

FUNCTION: Differs from FSET only in that indexing is ignored and the initial adress is transmitted through the B register.

LOCATIONS DIRECTLY ALTERED: UO, U1

LTR - Load Floating Point Transfer Register

TRANSFER VECTOR LOCATION: 554

CALL SEQUENCE: LBA <argument address>
SPB* LTR

FUNCTION: The contents of the B register are used to obtain a three-word quantity which is stored into the floating point transfer register. The call to LTR usually immediately follows a call to FSET, and it is used to obtain the second argument to binary floating point operations.

LOCATIONS DIRECTLY ALTERED: U1, TR

IATR - Interchange Floating Point Accumulator and Transfer Register

TRANSFER VECTOR LOCATION: 553

CALL SEQUENCE: SPB* IATR

FUNCTION: The contents of the floating point accumulator are exchanged with the contents of the floating point transfer register.

LOCATIONS DIRECTLY ALTERED: FA, TR, U1

NØRM - Normalize Floating Point Accumulator

TRANSFER VECTOR LOCATION: 552

CALL SEQUENCE: SPB* NØRM

FUNCTION: The quantity in the floating point accumulator is normalized.

LOCATIONS DIRECTLY ALTERED: FA, U1, U2

CFAS - Convert Floating Point Accumulator to String

See string processing system section.

CSFA - Convert String to Floating Point Accumulator

See string processing system section.

CMFV - Convert Multiplexer Reading to Floating Point Voltage

See input-output section.

TTS - Convert Time to String

See string processing system section.

STTI - Convert String to Time

See string processing system section.

INPUT-OUTPUT SYSTEM ROUTINES

TCI - Teletype Character Input

TRANSFER VECTOR LOCATION: 630

CALL SEQUENCE:	SEL ASSEMBLER		SYSTEM ASSEMBLER
	SPB*	TCI	TCI

REGISTER TREATMENT: A and B registers destroyed.

FUNCTION: The next character from the Teletype is input according to the prevailing mode of operation. The character is in the least significant 8 bits of the A register upon exit from the routine. The calling program will be dismissed on a wait if either no input is available or Teletype input is in use by another program.

LOCATIONS DIRECTLY ALTERED: U20-U22

SYSTEM ROUTINES CALLED: TCØ, TPT

LOCATIONS INDIRECTLY ALTERED: U24-U26

TCØ - Teletype Character Output

TRANSFER VECTOR LOCATION: 631

CALL SEQUENCE:	SEL ASSEMBLER	SYSTEM ASSEMBLER
	LAA <character>	LAA <character>
	SPB* TCØ	TCØ

REGISTER TREATMENT: A register saved, B register destroyed

FUNCTION: The least significant 8 bits of the A register are output as a character to the Teletype according to the prevailing mode of operation. The calling program will be dismissed on a wait if either the output buffer is full or Teletype output is in use by another program.

LOCATIONS DIRECTLY ALTERED: U24-U26, U21

PCI - High-Speed Paper Tape Character Input

TRANSFER VECTOR LOCATION: 632

CALL SEQUENCE:	SEL ASSEMBLER	SYSTEM ASSEMBLER
	SPB* PCI	PCI

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: Obtain characters from high-speed paper tape in the same manner as TCI from Teletype.

LOCATIONS DIRECTLY ALTERED: U20, U21

PCØ - High-Speed Paper Tape Character Output

TRANSFER VECTOR LOCATION: 633

CALL SEQUENCE:	SEL ASSEMBLER	SYSTEM ASSEMBLER
	LAA <character>	LAA <character>
	SPB* PCØ	PCØ

REGISTER TREATMENT: A register saved, B register destroyed

FUNCTION: Transmit characters to high-speed paper tape in the same manner as TCØ for Teletype.

UNIT - Convert Unit Number to Address

TRANSFER VECTOR LOCATION: 666

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<unit number>	LAA	<unit number>
SPB*	UNIT	UNIT	
BRU	<illegal unit>	BRU	<illegal unit>
BRU	<output unit>	BRU	<output unit>
BRU	<input unit>	BRU	<input unit>

FUNCTION: Determine unit from unit number, and skip as indicated according to results. The unit numbers are:

0 - TCI
1 - TCØ
2 - PCI
3 - PCØ

UNIT returns with the address of the indicated input-output routine in the A register, and the unit number in the B register.

LOCATIONS DIRECTLY ALTERED: U14, U15

STAS - Set Status of Unit

TRANSFER VECTOR LOCATION: 652

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<unit number>	LAA	<unit number>
SPB*	STAS	STAS	
DATA	<status word>	DATA	<status word>

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: To set the status of the input-output routine and determine the manner in which characters will be pretreated. The format of the status word is presented in table IV of the system general description.

LOCATIONS DIRECTLY ALTERED: Unit Status Word, U1, U2

CLRU - Clear Input-Output Unit

TRANSFER VECTOR LOCATION: 671

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<unit number>	LAA	<unit number>
SPB*	CLRU	CLRU	

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: Clears the specified unit by clearing its buffer and re-initializing any software and hardware parameters associated with it. Using normal unit numbers, the calling program will wait if another program is currently using the unit. If CLRU is called with an argument of the unit number plus 10 instead of just the unit number, clearing is performed without this wait, providing a means of error recovery. An argument of 14 clears the multiplexer without waiting.

LOCATIONS DIRECTLY ALTERED: U6, U7, U21, unit parameters

<u>PBT - Punch Binary Tape</u>

TRANSFER VECTOR LOCATION: 664

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<direction word>	LAA	<direction word>
SPB*	PBT	PBT	
DAC*	<first address>	DAC*	<first address>
DAC*	<last address>	DAC*	<last address>
BRU	<error>	BRU	<error>

REGISTER TREATMENT: A and B registers destroyed.

FUNCTION: Permits punching of absolute binary tapes which are compatible with the SEL load-dump routines in one mode and numeric binary tapes in a second mode. The "direction word" contains the unit number (1 or 3) in its least significant bits and indicates the action to be taken using bit 0 (sign) and bit 1.
<u>Absolute Mode</u>: Sign bit not set in direction word. A standard absolute binary tape is punched. The system is capable of recognizing two types of absolute dump records during loading (see RBT). The first type, indicated by having bit 1 not set in the direction word, is standard, while the second type, indicated by having bit 1 set, is recognized by the system loader as being only one record of a series to be loaded at the same time. Thus, if it is desired to absolute dump four different areas of memory which are all to be later loaded together, the first three areas should all be dumped with bit 1 set in the direction word, and the last area with bit 1 not set. The resulting record can be read with one call to RBT. Both types of records will work (one at a time) with the SEL loader, and the final record mode automatically causes a trailer to be punched.
<u>Numeric Mode</u>: Sign bit set in direction record. Produces a binary record which is not tied to a particular loading address. Can be read into any area of memory by the system loader. This mode is intended to allow binary output of blocks of numbers, as opposed to code, and is not compatible with the SEL loader.

LOCATIONS DIRECTLY ALTERED: U0, U3-U9

SYSTEM ROUTINES CALLED: SET, UNIT, STAS, TCØ, PCØ

LOCATIONS INDIRECTLY ALTERED: Unit status word, U1, U2, U20-U26

RBT - Read Binary Tape

TRANSFER VECTOR LOCATION: 665

```
CALL SEQUENCE:            SEL ASSEMBLER                SYSTEM ASSEMBLER
                          LAA   <direction word>       LAA   <direction word>
                          SPB*  RBT                    RBT
  (if numeric option)     DAC*  <load address>         DAC*  <load address>
                          BRU   <load error>           BRU   <load error>
```

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: Load binary tapes produced either by PBT or the SEL absolute dump routine. The "direction word" contains the unit number (0 or 2) in the least significant bits and the mode according to the sign bit.
Absolute Mode: Sign bit not set in direction word. Loads an absolute binary dump tape or series of dumps (see PBT).
Numeric Mode: Sign bit set in direction word. Loads a tape prepared by PBT in numeric mode, starting at the load address supplied as an argument.

LOCATIONS DIRECTLY ALTERED: UO, U3-U9

SYSTEM ROUTINES CALLED: SET, UNIT, STAS, TCI, PCI

LOCATIONS INDIRECTLY ALTERED: Unit status word, U1, U2, U20-U23

LDR - Leader to Unit

TRANSFER VECTOR LOCATION: 641

```
CALL SEQUENCE:  SEL ASSEMBLER                        SYSTEM ASSEMBLER
                LAA   <unit number>                  LAA  <unit number>
                SPB*  LDR                            LDR
```

REGISTER TREATMENT: A register saved, B register destroyed

FUNCTION: Blank leader will be punched on the designated unit, one hundred characters on Teletype and six hundred on high-speed paper tape.

LOCATIONS DIRECTLY ALTERED: U1-U4

SYSTEM ROUTINES CALLED: TCØ, PCØ

LOCATIONS INDIRECTLY ALTERED: U20-U26

TEL - Set Teletype to Key Mode

TRANSFER VECTOR LOCATION: 562

CALL SEQUENCE: SPB* TEL

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The teletype is put in keyboard mode. Caution. No check is made to see if the calling program has control over the unit, and due care should be taken.

LOCATIONS DIRECTLY ALTERED: None

TPT - Set Teletype to Paper Tape Reading Mode

TRANSFER VECTOR LOCATION: 561

CALL SEQUENCE: SPB* TPT

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The Teletype is put in paper tape reading mode. See caution under TEL.

LOCATIONS DIRECTLY ALTERED: None

HIØF - Turn High-Speed Paper Tape Reader Off

TRANSFER VECTOR LOCATION: 560

CALL SEQUENCE: SPB* HIØF

FUNCTION: The high-speed paper tape reader is turned off. See caution under TEL.

LOCATIONS DIRECTLY ALTERED: None

MUX - Input Block of Multiplexer Readings

TRANSFER VECTOR LOCATION: 661

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
SPB* MUX	MUX
DAC* <addr of direction>	DAC* <addr of direction>
DAC* <addr of result>	DAC* <addr of result>

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: To input a block of readings from the analog multiplexer. The first argument is the address of a block of storage having a (negative) count of the number of readings to be taken, and the remainder of the block being direction words for each reading in the form:

```
Count      Gain    Channel
|_|_|_|_|  |_|_|_| |_|_|_|_|_|_|_|_|_|
 0     3    4   6   7               15
```

Count: The number of readings to be taken and averaged to form a single result, minus one. Thus, a count of zero causes only one reading to be made.
Gain: The gain range to be used to make the reading:

- 0 - ± 5v
- 1 - ± 200 mv
- 2 - ± 100 mv
- 3 - ± 50 mv
- 4 - auto-ranging mode
- 5 - ± 20 mv
- 6 - ± 10 mv
- 7 - ± 5 mv

Channel: The number of the analog channel.

The auto-ranging function first causes two readings to be taken on the 5v range, R1 and R2, and then the function:

Q = ABS(MIN(R1,R2)) + 3* ABS(R1-R2)

is computed. If Q is greater than 160 (out of 4095), the 5-volt range is chosen. Otherwise, a new set of readings, R1' and R2', is taken on the 200 mv range, and a new value of Q computed which is used to select the most sensitive range in which less than a full-scale reading should be obtained. The selected range is then used to make a series of readings as indicated by the averaging count.

The second argument of MUX is the address of a block in which the results are to be placed. The first two words of this block are used to store the value of the clock at the start of the reading sequence, and the remaining words are used to sequentially store the results of the multiplexer readings in standard form:

```
Result                    Gain
|_______________________| |_____|
 0                    12   13  15
```

Result: Signed two's complement integer result from reading the multiplexer (and possibly averaging).
Gain: The gain range, as above, which was used in making the reading.

The entire block of readings will be made one at a time at 100 readings per second, and the multiplexer is unavailable for use by other programs during this period (any requesting programs will be dismissed until the multiplexer is finished). Since the readings are handled through an interrupt structure, all other time-shared programs will function normally, however. The direction and result blocks may overlap if desired so long as a result will not be stored over a direction word that has not been used yet.

LOCATIONS DIRECTLY ALTERED: U0, U4-U6

SYSTEM ROUTINES CALLED: SET

LOCATIONS INDIRECTLY ALTERED: U1, U2

MXSV - Single Multiplexer Reading to A Accumulator

TRANSFER VECTOR LOCATION: 660

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
LAA <direction word>	LAA <direction word>
SPB* MXSV	MXSV

REGISTER TREATMENT: A and B return results

FUNCTION: To obtain a single reading from the analog multiplexer. The routine is entered with a direction word in the A register and returns with a standard result word in the A register and the address of a two-word pair containing the corresponding clock reading in the B register. See MUX for further details.

LOCATIONS DIRECTLY ALTERED: U10-U15

SYSTEM ROUTINES CALLED: MUX, SET

LOCATIONS INDIRECTLY ALTERED: U0-U2, U4-U6

CMFV - Standard Multiplexer Result Word to Floating Point Voltage

TRANSFER VECTOR LOCATION: 651

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
LAA <MUX result>	LAA <MUX result>
SPB* CMFV	CMFV

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: Convert a standard multiplexer result word (see MUX) entered through the A register, to the equivalent voltage in the floating point accumulator. Due to the peculiarity that the multiplexer result range is not symmetric (-4096 to +4095), a full-scale negative will yield minus 5.0 volts, for example, while a full-scale positive will yield 5.0 - 1/4096 = 4.9987 volts.

LOCATIONS DIRECTLY ALTERED: U12-U16, FA

SYSTEM ROUTINES CALLED: FLØT, FMP, NØRM, FSET, LTR, FNEG

LOCATIONS INDIRECTLY ALTERED: ITMP, TR, UO-U4, U32

RDIN - Read Digital Input Unit

TRANSFER VECTOR LOCATION: 655

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<unit number>	LAA	<unit number>
SPB*	RDIN	RDIN	

REGISTER TREATMENT: A register returns results, B register destroyed.

FUNCTION: To read the setting of a digital input unit. The unit number is 0 or 1 to select one of the two available units, and the reading is made immediately.

LOCATIONS DIRECTLY ALTERED: UO, U1

DØDI - Digital Word Out Direct Mode

TRANSFER VECTOR LOCATION: 654

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<unit number>	LAA	<unit number>
SPB*	DØDI	DØDI	
DATA	<output word>	DATA	<output word>
DATA	<mask>	DATA	<mask>

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The unit number (0-3) selects one of the four digital output units, altering those bits selected with ones in the mask word so that they are the same as the corresponding bits in the output word argument. Unselected bits are not altered. When needed, the (indexed) address of the first word of a table containing the current status of each unit is found in location octal 541. Two programs using the same unit will cause no unexpected conflicts.

LOCATIONS DIRECTLY ALTERED: U4-U6

SYSTEM ROUTINES CALLED: SET

LOCATIONS INDIRECTLY ALTERED: UO-U2

DØIN - Digital Word Out Indirect

TRANSFER VECTOR LOCATION: 653

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<unit number>	LAA	<unit number>
SPB*	DØIN	DØIN	
DAC*	<addr of output word>	DAC*	<addr of output word>
DAC*	<addr of mask>	DAC*	<addr of mask>

FUNCTION: Provide an address mode of DØDI. See DØDI for remaining information.

MESG - Message on Output Unit

See string processing section.

SØUT - String to Output Unit

See string processing section.

SIU - String in From Unit

See string processing section.

STRING PROCESSING ROUTINES

In the following descriptions, "S" is considered to be the first address of a pair of string pointers (see section on "String Processing System"). Indirectness, where indicated, is optional in locating this address.

PAD - String Pointers from Word Address

TRANSFER VECTOR LOCATION: 640

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<word address>	LAA	<word address>
SPB*	PAD	PAD*	S
DAC*	S		

REGISTER TREATMENT: A and B registers destroyed.

FUNCTION: The word address in the A register is converted to a pair of string pointer character addresses pointing to a null string beginning at the word address. This pointer pair initializes "S".

LOCATIONS DIRECTLY ALTERED: U3

SYSTEM ROUTINES CALLED: SET

LOCATIONS INDIRECTLY ALTERED: UO-U2

WCI - Write Character and Increment

TRANSFER VECTOR LOCATION: 635

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
LAA <character>	LAA <character>
SPB* WCI	WCI* S
DAC* S	

REGISTER TREATMENT: A register saved, B register destroyed

FUNCTION: The character in the least significant 8 bits of the A register is written on to the end of the string indicated by the pointers "S", and the second pointer is incremented. (The other half of the object word is not altered.)

LOCATIONS DIRECTLY ALTERED: U6-U8

GCI - Get Character and Increment

TRANSFER VECTOR LOCATION: 634

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
SPB* GCI	GCI* S
DAC* S	BRU <empty string>
BRU <empty string>	

REGISTER TREATMENT: A register returns result, B register destroyed

FUNCTION: The first character in the string referenced by "S" is returned in the least significant 8 bits of the A register, and the first pointer is incremented. Skips unless string is empty.

LOCATIONS DIRECTLY ALTERED: U6, U7

SØUT - String Out to Unit

TRANSFER VECTOR LOCATION: 643

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
LAA <unit number>	LAA <unit number>
SPB* SØUT	SØUT* S
DAC* S	

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The referenced string is output to the Teletype or high-speed paper tape unit as indicated by the unit number (1 or 3). The string pointers are preserved. Note that SØUT does not print a carriage return at the end of the string unless one is actually present.

LOCATIONS DIRECTLY ALTERED: U3-U5, U9

SYSTEM ROUTINES CALLED: SET, UNIT, GCI, TCØ, PCØ

LOCATIONS INDIRECTLY ALTERED: UO-U2, U6, U7, U14, U15, U20-U26

<u>MESG - Message Out to Unit</u>

TRANSFER VECTOR LOCATION: 642

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
LAA <unit number>	LAA <unit number>
SPB* MESG	MESG* A
DAC* A	

REGISTER TREATMENT: A and B registers destroyed.

FUNCTION: Output a message to the unit indicated by the unit number (1 or 3). "A" is the word address of the first word of the message (not the address of string pointers). Two special characters are recognized: "$" will print a carriage return-like feed pair, and "/" denotes the end of the message. Maximum message length is 720 characters.

LOCATIONS DIRECTLY ALTERED: U3-U5, U9

SYSTEM ROUTINES CALLED: SET, UNIT, GCI, TCØ, PCØ

LOCATIONS INDIRECTLY ALTERED: UO-U2, U6, U7, U14, U15, U20-U26

<u>SIU - String in from Unit</u>

TRANSFER VECTOR LOCATION: 644

CALL SEQUENCE:

SEL ASSEMBLER	SYSTEM ASSEMBLER
LAA <unit number>	LAA <unit number>
SPB* SIU	SIU* S
DAC* S	BRU <control D exit>
BRU <control D exit>	

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: <u>Normal Mode</u>: To input a string from the unit indicated by the unit number (0 or 2). The string is nulled prior to use, and the editing characters A^C (delete previous character) and Q^C (delete entire line) are handled automatically. SIU exits upon scanning either a carriage return or a control D, the control D taking the indicated special exit. The carriage return or control D will be in the A register upon exit, but neither is actually entered at the end of the string. If the sign bit is set in the A register upon entry, spaces will be converted to and treated like carriage returns.

Special Mode: If the bit next to the sign bit in the A register is set upon entry, a special edit mode is used. In this mode, the string is not nulled prior to use, and SIU exits at its mark upon scanning any control character other than A^c and Q^c, allowing the calling program to interpret the control.

LOCATIONS DIRECTLY ALTERED: U0, U3-U5, U9-U11

SYSTEM ROUTINES CALLED: UNIT, SET, WCI, TCI, PCI, TCØ, CIN

CSFA - Convert String to Floating Point Accumulator

TRANSFER VECTOR LOCATION: 637

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	CSFA	CSFA*	S
DAC*	S		

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: To convert a number represented by ASCII characters in the string to numeric format in the floating point accumulator. The character representation of the number is expected to be FORTRAN-like, with or without sign, decimal point, and power of ten denoted with "E". Upon exit, the A register contains the ASCII character which stopped the conversion process (i.e., the first character in the string which was not "legal" in the number, or octal 377 if end of the string was reached). The string pointers, S, are set to the next character after the stop character. It should be noted that a space is treated as a stop character.

LOCATIONS DIRECTLY ALTERED: U1, U2, U4, U6, FA, TR

SYSTEM ROUTINES CALLED: SET, GCI

LOCATIONS INDIRECTLY ALTERED: U0, U7

CFAS - Convert Floating Point Accumulator to String

TRANSFER VECTOR LOCATION: 636

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	format	LAA	format
SPB*	CFAS	CFAS*	S
DAC*	S		

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: To convert the number in the floating point accumulator to ASCII string representation according to the format word supplied in the A register:

```
Control          TD         W
|_|_|_|_|_|_|   |_|_|_|_|   |_|_|_|_|_|_|
0         5      6     9     10        15
```

Control: Bits 0 and 1: 0 = FORTRAN "F" format
1 = FORTRAN "I" format
2 = FORTRAN "E" format
3 = free format option (see below)

Bit 2: set = print trailing zeroes
not set = do not print trailing zeroes (substitute spaces) applies only to "F" format

Bit 3: not set = right justify in field (except for trailing blanks as above)
set = left justify in field

Bit 4: not set = use full field width
set = do not enter blanks into string

Bit 5: not currently used

TD: trailing digits after decimal. "F" and "E" formats only.
W: field width

The number is rounded if less than nine significant digits are printed. For "E" format, it is assumed that the exponent will require only two digits. If three or four are required, the extra exponent digit spaces are "stolen" from the trailing digits specified. Any errors in fitting a number to a specified format cause the "free format" option to be automatically used. This option has a field width of 16, with control bits 2, 3, and 4 of the format word being effective. If the absolute value of the number is less than 0.1 or greater than $10^9 - 1$, E16.7 is used. Otherwise, if the number is an integer, I16 is used; and if not, F16.q, where q = 8 - decimal exponent, is used. The floating point accumulator is destroyed.

LOCATIONS DIRECTLY ALTERED: U1-U9, U16-U21, FA, TR

SYSTEM ROUTINES CALLED: SET, FNEG, IATR, WCI

LOCATIONS INDIRECTLY ALTERED: U0

TTS - Time to String

TRANSFER VECTOR LOCATION: 662

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	TTS	TTS*	S
DAC*	S		

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The double word integer quantity in the second and third words of the floating point accumulator is taken to be a number of seconds and is converted to the string in the form HH:MM:SS. The double integer quantity may be obtained by using DFIX. Note that after calling TTS, simply subtracting 3 from the second string pointer will delete the seconds (:SS) entry.

LOCATIONS DIRECTLY ALTERED: TR, U3-U5

SYSTEM ROUTINES CALLED: SET, WCI

LOCATIONS INDIRECTLY ALTERED: UO-U2, U6-U8

STTI - String to Time

TRANSFER VECTOR LOCATION: 663

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	STTI	STTI*	S
DAC*	S		

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: The string is assumed to contain a quantity of the form, HH:MM:SS, and is decoded to a double integer quantity of seconds placed in the second and third words of the floating point accumulator. Trailing zero quantities (and accompanying colons) may be omitted, but leading colons must be present, i.e., 8:30 means 8 hours and 30 minutes, while :8:30 means 8 minutes and 30 seconds. The routine exits with the stop character in the A register, and the string pointers set to the remainder of the string after the stop character. Octal 377 denotes end of string reached.

LOCATIONS DIRECTLY ALTERED: FA, U1-U5

SYSTEM ROUTINES CALLED: SET, GCI

LOCATIONS INDIRECTLY ALTERED: UO, U6, U7

SYSTEM CONTROL ROUTINES

EXIT - Program Dismissal Request

TRANSFER VECTOR LOCATION: 657

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<code>	LAA	<code>
SPB*	EXIT	EXIT	
DATA	<control word>	DATA	<control word>

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: To dismiss a program under the condition given in the code:
Code = 0. Dismiss on scheduler check. The program is dismissed, allowing any other active programs to be scheduled. This permits a program to perform its own condition testing in an efficient manner. A test may be made to see whether some event has occurred, and if the test is not successful, this code may be used to dismiss the program rather than wasting its entire time quanta in a series of repetitions of the same test. After all other active programs have been scheduled (typically about 1/10 second), the program will be reactivated, allowing the next test.
Code = 1. Dismiss on wait for clock. The program will be restarted at the location following the EXIT call sequence when the system clock value is greater than or equal to the number of seconds given as a double word integer in the two locations following the subroutine jump to EXIT.
Code = 2. Dismiss on wait for software flag set. The data word following the call to EXIT will be used as a mask to select bits from the system FLAG word (location octal 570), and the program will be restarted whenever any of the selected bits are set.
Code = 3. Dismiss on wait for sense switch set. Similar to code = 2, except that the console sense switches are tested rather than FLAG.
Code = 4. Dismiss on wait for digital input set. Similar to code = 2, except that the first digital input unit is tested rather than FLAG.
Code = 5. Program halt. The program is dismissed and can be reactivated only through intervention of another program.
Code = 6. Panic - A panic is simulated, just as though two successive "ALTMODE" characters had been typed, and a Program Halt (code = 5) is performed.

LOCATIONS DIRECTLY ALTERED: U0-U2, U5, U6, program status words

SYSTEM ROUTINES CALLED: None (scheduler interrupt is activated)

EXIA - Program Dismissal Request, Address Mode

TRANSFER VECTOR LOCATION: 656

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
LAA	<code>	LAA	<code>
SPB*	EXIA	EXIA*	<address of control word>
DAC*	<address of control word>		

REGISTER TREATMENT: A and B registers destroyed

FUNCTION: EXIA is the same as EXIT except that the address of the control word or words is given rather than the control words themselves.

SCT - Set Clock Trap

TRANSFER VECTOR LOCATION: 676

CALL SEQUENCE:

SEL ASSEMBLER		SYSTEM ASSEMBLER	
SPB*	LFA	LFA	<time>
DAC	<time>	LAA	<prog no.>
LAA	<program no.>	SCT	
SPB*	SCT		

REGISTER TREATMENT: A saved, B destroyed

FUNCTION: This routine sets up a program, designated by the program number in the A register, to be activated at an exact clock time. The time is in the floating point accumulator, the mantissa being taken as the standard time in seconds in double integer format, and the exponent containing the negative 1/60 second clock tick count, between minus 60 and minus 1, inclusive. When this time is exactly matched by the system clock, the specified program number will be placed in location octal 677, causing immediate activation of the program (within a millisecond). This same action of placing the program number of the program to be activated into location 677 must be used in any programmer defined interrupt handler for which immediate response by a program is desired.

The seconds counter for system time is found in locations octal 576-577, and the address of the 1/60 second tick value is found in location 575.

LOCATIONS DIRECTLY ALTERED: U10, U11

SYSTEM ROUTINES CALLED: SFA

LOCATIONS INDIRECTLY ALTERED: U0, U1

CLRP - Clear Program

TRANSFER VECTOR LOCATION: 551

CALL SEQUENCE:

LAA	<program number>
SPB*	CLRP
BRU	<no such program>

REGISTER TREATMENT: A and B destroyed

FUNCTION: Exits at mark if the program does not exist; mark + 1 otherwise. The program number is the standard (negative) index of the program in the scheduler table, as found in PNUM in the Program Definition Area. Any input-output units owned by the program are cleared and released, and the program is halted. Upon exit, the address of the table is in the B register and A is cleared. Thus, the sequence:

```
LAA     4,1
STA     7,1
CLA
STA     0,1
```

starts the program at its starting location.

LOCATIONS DIRECTLY ALTERED: U2-U5

SYSTEM ROUTINES CALLED: CLRU

LOCATIONS INDIRECTLY ALTERED: U6, U7, U21, unit parameters

RESC - Scheduler Interrupt

TRANSFER VECTOR LOCATION: 557

CALL SEQUENCE: SPB* RESC
DAC <address of word containing return address>
(See section on Interrupt Routine Conventions)

FUNCTION: The scheduler interrupt allocates time to the various programs on the system and handles the stopping and starting of programs according to the availability of input-output devices. RESC is the lowest priority interrupt (software imposed) and may be entered either from other interrupts or from regular programs. It is not re-entrant, and precautions are taken to ensure that no attempt at re-entry be made. RESC may be locked out by a program by simply placing a positive non-zero number in location octal 571, and it may be re-enabled by re-zeroing this location, by calling EXIT, or by calling any system routine which performs an input-output wait (such as TCI or TCØ).

THE EXECUTIVE SYSTEM

The executive is a primary program which provides a means to set up, test, and alter a system of programs in real time. The system requests instructions by printing a dash on the Teletype and waiting. In general, instruction words are recognized from the first letter (or character) and the remainder of the word may be typed or not, as desired. Spaces are significant, and instruction lines should be terminated with a carriage return (CR). Control A (delete a character) and control Q (delete the line) may be used at any time, and panics (repeated ALTMODES) will cause the executive to return to the instruction request point. The executive uses locations octal 10200-11777, and the associated desk calculator uses 12000-12777.

Display Instructions

There are four basic display instructions:

-/ Display as unsigned octal integer
-# Display as signed decimal integer
-" Display as ASCII character pairs
-\ Display as floating point

These instructions may be used to dump the contents of successive memory locations in the format specified by following the display instruction with a range of octal addresses:

-/13000-13100 CR

The addresses must be in octal, no matter which display format is specified. The above instruction would print the contents of location 13000 through 13100 as unsigned octal integers, eight values per line. The address of the first value in the line is printed at the left of the line. The same procedure is used to list in the other formats, except that only three floating point numbers are printed per line. The dump may be terminated before the range of addresses is completed by typing two ALTMODES. In ASCII dumps, control characters are denoted by placing a square bracket next to the corresponding printing character; for example, the character pair control H, Q would be printed as [HQ and H, Control Q would be printed as HQ]. A word containing zero is printed as [@@].

If a display instruction is followed by a single octal address, it is interpreted differently, simply printing the contents of the specified location and waiting:

-/13000 CR
017244

At this point, a new value may be entered into the location simply by typing it in the specified format and following it with a carriage return. If a carriage return is typed without anything preceding it, the contents of the location will remain unaltered. Control characters may be entered in the ASCII mode by preceding the corresponding printing character with a control V.

If a control D is used in place of a carriage return to terminate an entry (or nonentry), it will cause the next address and its contents to be printed, pausing again for an entry. This process may be continued to serve as a means of entering blocks of information. A carriage return must eventually be used to return to the instruction request point.

Program Management

Up to 19 programs may exist on the system in addition to the primary program, and the executive numbers these from 0 (the executive) to 19. These program numbers are related to the negative system program numbers as:

Executive Program Number = - System Program Number - 1.

To enter a program into the scheduler list:

```
-ESTABLISH 13000 CR
PROG 19
```

The octal address specified is that of the first word in the program definition area for the program being put on the list. The executive puts the program in the halt state, assigns it a number (starting at 19 and decreasing), and prints the above message containing the number. The program will remain in the scheduler list until it is removed:

```
-REMOVE 19 CR
```

The remove instruction clears the program before removing it from the list.

Three instructions are available for starting and stopping programs. To start a program that has been established:

```
-START 19 CR          or
-START 19,700 CR
```

Both forms clear the program and start it at the location specified by SLØC in the program's Program Definition Area. The second form is logically equivalent to:

```
-START 19 CR
-WAIT 700 CR
```

the WAIT command implied by the presence of the comma. This form dismisses the executive at the same time that the specified program is started, and is used primarily when the program being started will request input from the Teletype, conflicting with the executive. The instruction:

```
-CONTINUE 19 CR          or
-CONTINUE 19,700 CR
```

is similar to START except that the program is not cleared first, and execution is begun where it last left off, rather than at SLØC. Finally, the program may be halted from the executive by:

```
-HALT 19 CR
```

which simply puts the program in the halt state.

Program Loading and Dumping

Two instructions are provided for loading and dumping absolute binary programs on paper tape:

-LØAD HPT CR

Causes the tape on the high-speed paper tape reader to be read. This tape should be in standard system format, and may be in several blocks, as outlined in the description of the "RBT" system function. If "TPT" is specified rather than "HPT", the tape will be read from the Teletype. To dump an absolute tape, the following format is used:

-DUMP 13000-14767,HPT,LDR,MØRE CR

An absolute dump of the contents of the range of octal addresses will be produced on the high-speed paper tape punch by the above instruction. "LDR" and "MØRE" are options, which may or may not be specified, as desired. "LDR" causes an initial length of leader to be punched before the dump is started, and "MØRE" indicates that more blocks are to be included in the same dump. This alters the first character of the dump, as described under the "PBT" system function, and if "MØRE" is specified, no trailer is punched. Thus, the following sequence:

-DUMP 13000-14767,HPT,LDR,MØRE CR
-DUMP 100-124,HPT CR

Might be used to prepare a tape of a program and its map zero references that could later be read using a single load instruction. "TPT" may be used instead of "HPT", as in the load instruction, but the executive waits for a control D to be typed on the Teletype both before and after each dump so that the paper tape punch may be turned on and off. Since the control D is not echoed back to the Teletype, this procedure allows a "clean" tape to be produced.

Device Inquiry and Testing

Several instructions are provided to deal with hardware devices. Voltage readings from the multiplexer may be displayed as follows:

-MUX CH,G,N CR

Where CH is the decimal channel number to be read, G is the gain range index, and N is the number of readings to be averaged (16 or less). If N is omitted, it is taken to be 1, and if G is also omitted, it is taken to be 4 (auto-ranging). Thus,

-MUX 25 CR

is the same as:

-MUX 25,4,1 CR

and means to read a single value from channel 25, auto-ranging, and display the floating point voltage resulting. Values of G, the gain index, are:

Gain Index	Full-Scale Voltage Range
0	5V
1	200MV
2	100MV
3	50MV
4	Auto-Ranging
5	20MV
6	10MV
7	5MV

To display the status of a digital input word, the instruction:

-INPUT N CR

is used, with "N" being either 0 or 1 to select one of the two digital input units. The current state of the unit is displayed as an unsigned octal integer. Digital outputs are tested with the instruction:

-OUTPUT N CR

But in this case, the executive will wait just as it does for a single display and allow the state of the unit to be altered. Alterations must be two words, the first being the data word indicating how the bits are to be set, and the second, a mask word, having bits set to select those positions which will be changed to the state in the data word, for example:

```
-OUTPUT 2 CR
070007 111111 133333 CR
-OUTPUT 2 CR
151115 CR
```

The value of N may range between 0 and 3 to select one of the digital output units. The system clock is handled through the TIME instruction:

```
-TIME CR
00:00:12 8:30:45 CR
-TIME CR
08:31:11 CR
```

The first example shows initializing the clock by typing in the correct time. This may be reset, if desired, at any time. The time may be checked as in the second example, which is eleven seconds past eight thirty-one.

System Calls

A system call is denoted by a semicolon and may take four forms:

```
-;D CR     Enter desk calculator, cleared
-;C CR     Enter desk calculator without clearing
-;K CR     Kill the desk calculator
-;S CR     Load primary program from high-speed paper tape
```

The first two operations are used to reference the desk calculator, as described in a separate section. The third is used to delete the desk calculator so that the space can be used by other programs. After ;K, further references to ;D or ;C will not be recognized until the calculator is reloaded. The ;S form actually may use any letter other than D, C, or K, and is used to initiate the primary program loading sequence. The executive will respond with the question "OK?", and after an affirmative response, will load a new primary program from high-speed paper tape and start it.

Other Executive Instructions

Comments may be typed in response to the executive's "-" by preceding them with an asterisk:

```
-* THIS IS A COMMENT CR
```

The system software FLAG, location octal 570, may be accessed with a special command:

```
-FLAG CR
```

which is exactly equivalent to:

```
-/570 CR
```

The executive may be put into a wait state by using the wait instruction:

```
-WAIT 700 CR
```

This will cause the executive to be dismissed on a wait for sense switch (code = 3). It can be reactivated by setting any of the sense switches which are selected by the octal word following the wait instruction word (switches 7, 8, or 9 in the above example), or by a panic (two or more ALTMODES). The octal word must be present, but it may be zero, in which case, only a panic will reactivate the executive. The wait command is also activated by special forms of the START and CONTINUE instructions.

The ZERO instruction is used to initialize the system and has three forms:

-ZERØ CR
-ZERØ N CR
-ZERØ @N CR
The first form is exactly equivalent to a power failure response, and all programs and peripherals are cleared and re-initialized in the same manner. The second form is used to clear programs, where N is the program number and is equivalent to the "CLRP" system instruction. The third form is used to clear peripheral units, given the unit number, exactly as in "CLRU", including the same options. With these three commands, it is usually possible to recover from a situation in which a program has become "hung".

THE DESK CALCULATOR SYSTEM

The desk calculator is usually present with the executive, although the two are separate programs. Unlike many "desk calculator" programs, this one merely simulates a common electronic desk calculator as nearly as can be done with Teletype input and does not provide algebraic input forms.

Calling the Calculator

-;D CR

Calls the desk calculator from the executive and executes a reset (R), clearing it. The calculator may be entered without clearing using:

-;C CR

and to return from the calculator to the executive,

;E

is used. The calculator does not print a "ready" character like the "-" of the executive, and in general, carriage returns are automatically generated rather than being typed by the user in keeping with the "one keystroke per function" concept of a desk calculator.

General Information

The calculator has five cascaded numeric entry registers, which will be called R1-R5 here. All entries go into R1, and when an entry is made, the registers are pushed up: R5←R4, R4←R3, R3←R2, R2←R1, and R1← the new entry. The former contents of R5 are lost. Entries may be made in FORTRAN-like numeric format, although negative entries or entries containing exponents must be enclosed in parentheses (in the manual mode).

14.56 CR
(-14.56) CR
(14.56E123) CR

are all legal entries. Control A (delete a character) and control Q (delete the line) are both permitted in entries. Entries are terminated with spaces, carriage returns, matching ')', or operation characters. Unary operations, such as taking logarithms or square roots, operate on R1, and the printing operation, =, also operates on R1. Thus, the square root of four is obtained by:

```
4S
= 2
```

Note that the result is not printed until requested. Binary operations take place between R1 and R2, the result replacing R1. A cascade down is also performed, so that R2←R3, R3←R4, R4←R5, and R5←0.

```
5 CR
3-
= 2
```

shows the procedure. Note the result of the following sequence, which shows a common beginning error in using the calculator:

```
3+
2= 2
```

In the above, 3 is entered into R1 and the registers are pushed up. The "+" then causes the addition of R1, containing 3, with R2, containing an unknown value. The 2 is then entered into R1 with the registers pushed up. The "=" simply says to print R1, and hence, the result 2 instead of the 5 that might have been casually expected.

<u>Basic Operations</u>

Unary arithmetic operations include exponential "E", natural logarithm "L", square root "S", inverse "←", and integer part "@". Binary operations include add "+", subtract "-", multiply "*", divide "/", and power "↑".

Cascade register operations are also provided, in three forms. ":" is repeat and push up, causing a second copy of the contents of R1 to be copied into R2 after pushing up the cascade (R5←R4, R4←R3, R3←R2). "C" is cascade and clear, and serves to eliminate the contents of R1 by R1←R2, R2←R3, R3←R4, R4←R5, and R5← zero. "I" serves simply to interchange R1 and R2. Examples:

Calculate 3.45/EXP(3.45)

```
3.45:
E
/
= 0.10952244
```

Calculate (3.6 + 1.5)/(1.7 + 4.8)

```
3.6 CR
1.5+
1.7 CR
4.8+
/
= 0.78461539
```

Calculate (1.3 + 5.6)/1.3

```
1.3:
5.6+
I
/
= 5.3076923
```

Constant Storage

In addition to the cascade registers, there are also ten constant storage registers, A-J. The instruction:

```
>A
```

stores the current value of R1 into the first constant storage register, not altering the value of R1, while:

```
#A
```

recalls the value of the first constant storage register into R1 and pushes up the cascade (R5←R4, R4←R3, R3←R2, R2←R1).

Programmed Routines

An elementary programming capability is available in the calculator. A string of operations may be "learned" and repeated as many times as desired. The learning mode is initiated by typing:

```
;L
```

A sequence of operations is then accepted until a semicolon is typed to end it. Spaces are used to denote where keyboard entries are desired in the sequence, and any unrecognized instructions will cause the printing of a question mark, but will otherwise be ignored. After entry, the sequence of operations may be started by typing:

```
;A
```

To start the "automatic" mode. The automatic mode will continue, restarting at the beginning of the string of instructions when it reaches the end, until a semicolon is typed when the calculator is requesting a numeric input. Alternatively, typing an ALTMODE will cause the sequence to stop after the next printing operation. Two ALTMODES in succession will cause a return to the executive (to handle a loop which has no prints or entry requests). If the learned sequence is restarted using ;A, it will start with the instruction immediately after the one on which it was stopped, meaning that programs which have been stopped on a request for an entry must have that entry entered manually before restarting. Parentheses are never required for numeric entries made in the automatic mode.

Examples:

Program (3.5+x)/(3.5-x)
and evaluate for x=1, 1.5, 2, 2.5

```
;L
 :#A+I#AI-/=;       (Note leading space for entry)
3.5>A
;A
1.0 1.8
1.5 2.5
2.0 3.6666667
2.5 6
;
```

Program ((1+x)/(1+Y))*EXP(1+x)*SQRT(1+Y)
and evaluate for (x,Y)= (.3,.3),(.3,.8),(.8,.3)

```
;L
#A +>B#A +>C/#BE*#CS*=;     (Note spaces for entry of x and y)
1>A
;A
.3 .3  4.1836419
.3 .8  3.5554119
.8 .3  9.5506052
;
```

Analysis of the above examples should make both the programming and calculator functions more clear.

<u>Additional Controls</u>

Since it is sometimes desirable to have copy printed across the rather wide Teletype page instead of down it, a "SPACE MODE" can be entered by typing:

;S

In this mode, all automatically generated carriage returns are converted to spaces with the exception of those generated at the end of a numeric print in the automatic mode. The user may insert his own carriage returns except in the middle of numeric entries. To revert to the "RETURN MODE",

;R

is used. As a final instruction,

R

is reset, and clears all the calculator registers to zero.

APPENDIX A

NUMERIC LISTING OF SYSTEM ROUTINES

Name	Code	Page	Function
CLRP	551*	55	Clear Program
NØRM	552*	38	Normalize FA
IATR	553*	38	Interchange FA and TR
LTR	554*	38	Load TR
SET	555*	38	Set Up Arguments
RESC	557*	56	Scheduler Interrupt
HIØF	560*	44	Turn Off HPT Input
TPT	561*	44	Set Reader Mode
TEL	562*	44	Set Key Mode
FSET	564*	37	Set Up Indexed Arguments
LFA	600	27	Load FA
SFA	601	27	Store FA
FAD	602	28	Floating Add To FA
FSB	603	28	Floating Subtract from FA
FSBI	604	29	Floating Subtract FA from
FMP	605	29	Floating Multiply
FDV	606	30	Floating Divide FA by
FDVI	607	30	Floating Divide by FA
FLØT	610	32	Float "A"
FIX	611	32	Fix FA Into "A" and FA
FNEG	612	33	Floating Negate
FABS	613	33	Floating Absolute Value
FPW	614	31	Floating FA to Argument Power
FPWI	615	31	Floating Argument to FA Power
EXP	616	33	Exponential
LN	617	34	Natural Logarithm
SQRT	620	34	Square Root
FLC	626	35	Floating Compare Three-Way
FLCZ	627	35	Floating Compare Zero
TCI	630	39	Teletype Character In
TCØ	631	40	Teletype Character Out
PCI	632	40	HPT Character In
PCØ	633	40	HPT Character Out
GCI	634	49	Get Character and Increment
WCI	635	49	Write Character and Increment
CFAS	636	51	Convert FA to String
CSFA	637	51	Convert String to FA
PAD	640	48	Pointers from Address

* Rarely used special purpose routine.

NUMERIC LISTING OF SYSTEM ROUTINES (Contd.)

Name	Code	Page	Function
LDR	641	43	Leader
MESG	642	50	Message to Unit
SØUT	643	49	String Out to Unit
SIU	644	50	String in From Unit
FDBL	647	36	Float Double Integer
DFIX	650	36	Fix Double Integer in FA
CMFV	651	46	Convert MUX Reading to FA
STAS	652	41	Set Status of Unit
DØIN	653	48	Digital Output - Indirect
DØDI	654	47	Digital Output - Direct
RDIN	655	47	Digital Input
EXIA	656	54	Exit - Address Mode
EXIT	657	53	Exit (System Call)
MXSV	660	46	Single MUX Value
MUX	661	44	Read Multiplexer Block
TTS	662	52	Time to String
STTI	663	53	String to Time
PBT	664	42	Punch Binary Tape
RBT	665	43	Read Binary Tape
UNIT	666	40	Convert Unit Number to Address
LDI	667	37	Load FA from Double Integer
SDI	670	37	Store FA as Double Integer
CLRU	671	41	Clear Unit
CIN	674*	51	Character Input
SCT	676	55	Set Clock Trap

* Rarely used special purpose routine.

APPENDIX B

ALPHABETIC LISTING OF SYSTEM ROUTINES

Name	Code	Page	Function
CFAS	636	51	Convert FA to String
CIN	674*	51	Character Input
CLRP	551*	55	Clear Program
CLRU	671	41	Clear Unit
CMFV	651	46	Convert MUX Reading to FA
CSFA	637	51	Convert String to FA
DFIX	650	36	Fix Double Integer in FA
DØDI	654	47	Digital Output - Direct
DØIN	653	48	Digital Output - Indirect
EXIA	656	54	EXIT - Address Mode
EXIT	657	53	EXIT (System Call)
EXP	616	33	Exponential
FABS	613	33	Floating Absolute Value
FAD	602	28	Floating Add to FA
FDBL	647	36	Float Double Integer
FDV	606	30	Floating Divide FA by
FDVI	607	30	Floating Divide by FA
FIX	611	32	Fix FA into "A" and FA
FLC	626	35	Floating Compare Three Way
FLCZ	627	35	Floating Compare Zero
FLØT	610	32	Float "A"
FMP	605	29	Floating Multiply
FNEG	612	33	Floating Negate
FPW	614	31	Floating FA to Argument Power
FPWI	615	31	Floating Argument to FA Power
FSB	603	28	Floating Subtract from FA
FSBI	604	29	Floating Subtract FA From
FSET	564*	37	Set Up Indexed Arguments
GCI	634	49	Get Character and Increment
HIØF	560*	44	Turn Off HPT Input
IATR	553*	38	Interchange FA and TR
LDI	667	37	Load FA from Double Integer
LDR	641	43	Leader
LFA	600	27	Load FA
LN	617	34	Natural Logarithm
LTR	554*	38	Load TR
MESG	642	50	Message to Unit
MUX	661	44	Read Multiplexer Block
MXSV	660	46	Single MUX Value

* Rarely used special purpose routine.

ALPHABETIC LISTING OF SYSTEM ROUTINES (Contd.)

Name	Code	Page	Function
NØRM	552	38	Normalize FA
PAD	640	48	Pointers from Address
PBT	664	42	Punch Binary Tape
PCI	632	40	HPT Character in
PCØ	633	40	HPT Character out
RBT	665	43	Read Binary Tape
RDIN	655	47	Digital Input
RESC	557*	56	Scheduler Interrupt
SCT	676	55	Set Clock Trap
SDI	670	37	Store FA as Double Integer
SET	555*	38	Set up Arguments
SFA	601	27	Store FA
SIU	644	50	String in From Unit
SØUT	643	49	String out to Unit
SQRT	620	34	Square Root
STAS	652	41	Set Status of Unit
STTI	663	53	String to Time
TCI	630	39	Teletype Character in
TCØ	631	40	Teletype Character out
TEL	562*	44	Set Key Mode
TPT	561*	44	Set Reader Mode
TTS	662	52	Time to String
UNIT	666	40	Convert Unit Number to Address
WCI	635	49	Write Character and Increment

* Rarely used special purpose routine.

APPENDIX C

FLOATING POINT ARITHMETIC ROUTINES

Name	Code	Page	Function
CFAS	636	51	Convert FA to String
CMFV	651	46	Convert MUX Reading to FA
CSFA	637	51	Convert String to FA
DFIX	650	36	Fix Double Integer in FA
EXP	616	33	Exponential
FABS	613	33	Floating Absolute Value
FAD	602	28	Floating Add to FA
FDBL	647	36	Float Double Integer
FDV	606	30	Floating Divide FA by
FDVI	607	30	Floating Divide by FA
FIX	611	32	Fix FA into "A" and FA
FLC	626	35	Floating Compare Three Way
FLCZ	627	35	Floating Compare Zero
FLØT	610	32	Float "A"
FMP	605	39	Floating Multiply
FNEG	612	33	Floating Negate
FPW	614	31	Floating FA to Argument Power
FPWI	615	31	Floating Argument to FA Power
FSB	603	28	Floating Subtract from FA
FSBI	604	29	Floating Subtract FA from
FSET	564*	37	Set up Indexed Arguments
IATR	553*	38	Interchange FA and TR
LDI	667	37	Load FA from Double Integer
LFA	600	27	Load FA
LN	617	34	Natural Logarithm
LTR	554*	38	Load TR
NØRM	552*	38	Normalize FA
SDI	670	37	Store FA as Double Integer
SET	555*	38	Set up Arguments
SFA	601	27	Store FA
SQRT	620	34	Square Root

* Rarely used special purpose routine.

APPENDIX D

SYSTEM CONTROL ROUTINES

Name	Code	Page	Function
CLRP	551*	55	Clear Program
CLRU	671	41	Clear Unit
EXIA	656	54	Exit - Address Mode
EXIT	657	53	Exit (System Call)
HIØF	560*	44	Turn Off HPT Input
RESC	557*	56	Scheduler Interrupt
SCT	676	55	Set Clock Trap
STAS	652	41	Set Status of Unit
TEL	562*	44	Set Key Mode
TPT	561*	44	Set Reader Mode

* Rarely used special purpose routine.

APPENDIX E

INPUT-OUTPUT ROUTINES

Name	Code	Page	Function
CIN	674*	51	Character Input
CLRU	671	41	Clear Unit
DØDI	654	47	Digital Output - Direct
DØIN	653	48	Digital Output - Indirect
HIØF	560*	44	Turn Off HPT Input
LDR	641	43	Leader
MESG	642	50	Message to Unit
MUX	661	44	Read Multiplexer Block
MXSV	660	46	Single MUX Value
PBT	664	42	Punch Binary Tape
PCI	632	40	HPT Character In
PCØ	633	40	HPT Character Out
RBT	665	43	Read Binary Tape
RDIN	655	47	Digital Input
SIU	644	50	String in From Unit
SØUT	643	49	String Out to Unit
STAS	652	41	Set Status of Unit
TCI	630	39	Teletype Character In
TCØ	631	40	Teletype Character Out
TEL	562*	44	Set Key Mode
TPT	561*	44	Set Reader Mode
UNIT	666	40	Convert Unit Number to Address

* Rarely used special purpose routine.

APPENDIX F

STRING AND CHARACTER PROCESSING ROUTINES

Name	Code	Page	Function
CFAS	636	51	Convert FA to String
CIN	674*	51	Character Input
CSFA	637	51	Convert String to FA
GCI	634	49	Get Character and Increment
MESG	642	50	Message to Unit
PAD	640	48	Pointers from Address
PCI	632	40	HPT Character In
PCØ	633	40	HPT Character Out
SIU	644	50	String in From Unit
SØUT	643	49	String Out to Unit
STTI	663	53	String to Time
TCI	630	39	Teletype Character In
TCØ	631	40	Teletype Character Out
TTS	662	52	Time to String
WCI	635	49	Write Character and Increment

* Rarely used special purpose routine.

APPENDIX G

EXECUTIVE INSTRUCTION SUMMARY

-/	Display in unsigned octal
-#	Display in signed decimal
-"	Display in ASCII
-\	Display in floating point

Address range for dump, single address for allowing alterations. Use control D to terminate alteration if continuation is desired.

-LOAD HPT (OR TPT)	Load Absolute Tape
-DUMP HPT,LDR,MØRE	Dump Absolute Block
-ESTABLISH 14700	PDA to Scheduler List
PRØG 19	
-REMOVE 19	
-START 19	
-START 19,700	START 19 and WAIT 700
-HALT 19	
-CONTINUE 19	
-CONTINUE 19,700	CONTINUE 19 and WAIT 700
-MUX CH,G,N	Channel,Gain,Number-to-Average
-INPUT 0	Digital Input
-OUTPUT 1	Digital Output
003017 60 70	(Current Value-Data-Mask)
-TIME	
00:00:53 9:15:00	
-FLAG	-/570
-WAIT 700	
-* THIS IS A COMMENT	
-ZERO	Reinitialize System
-ZERO 19	Clear Program 19
-ZERO @3	Wait and Clear HPT Punch
-;D	Clear and Enter Desk Calculator
-;C	Continue Desk Calculator
-;K	Delete Desk Calculator
-;S	Read in New Primary Program from HPT

APPENDIX H

DESK CALCULATOR SUMMARY

-;D (CR)	Call Calculator from Executive
-;C (CR)	Continue (Do Not Clear) Into Calculator from Executive
-;D (CR)	"Kill" Desk Calculator

5 Cascaded Entry Registers (R1-R5)
10 Storage Registers (A-J)
80 Programming Steps

NUMBER ENTRY

Entries go into R1.
Numbers Containing a Sign (-12.3) Or an Exponent Form (3.6E5) Must Be Entered in Parentheses in Manual Mode. Parentheses Not Necessary for Most Entries or in Automatic Mode. Entries are Terminated by a Space, or Carriage Return, or ")".

Control Q Deletes Entire Entry.
Control A Deletes Last Character.

OPERATIONS

Operations are on R1 or between R1 and R2 as described.

+ Add	R1←R2+R1
- Subtract	R1←R2-R1
* Multiply	R1←R2*R1
/ Divide	R1←R2/R1
↑ Power	R1←R2↑R1
E Exponential	R1←EXP(R1)
L Logarithm	R1←LN(R1)
S Square Root	R1←SQRT(R1)
← Inverse	R1←1/R1
@ Integer	R1←INTEGER(R1)
= Print	R1 is printed
: Repeat and Push-Up	R1 saved; R2←R1 R3←R2; R4←R3 R5←R4
I Interchange	R1<->R2
C Cascade and Clear	R1←R2; R2←R3 R3←R4; R4←R5 R5←0

>A Store	A←R1; "A" may be A-J
#A Recall	R1←A; "A" may be A-J, cascade up

CONTROL

R Reset	Clear Everything

;S Space Mode
;R Return Mode
;L Learn Mode - Enter Operations in order to form repeated sequence. Space denotes place for numeric entry, and ; denotes end of learn sequence.
;A Automatic mode - Perform learned sequence. ";" during numeric entry or ALTMODE typed during printout will return to manual. Restarting will start at next operation after the one that was stopped on.
;E Return to Executive program.

ALTMODE - Typed twice in a row will cause return to the Executive at any time.