NASA Technical Paper 1424

INFORM - An Interactive Data Collection and Display Program With Debugging Capability

David S. Cwynar

JANUARY 1980

NASA

NASA Technical Paper 1424

INFORM - An Interactive Data Collection and Display Program With Debugging Capability

David S. Cwynar
Lewis Research Center
Cleveland, Ohio

NASA
National Aeronautics
and Space Administration

Scientific and Technical
Information Office

1980

CONTENTS

SUMMARY

INFORM was developed to aid assembly-language programmers of mini- and micro-computers in solving the man-machine communications problems that exist when scaled integers are involved. In addition to producing displays of quasi-steady-state values, INFORM provides an interactive mode for debugging programs, making program patches, and modifying the displays. Auxiliary routines SAMPLE and DATAO add dynamic data acquisition and high-speed dynamic display capability to the program.

This report contains detailed programming information and flow charts to aid in implementing INFORM on various machines. It also serves as a user's guide for the program. (An assembly language listing is available through COSMIC.) The text is designed to be tutorial to novice programmers, yet not cumbersome to experienced persons. Detailed descriptions of all supportive software are provided.

Throughout the report, consideration is given to possible program modifications to satisfy the individual user's needs and to identifying likely modifications and explaining how they can be accomplished. The program is organized in a modularized, straightforward manner to simplify modifications and additions.

INTRODUCTION

The advent of the microprocessor and the reduction of the cost of computing machines has led to the use of digital computers for control of everything from television games to sophisticated industrial processes. Ruggedized digital hardware is now entering the heretofore mechanical domain of automobile and aircraft controls and other applications in hostile environments.

To be cost effective and to improve reliability, the number of parts in digital systems is being cut to a minimum. Such reductions, along with requirements for increased computing speed, demand assembly or machine language programming techniques that use scaled-integer arithmetic calculations. Unfortunately for the programmers of such systems, few aids exist that allow the programmer to think in terms of real-world values as is afforded by the use of higher level languages such as Fortran, Basic, and PL/M. This makes locating errors within assembly-language programs extremely difficult, increases the time required to get such programs up and running, and significantly increases the cost of software development.

The INFORM program is designed to overcome such obstacles. It might be thought of as an assembly-language programmer's dynamic debug, development, and data collection tool. INFORM was developed to meet the needs of engineers who were developing real-time digital controls under time and hardware constraints that made the use of integer arithmetic and scaled parameters necessary. The program was designed to meet the steady-state data display requirements of such controls. An interactive mode provides for dynamic display programming as well as some debugging and modification capability. The SAMPLE and DATAO subroutine additions were designed to meet dynamic data collection and real-time data display requirements of transient data.

Although originally developed on a 16-bit SEL 810B computer, the program was designed for diverse applications. It uses only 5 k words of the SEL core and is organized in a modularized, straightforward manner, permitting easy addition to, or modification of the program to meet the recurring or one-shot needs of any specific programmer. The SAMPLE and DATAO programs are examples of additions developed for the recurring needs of aerospace control systems being designed and tested at the Lewis Research Center (ref. 1). The use of these routines within the first year of operation saved sufficient time to justify the development costs. Aimed at general purpose applications, INFORM, SAMPLE, and DATAO have been used for nearly every project, large or small, performed by the SEL 810B propulsion control computer at the Lewis facility and are now being expanded to speed operation of Lewis's hybrid computing facility.

The INFORM program makes extensive use of an efficient input/output library called CIPHER. The function of each subroutine contained within this library, as well as information regarding other required supportive software, is given in detail in appendixes A to I.

The report first describes the overall capabilities of the INFORM, DATAO, and SAMPLE subroutines and how they are intended to operate within a user's system. The report then describes their operation. Details on the use of the interactive command structure for accomplishing the possible tasks follows. Examples are given to illustrate use of the software within a system. The report supplies all the information required by programmers to translate the program to their machines. Execution times and core requirements are given for an SEL 810B computer. Complete explanations and flow charts are given for all facets of the program. Reasons for using particular techniques are given to aid in the adaptation of the program to varing needs and to serve a tutorial function for novice assembly-language programmers. In addition, many possible modifications have been identified and explained.

Functional assembly-language listings for SEL 810B computers are available upon request through COSMIC.

PROGRAM DESCRIPTION

To understand how INFORM, SAMPLE, and DATAO operate, we will first discuss how these subroutines are incorporated into a functional system. Being subroutines, these programs do not set up an interrupt environment, nor do they perform timing functions usually performed by an executive. Instead, they are designed to operate in a user-structured environment. Two examples of how they fit into a typical operational environment will be given. An overall description of what each subroutine does follows these examples. All numerical values within the user's program to be manipulated by the INFORM, SAMPLE, and DATAO software are assumed to be integers representing parameters that have been scaled to range between the negative and positive integer value limits of the machine. The program also handles unity scaled integer parameters.

For clarity, this report makes a distinction between operators, programmers, and users. The term "operator" refers to a user who is conversing with INFORM in an interactive environment. "Programmer" refers to the person or persons who have implemented the INFORM software or its calling main programs on the user's computer and are responsible for the object modules. The term "user" is used when no distinction need be made between an operator or programmer. An operator or programmer is always a user, but the reverse is not always true.

Interface to User's Environment

The INFORM package comprises four independently callable subroutines: INFORM, DATAO, SAMPLE, and CLRSMP. Specifics of the calling sequences and use of each subroutine are given in a separate section of the report. Although they may be called by any main program, they are primarily intended to be executed on the lowest or "spare-time" levels of a priority interrupt system, where the main function of the computer (e.g., a process control algorithm) is executed on higher levels. These higher levels are usually driven by recurring external interrupts such as those of a real-time clock or interval timer. When used in this fashion, the command structure made available to the operator by INFORM creates an interactive operating environment for any passive, real-time program. Further, the programmer creates this environment by simply defining the interrupt structure and executing a simple CALL INFORM statement on the lowest level.

If the system also uses the digital computer for data collection, a simple call to SAMPLE, where the storing of data would normally be programmed, gives the INFORM subroutine interactive control over the data-collection process. This allows the operator to store any parameter available within the machine at the time of call. The operator also gains the ability to restructure the available storage. For example 200 time points of 20 variables, 800 time points of 5 variables, or one time point of 4000 variables are all possible with 4 k words of storage. In addition, using the SAMPLE subroutine automatically provides a means for transferring the stored data to a bulk storage device by means of INFORM's interactive command structure.

In a similar manner, dynamic displays of system variables may be controlled by the INFORM subroutine if the programmer uses the DATAO subroutine for his displays. The programmer need only insure that all parameters to be displayed are in the core and that DATAO is called in the interrupt sequence every time he wishes to update the display. All programming is then complete to give the operator interactive control over the display. The operator may then select which parameters are displayed and expand or compress scales at will without concern for scaling or recalibration of the display recorders. As written, DATAO is intended for use with fixed-calibration analog displays using 0- to 10-volt inputs, but it may be modified to work with any display that can be driven to its scale extremities by system digital-to-analog converters (Dacs).

Figures 1 and 2 show the execution sequence of the INFORM, DATAO, SAMPLE, and CLRSMP subroutines for a typical system. The updating of the main program is controlled by the system clock, with data samples being saved by SAMPLE every sixth clock period. The DATAO display is updated every other clock period. If sufficient time is available between updates to execute the main program and update SAMPLE or DATAO, a simple one-level interrupt scheme (as shown in fig. 1) will suffice. In this system the system clock interrupts the INFORM background task and forces execution of the main program. After each completion of the main task, a decision whether SAMPLE or DATAO should be updated is made before returning to INFORM. The flow chart for such a system is shown in figure 2. Flow-chart symbols are defined in appendix G.

A multilevel interrupt sequence may be executed as shown in figures 3 and 4. This system no longer requires that sufficient time be available during each update to execute SAMPLE or DATAO. Instead, it may be treated as any other interrupt level with its execution being determined on a priority basis with requests for service being generated by devices such as system clocks or direct memory access controllers. In such an asychronous environment, conflicts might arise when attempting to change the DATAO display or to restructure the SAMPLE storage through use of INFORM commands while higher priority routines are using SAMPLE and DATAO. Such problems have been eliminated within the INFORM software, provided the INFORM interactive commands are always performed on a lower level of priority than DATAO, SAMPLE, or CLRSMP.

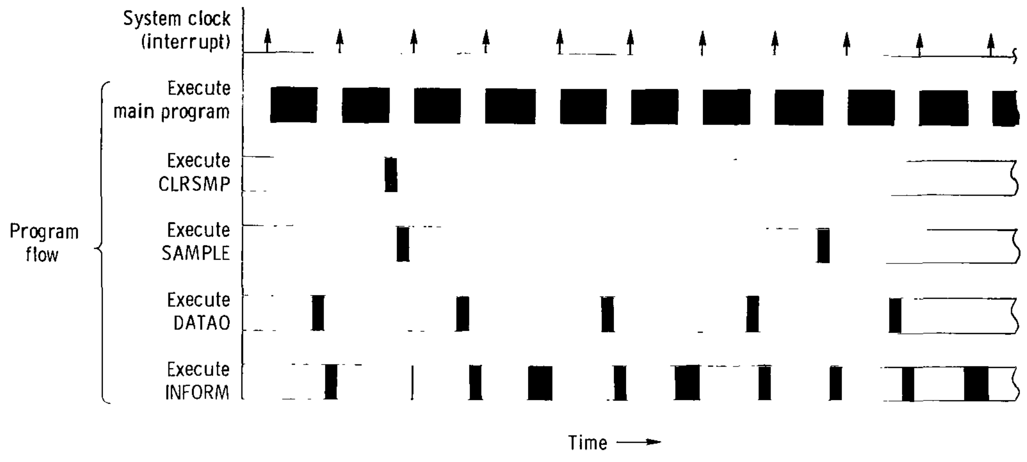


Figure 1. - Typical execution sequence for systems using one priority interrupt.

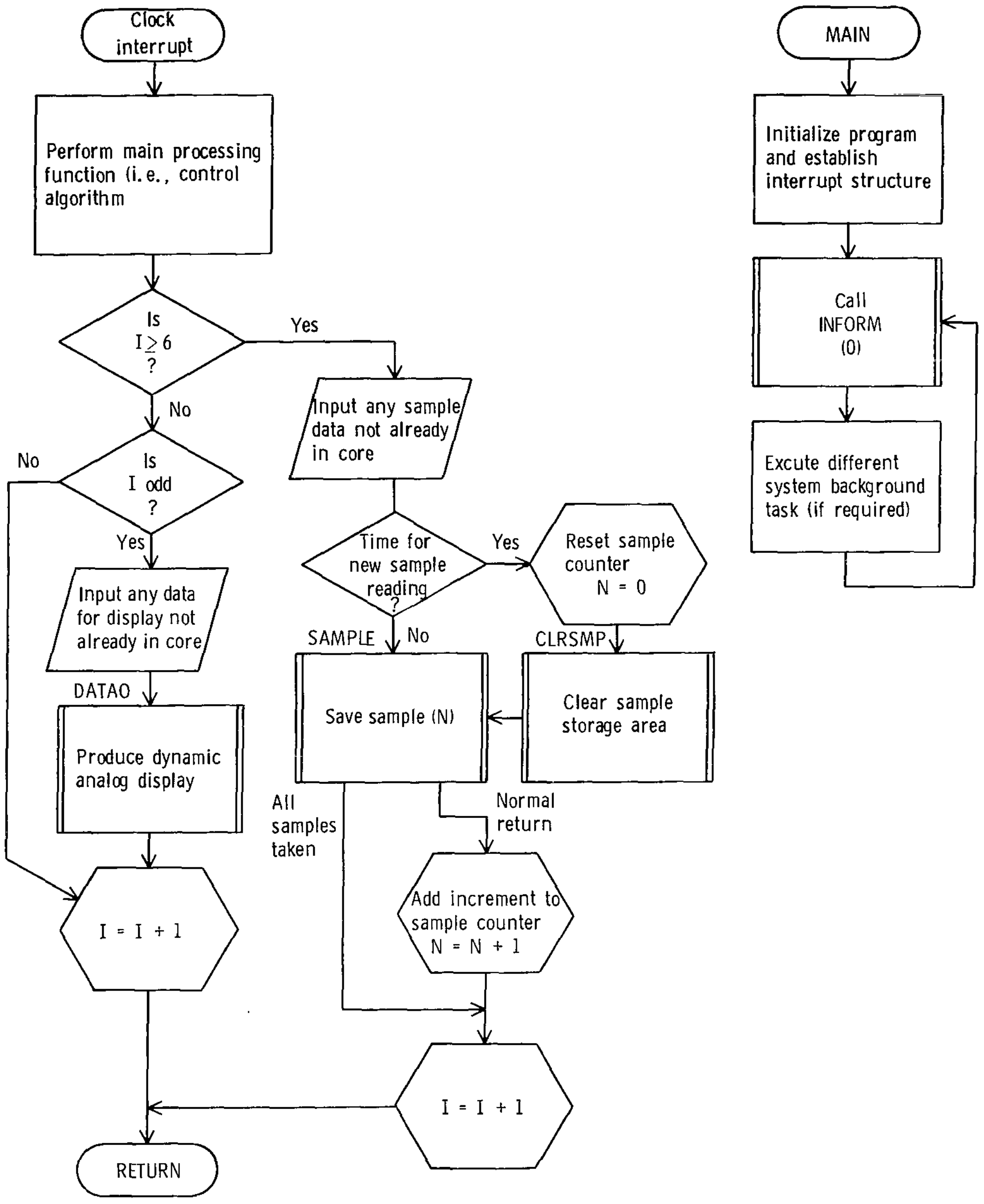
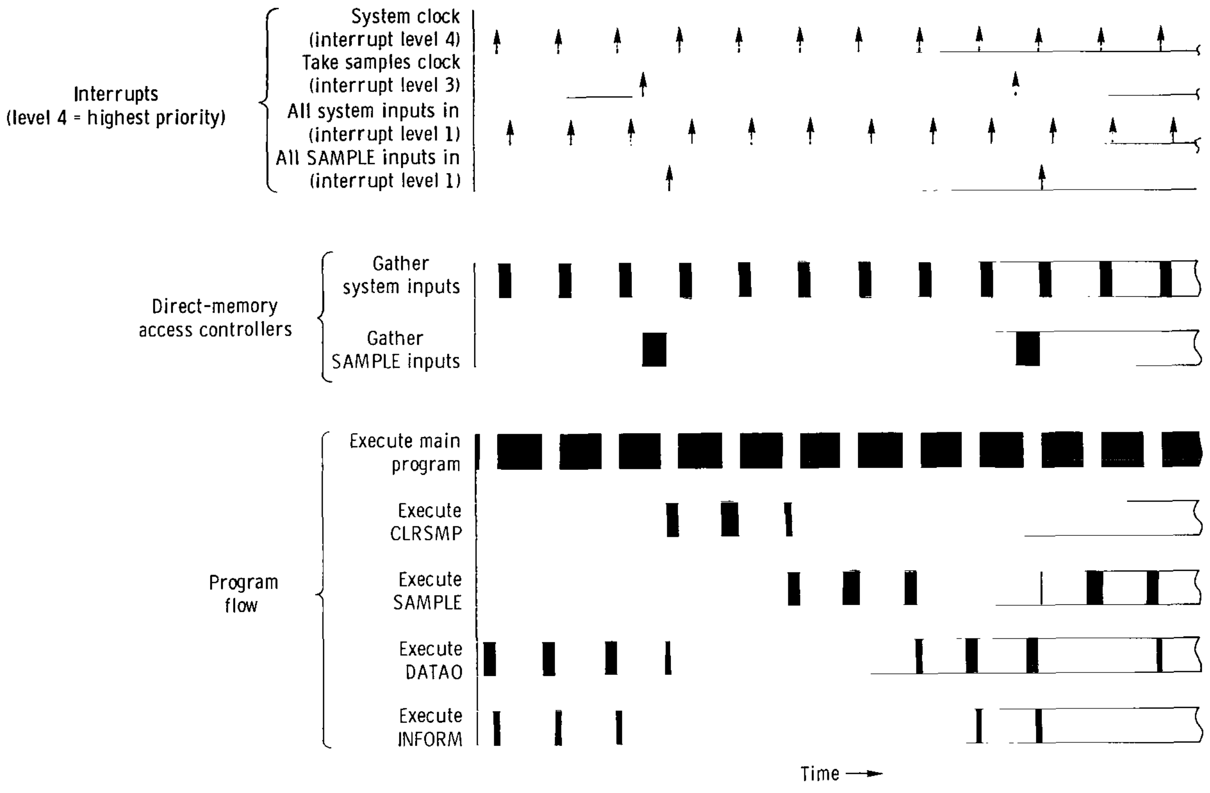

Clock interrupt
Perform main processing function (i.e., control algorithm
Is I ≥ 6 ?
Yes
No
Is I odd ?
No
Yes
Input any data for display not already in core
DATAO
Produce dynamic analog display
I = I + 1
Input any sample data not already in core
Time for new sample reading ?
Yes
Reset sample counter N = 0
SAMPLE
No
CLRSMP
Save sample (N)
Clear sample storage area
All samples taken
Normal return
Add increment to sample counter N = N + 1
I = I + 1
RETURN
MAIN
Initialize program and establish interrupt structure
Call INFORM (0)
Excute different system background task (if required)

Figure 2. - Typical flow chart for systems using one priority interrupt.



Interrupts (level 4 = highest priority)
System clock (interrupt level 4)
Take samples clock (interrupt level 3)
All system inputs in (interrupt level 1)
All SAMPLE inputs in (interrupt level 1)
Direct-memory access controllers
Gather system inputs
Gather SAMPLE inputs
Program flow
Execute main program
Execute CLRSMP
Execute SAMPLE
Execute DATAO
Execute INFORM
Time

Figure 3. - Typical execution sequence for multilevel interrupt system with data input through direct-memory access channels.

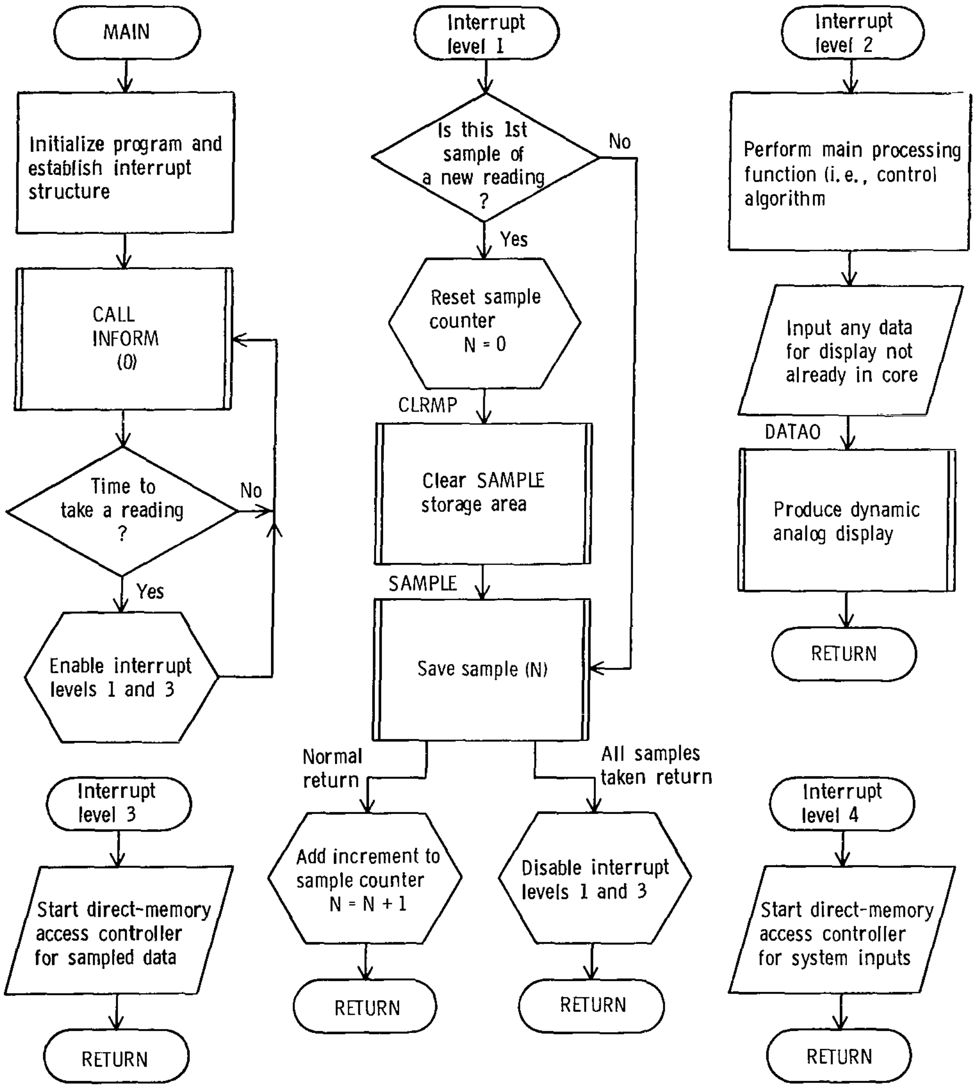

MAIN
Initialize program and establish interrupt structure
CALL INFORM (0)
Time to take a reading ?
No
Yes
Enable interrupt levels 1 and 3
Interrupt level 1
Is this 1st sample of a new reading ?
No
Yes
Reset sample counter N = 0
CLRMP
Clear SAMPLE storage area
SAMPLE
Save sample (N)
Normal return
All samples taken return
Add increment to sample counter N = N + 1
RETURN
Disable interrupt levels 1 and 3
RETURN
Interrupt level 2
Perform main processing function (i.e., control algorithm
Input any data for display not already in core
DATAO
Produce dynamic analog display
RETURN
Interrupt level 3
Start direct-memory access controller for sampled data
RETURN
Interrupt level 4
Start direct-memory access controller for system inputs
RETURN

Figure 4. - Typical flow chart for multi level interrupt system with data input through direct-memory access channels.

INFORM, DATAO, SAMPLE, and CLRSMP provide nearly everything the programmer requires to perform the complete data collection and display task. INFORM does, however, require use of some additional software in the form of standard Fortran arithmetic routines and a special input/output (I/O) library. The effect on the operating environment of the specifics of the particular Fortran library available to the programmer cannot be given here because it will vary from machine to machine. The I/O library used is the CIPHER library developed at Lewis. It is this library that is responsible for many of the convenience features available in INFORM. A complete list of all required subroutines and their functions is given in appendix F.

INFORM Subroutine

The INFORM subroutine provides an on-line display and data manipulation capability for single-precision, integer values retained in absolute memory. The operator programs the display while the program is running in the interactive mode. Memory locations are referred to by operator-assigned, one- to five-character, alphanumeric names. Values for these named locations may be displayed in engineering units (EU) or octal. INFORM determines these EU's by multiplying each integer value by a scale factor associated with the name. Once defined, the names and their associated locations and scale factors remain fixed unless redefined by the operator.

The display may be one of five types. Four are tabular, and the fifth is single-variable interactive. The display may be produced on any available I/O device. Up to three different data-table sequences, any one of which may be in any of the four tabular formats, may be saved for use by the display. To save laborious reprogramming, a load-or-dump option enables the operator to load previous definitions of named locations and data-table sequences, or to save the existing definitions via a relocatable binary-dump tape. This option is especially useful when INFORM is used for programs that have their named locations defined as being in common core. In this case programs may be updated and reloaded without having to redefine the INFORM names. As long as the common locations remain fixed, the dump tape may be used to "teach" a new load the old definitions.

Three modes of INFORM operation are possible, one interactive and two passive. In the interactive mode, the operator can issue a command and obtain an immediate response. In this mode the operator can define variable names, scale factors, and locations; display current values of named locations in octal, decimal, EU, or other programmed formats; display the results of numerical calculations in either octal or decimal; display, in EU's, ratios, or other algebraic combinations of named locations; and alter or set named locations to given or calculated values. The sequentially formed arithmetic expression (SFAE) as described in appendix B is the basis for numerical or algebraic calculations.

A default-name (no name) option allows temporary definitions for display or permanent alteration of any value in core. Alterations are made in a format that is convenient to the operator as well as in the machine's inherent binary code, thus enabling program patches for debugging purposes.

The two passive modes of operation are a data-table-printout mode and a variable-trip mode. Only one mode of operation may be in effect at a time. A change from the active mode to one of the passive modes or vice versa is accomplished by a sense switch, which is tested each time INFORM is entered and upon completion of a data-table printout. If the data-table printout is selected as the current passive mode, a data table is printed each time INFORM is entered. The data-table format and the I/O device for this display may be specified when the program is in the interactive mode. Also in this mode, the operator may select the third or variable-trip mode as the current passive mode. This mode will suppress a data-table printout until a specified named location becomes either greater than, less than, or equal to a defined trip point. When the trip condition is reached, an optional computer halt, before the table printout, is possible. More detailed information on how to select and enter the different modes of operation is given in the Program Operation section.

DATAO Subroutine

DATAO is used to display the values of INFORM named locations on analog devices driven by system Dacs. The routine is designed for high-rate operation so that it can provide acceptable displays when used for transient data. Speed of execution on the SEL 810B is 5 microseconds plus 50 microseconds for each Dac written. Normally, DATAO is programmed to drive fixed-calibration recorders whose scale extremities correspond to inputs of 0 and +10 volts. To protect such recorders, output voltages are limited to -0.1 and +10.24 volts by DATAO. The EU values that correspond to these levels, as well as the named location to be assigned to any given Dac, or recorder, may be defined or altered when in the INFORM interactive mode. Hence, the operator has the ability to expand, compress, shift, or reverse scales without having to adjust recorder gains or repeat recorder calibration.

A load-or-dump option is also available for the definitions of the DATAO display channels. If less than 24 DATAO channels are defined, the unused channels are skipped, and no execution time is required for those channels. The maximum channel limit of 24 is a programmer's option and may be changed.

SAMPLE Subroutine

SAMPLE is a transient data collection routine that samples and stores values of INFORM named locations. Designed for dynamic data collection, its execution time is only 15 microseconds plus 24 microseconds per sample taken on the SEL 810B computer. The named-location values to be saved and the memory locations to be used to store the values collected are defined or altered in the INFORM interactive mode. The operator must also define the size of the storage blocks (i.e., the number of samples to be saved for each variable) and an averaging parameter used for an optional data filtering mode. A relocatable binary-dump tape option similar to that provided for INFORM is available for these definitions.

In the simplest form of operation, one value for each specified variable is sampled and stored each time SAMPLE is called by the main program. The operator specifies the variables to be sampled by defining SAMPLE channels; that is, by specifying a named location to be sampled and the location of a storage block to be used to save the samples. The length, which is common to all storage blocks, must also be defined by the operator. Each time the main program calls SAMPLE, it also supplies a sample number that indicates the relative location in the storage blocks to be used for storage. Each time SAMPLE is called, SAMPLE stores one sample for each currently defined SAMPLE channel in this relative location. If a sample number received from the main program exceeds the defined storage-block lengths, no samples will be stored, and the return to the calling program will be different to indicate that all samples have been taken. The main program may then use this information to stop the sampling process or to recycle.

Another mode of operation of SAMPLE is possible: Instead of storing each sample in its relative location, it is added to a summation value already stored in the relative location. This mode is intended for the averaging type of data filtering. The summation values are automatically divided by the number of points added together (i.e., an averaging parameter supplied by the operator) when the sample values are output. The sampling mode is determined by the value of the averaging parameter defined by the operator. If the averaging mode is in effect, the main program, which calls SAMPLE, must also call CLRSMP to set the initial summation value to zero.

If averaging is being used, the storage-block length defined by the operator must be at least two times the number of samples to be taken. The extra locations are needed to store the summation as a double-precision value.

The contents of any or all of the storage blocks may be dumped in either of two fixed formats on any available I/O device. The output includes a coded number to indicate the first and last variables dumped, the scale factors, the number of samples per variable, and the data in integer or floating-point format. When integers are output, they represent the scaled values; while the floating-point data are output as EU values. The formats provide for convenient interpretation by the user and for interfacing with Fortran data-reduction programs. Integer valued checksums are provided for each line of output. For further details consult the APPLICATIONS AND EXAMPLES section and the $1 and $2 command descriptions in appendix C (pp. 82 and 84).

The operator may also scan data collected by SAMPLE by using an INFORM command designed to list core. Details of this procedure are given in the APPLICATIONS AND EXAMPLES section.

CLRSMP Subroutine

CLRSMP is provided to set the values in all the SAMPLE-channel storage blocks to zero. It must be called to initiate the start of a new summation for the averaging-filter mode of SAMPLE. CLRSMP may also be called if SAMPLE is not in the averaging mode. However, it should never be called by the main program until the operator has dumped the sampled values in the storage blocks, as these values are destroyed by CLRSMP.

PROGRAM STATISTICS

Core Requirements

Because different users will implement different portions of the program, the core requirements are broken down into convenient categories (table I, p. 14). Although the values shown are for an SEL 810B computer, they should be typical for any 16-bit, two-accumulator machine with index register.

Calling Sequences

The calling sequences given herein are for the SEL 810B computer. The 810B is a 16-bit minicomputer with a maximum addressing range of 32 k words. It is a two-accumulator machine with index register and uses two's complement arithmetic. Addressing is either direct or program-counter relative. Infinite indirect addressing with either post- or pre-indexing is available. Two-operand instructions always use an accumulator as one of the operands. Memory cycle time is 750 microseconds, with most memory referencing instructions requiring two memory cycles for execution. Use of indexing or indirecting requires one additional memory cycle for each indirect or indexing operation. Implied operand or single-register instructions require one memory cycle for execution. When both indirect and indexing appear in the same instruction, indexing is done before the indirect.

The call to INFORM must be followed by a coded data word to indicate the type of format desired for passive-mode data table printouts. The meaning of the bits of the coded word are defined in figure 5. The least significant three bits of the word are an output unit number. The next most significant three bits are a data table number, and the least significant three bits of the bits remaining signify the type of format. If any of these three-bit values is zero, default values DF0, DF1, and DF2, as defined by the operator, are used for the format type, data table number, and output unit number, respectively. On program load or re-initiation of the program due to a definition-dump read-in error, these default values are set to initial values.

When SAMPLE is called, the hardware A accumulator must contain the sample number. The first location following the call is the abort return location used when the sample number supplied lies outside the currently defined storage-block length. The second location following the call is the normal return location. Whenever it is desired to clear the SAMPLE-program-storage blocks, simply call subroutine CLRSMP.

There is no special calling sequence for DATAO. Simply call DATAO whenever it is desired to update the Dac display.

Timing

The times given below are for the SEL 810B computer. The execution time for SAMPLE in the nonaveraging mode is approximately 15 microseconds plus 24 microseconds per channel sampled. The averaging mode requires 16 microseconds plus 36 microseconds per channel. The timing for CLRSMP is the same as for the nonaveraging mode of SAMPLE. The time required to execute DATAO is approximately 5 microseconds plus 50 microseconds per channel displayed. INFORM requires 10.5 microseconds per named location to collect the data for a data table printout. A time lag of approximately 66 microseconds plus 18 microseconds per named location occurs between entry of subroutine INFORM and the start of data collection.

DESCRIPTION OF PROGRAM OPERATION

Certain definitions and conventions will be used when describing the operation of the program. For reference, these conventions are listed in appendix B.

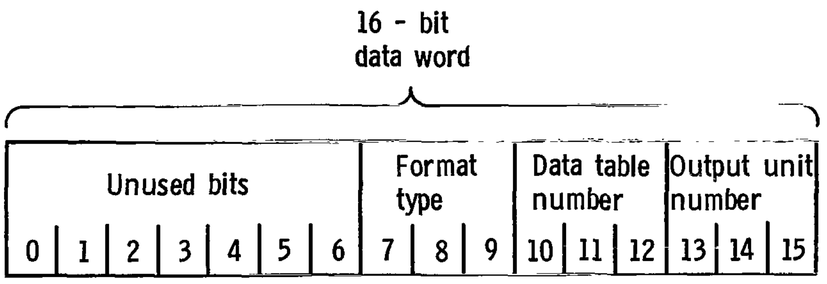


Figure 5. - Format for coded data word used when calling INFORM to establish form of passive-mode data-table printouts.

TABLE I. - CORE REQUIREMENTS FOR SEL 810 B SYSTEM

INFORM:		
Main program and subroutines	776	
Command processors (excluding >, <, /, [, and] commands)	286	
Named location definition tables (250 names)	1251	
Data tables	179	
STOR (temporary storage buffer)	174	
Default options and processors	384	
		3050
SAMPLE:		
Central command processors (< and > commands)	208	
Channel definition tables (80 channels)	165	
SAMPLE subroutine	43	
CLRSMP subroutine	13	
		429
Command Processor common to DATAO and SAMPLE		131
DATAO:		
Control command processor (> command)	200	
Channel definition tables (80 channels)	169	
DATAO subroutine	40	
		409
INPUT/OUTPUT subroutines:		
MESAGE	27	
OUT00, OUT10	45	
OUT11, OUT12, OUT22, OUT21, and OUT44	243	
PUNCH, LOAD, VERIFY	115	
TRYR, TTYR2	207	
FDDH, FDDEND, FDD11, FDD12, FDD22, FDD21, FDD44	105	
INPT, AIP, AOP, AIPX, AIPRX, and AOPX	93	
		835
Fortran arithmetic and conversion		519
Grand total		5373

INFORM Mode Selection

Mode selection is most easily understood by following the flow chart of figure 6. In general, the interactive mode may be entered at any time by setting the operator's mode-select switch (a sense switch). The passive mode may only be entered immediately following entry from the calling program with the operator's mode-select switch reset. Entry of the variable-trip mode is controlled by the variable-trip-mode latch (VTLH), which is set or reset by interactive mode commands. The variable-trip mode will not be entered until the INFORM subroutine is exited and re-entered with the operator's mode-select switch reset and VTLH set.

Use of Commands

Generally, INFORM is run in the interactive mode with the user obtaining information from the program by issuing a command and observing the response. A command consists of a key character preceded and/or followed by substantiating characters called operands. For example, if we desire the current value for a parameter called SPEED, we would type

```
SPEED=
```

Here, = is the key character and the name "SPEED" is an operand. If the operands follow the key character, they must be separated from the key character by a space or comma. A complete list of commands and an explanation of their operands and functions for INFORM, DATAO, and SAMPLE are given in appendixes C to E.

Commands are separated into three groups: control-edit commands provided by INFORM, commands used to control the DATAO subroutine, and commands used to control the SAMPLE subroutine. The = command of the above example is an INFORM control-edit command. The operator always knows when the program is ready to accept a command as it will print an underscore for control-edit commands, < for SAMPLE control commands, or > for DATAO control commands at the beginning of a new input line it creates by supplying a carriage return and line feed to the operator's input console. This character is followed by the ringing of the bell on the console. Assuming SPEED has a current value of 1500 rpm, the complete line for the preceding example would be

```
_SPEED=  15000-01
       ----------
```

The underscore is used throughout this report to indicate characters printed by the machine, thus separating them from operator entries.

The decimal value produced by the program is actually the EU value printed in the Fortran format of 5PE10.5 except that the decimal point, which occurs to the right of the five-digit mantissa, and the E are omitted. This format is used for all displays of decimal numbers, as it increases speed of printout on slow I/O devices and shortens line length for cathode-ray tube (CRT) devices. Little readability is sacrificed since the beginning of the exponent can always be identified by the space or minus sign following the mantissa. The increased speed and shortened lines are especially beneficial when lengthy data table printouts occur. Hence, the above value is 1,500.0 If the octal value were being displayed, it would be indicated by an apostrophy preceding a six-digit octal value. These formats are characteristic of the OUT22 and OUT00 subroutines of the CIPHER I/O library (ref. 2), and hence may be changed accordingly.

It is not always necessary to enter all operands listed for a command. Some operands may be defaulted (omitted). Those operands that may be defaulted have default values listed under their description in the appendices. If the operand is defaulted, the default value will be used. For example, if we wish to display the value of SPEED a second time, we need only type the "=" key character, since the default value for the operand is the name operand used by the previous command, that is, SPEED.

Thirteen of the default values of the various commands are modifiable by the user or operator. These defaults are identified as DF0 through DF12. The programmer can alter them by changing the initial values within the INFORM software. The operator can change them using the "%" change default value command. These default values are included on INFORM definition-dump tapes. Hence, the status of the defaults when the dump was produced will be restored whenever the dump is loaded.

Commands are processed character by character but are not executed until all entries have been received. This enables the user to maintain the status quo by cancelling the command before completion by entering "#" (ASCII '245). A successful cancellation will be indicated by "?" printed on the operator's console. If the operator types an erroneous or illegal operand, the program will cancel the entire command line and indicate same by typing "?" on the operator's console.

Most commands, unlike the "=" command, have additional operands following the key character and, therefore, require a terminating character to indicate the end of the operand entries. This terminating character is a carriage return. If a carriage return occurs before the entry of all operands, it is assumed that the remaining operands are defaulted. If no default value exists for one of these remaining operands, the "?" error message is issued.

Defining named locations. - If, in the preceding example, the location and scale factor (Sf) for SPEED had not been previously defined, the machine would have issued the error message "U ?" and cancelled the instruction. To be able to use SPEED as a name operand, it must have been previously defined using the "space" define-redefine variable name command. Using the "space" command on a variable name that has already been defined invokes the redefine name processor and causes the subsequent new definition to replace the existing definition.

To define a name using the "space" command, the name to be defined must precede the "space" entry. A name must consist of from 1 to 5 of the 36 possible alphanumeric characters, with the first character being one of the 26 alphabetic characters. No delimiter, such as a space or comma, is needed to separate the "space" commands from its operands, as the machine will type a double asterisk or the word "REDEF" following the "space" to alert the operator that operands for a permanent name definition are to follow. If the new name being defined is not unique, or is not the same as a name being redefined, an error message will be received, and the command will terminate. The error message "Q?" indicates that the name-learning capacity has been reached and no new names may be defined unless old names are redefined.

A typical line to define SPEED using the space command would look as follows:

```
_SPEED **'256,15000/32000
```

Note that the above line was terminated with a carriage return. To define a name, entry of the location and scale factor corresponding to the name used must follow the double asterisk or "REDEF" message (i.e., SPEED). Once these entries are made, the name is properly defined and will appear on any subsequent relocatable binary-dump tapes. If one wishes not to proceed with the definition, the line must be cancelled by entering "#" before the carriage return.

Of the two operands required to define a name, the location must be entered first. It may consist of any sequentially formed arithmetic expression as defined in appendix B. If a name occurs within the expression, the location, as currently defined for that name, is used as the numerical value. The only restriction is that the calculated value must be a legitimate core location. To facilitate the defining of names for successively increasing locations, the location entry may be defaulted by entering only a comma where the octal or decimal address would be. In this event the address is assumed to be one greater than the address defined for the last-used name operand of previous commands. When an address is defaulted, the calculated value is displayed to the operator before the acceptance of following characters.

The address thus entered must be followed by the scale factor (Sf), which may consist of any sequentially formed arithmetic expression defined in appendix B. If a name occurs within the expression, the scale factor as currently defined for that name is used as the numerical value. The resulting Sf becomes the number to be used to multiply by a machine-stored number to produce the EU value. The use of the divide option of the sequentially formed arithmetic expression facilitates the entry of fractions, such as, 100 percent per 2**(15) (the largest 16-bit word or 1S) or 15 000 rpm per 32 000, where 32 000 counts are produced by an analog-to-digital converter for a 10-volt input. These entries could then look like 100/1S and 15000/32000. If the value entered is zero, the scale factor is assumed to be one, and an octal instead of decimal display of the number is desired.

To change a name, or to replace an existing definition with a new one, we proceed as before except that the old name must precede the "space" and the new name must be entered as the first operand. For example,

```
_SPEED REDEF_REVS,'256,1000/32000
       ------
```

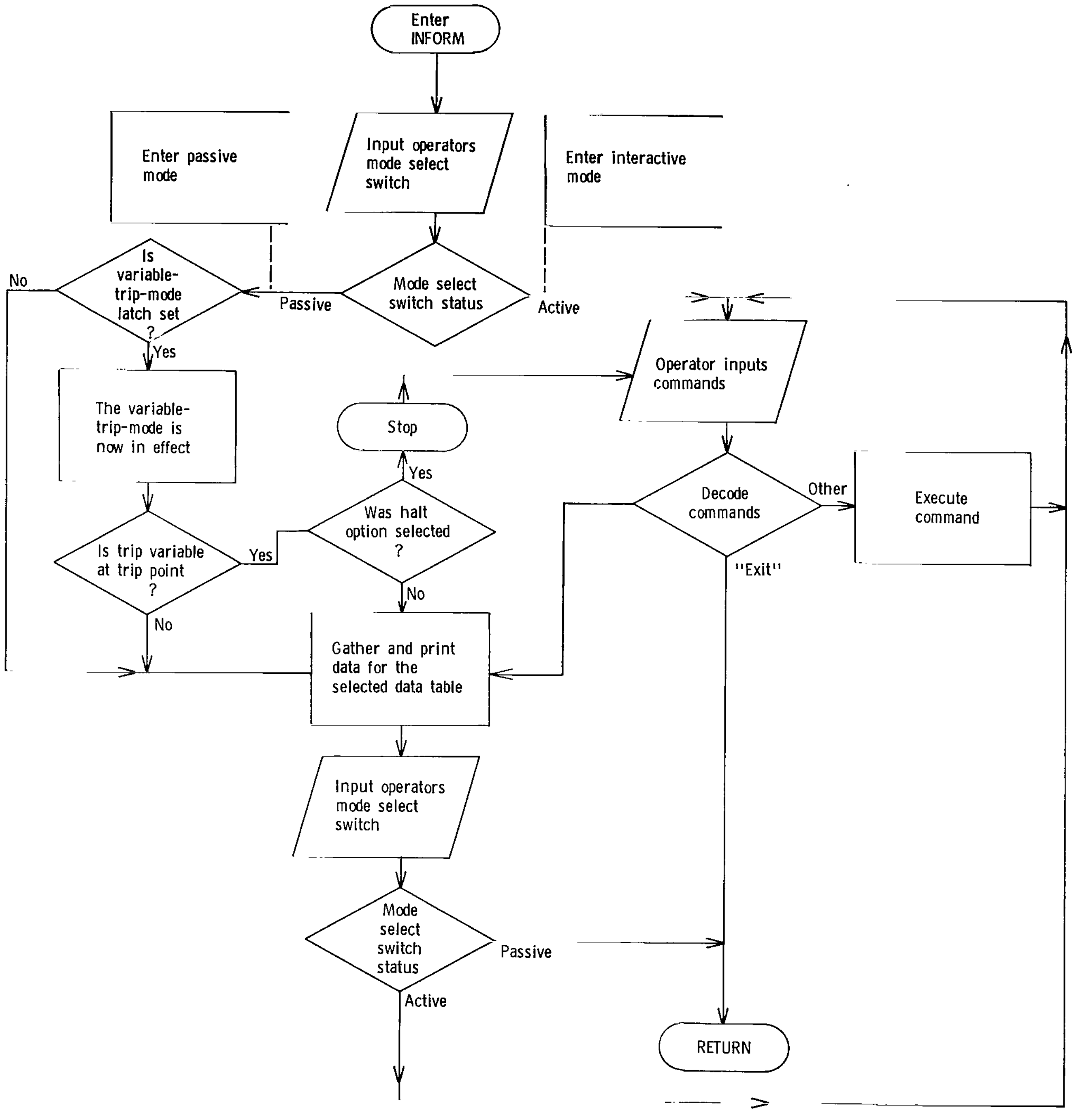
would redefine SPEED as the new name of REVS and change its scale factor to 1000/32000.

Notice that a core location may have any number of different names and scale factors assigned to it. However, each name must have a unique location and scale factor associated with it. Storage for a maximum of 250 different names is provided, but it is a simple matter to change this to any value between 1 and 254.

Data input and output control. - Data input and output are controlled through use of INFORM interactive commands (appendixes C to E) and by the user's manipulation of the INFORM operating mode. Displays produced by DATAO and samples taken by SAMPLE will, in general, also be under some form of control provided by the calling program. This procedure is illustrated in the section Applications and Examples for a typical operating system. The displays and manipulations provided by the INFORM software are described in the next section.

Data tables. - When and how any of the three data-table sequences (data tables 1, 2, or 3) gets printed is determined by the INFORM operating mode. These modes were previously described and illustrated in figure 6. The table number, format type, and unit used for passive-mode printouts are governed by the coded data word (fig. 5) following the INFORM calling statement and the values of defaults DF0, DF1, and DF2. If the format type supplied by the coded data word is zero, DF0 will be used as the format type. In this event, if DF0 is not one of the four defined types of format (0, 1, 2, or 3), an error will occur and the program will revert to the interactive mode. A similar situation exists for the data table number and DF1. Default value DF2 will be used as the unit number whenever the coded data word supplies a unit number of 0. If printout is initiated in the interactive mode, the operands of the "form" command supply these parameters.

An example of formats 1, 2, and 3 for displaying data in tables is given in figure 7. Format 0 is designed for computer to computer communication dumps. It is identical



Enter INFORM
Input operators mode select switch
Enter passive mode
Enter interactive mode
Mode select switch status
Passive
Active
Is variable-trip-mode latch set ?
No
Yes
The variable-trip-mode is now in effect
Is trip variable at trip point ?
Yes
No
Was halt option selected ?
Yes
No
Stop
Operator inputs commands
Decode commands
Other
"Exit"
Execute command
Gather and print data for the selected data table
Input operators mode select switch
Mode select switch status
Passive
Active
RETURN

Figure 6. - Main flow of the INFORM program.

```
 K1                                                                K2
 ERROR     SETP       SPEED      ITGRL      VCMD       TACH       MOTOR


62500-05                                                        66732-08
91553-05  15088-01  15079-01  16028-04  10022-04  75394-02  15033-01
```

Data table format type 1

```
62500-05                                                        66732-08
18311-04  15079-01  15060-01  16030-04  10031-04  75302-02  15047-01
```

Data table format type 2

```
 K1    =  62500-05                                   K2    =  66732-08
ERROR  =  18311-04  SETP  =  15088-01  SPEED =  15070-01  VCMD  =  10031-04
```

Data table format type 3

Figure 7. - Typical examples of data table printout formats.

to the $2 dump format used for dumping SAMPLE data and is described in appendix C. In the type 1 format, a heading is printed first, then the data. In the type 2 format, only the data are printed. In the type 3 format, the variable name is printed, followed by an equal sign and the data. Note that for formats 1 and 2, each variable takes up 10 spaces of the line, while in format 3 each one requires 18. If an attempt to print an undefined data table is made, an error message will be issued, and the program will immediately revert to the interactive mode.

The number of variables printed on each line of a data table, their names, and the occurrence of any blanks within a given table are set by the operator. Three data-table sequences are provided and are referred to as data tables 1, 2, and 3. To define or alter one of the three stored data-table sequences, the data table being modified must first be in the open condition by applying the "@" or "(" command. Only one table may be open at any one time. Once open, the names, line-terminators (carriage-return-line-feeds), or blanks may be added using the "," and "-" commands. The last line of the table may be removed using the "↑" command. Once set, the named-location variables in a table will remain fixed unless a given name is redefined with the "space" command. The new named-location subsequently defined will then replace the old variable in the table.

A named location may occur any number of times within a table or tables. However, each occurrence will count as an entry in the table. Line terminators and blanks also count as entries. Up to 200 entries for table 1, and up to 86 entries each for tables 2 and 3 are allowed. These values, however, may be increased or decreased by a simple program change.

Once completed, a data table should be closed using the ")" command to prevent accidental modifications. Appendix C, as well as the APPLICATIONS AND EXAMPLES section should be referred to for a clarification of this procedure.

To minimize data skew between variables, the program gathers the current values for all the named locations of a data table at a fast rate before the table printout. For the SEL 810B this rate is 10.5 microseconds per point. An additional time lag (approximately 66 μsec plus 18 μsec per point for the SEL 810B) occurs between the issuance of a printout command and the time the first data point is fetched. Hence, to be time consistent, the values in core need be constant for only a short time immediately before a data table printout.

DATAO Displays

As previously described, updating of the display produced by the DATAO subroutine is governed by the main program and the operating environment established by the programmer. Thus, the operator has no control over how the display is produced, but he does have full control over what is displayed. This control is obtained by entering the DATAO control command processor of the interactive mode. This routine is entered from the normal INFORM control-edit command-acceptance routine by using the > INFORM control-edit command. The program will return > instead of the underscore to indicate readiness to accept DATAO control commands. A return from this routine to normal INFORM control-edit command acceptance is accomplished using the underscore DATAO control command. The DATAO subroutine control commands are listed in appendix D.

Displays from the DATAO subroutine are referenced by channel number. The 24 channels available are labeled 0 to 23. Assigned to each channel is a single named location, two EU values representing zero and full-scale outputs, an output unit number, and a Dac channel number for the output unit. Two operands are used to define a Dac because both a unit number and the Dac number are required for the SEL 810B system. Other systems may require only one of these parameters. This determination must be made by the INFORM programmer. All of these values for each channel must be defined by the operator before a display of that channel will occur. Only unit and channel numbers corresponding to system Dacs are accepted. Channel numbers must be defined sequentially, but, once defined, they may be modified, but not deleted, at any time. Only the highest number channel may be deleted.

For example, if we desired to create several DATAO displays of the SPEED parameter, we would proceed as follows:

```
_>
>A 00__SPEED,-15000,15000,'60,0
>A 01__,10000,11000,'50,0
>A 02__,0,15000,'60,1
>A 03__,0,-15000,'60,2
```

Here, we have created four channels to display SPEED. Notice the good use of the name-default option when defining channels 1 to 3. Thus, Dacs 0, 1, and 2 of Dac unit '60 have been assigned to DATAO channels 0, 2, and 3. For the example system, these Dacs are physically connected to strip-chart recorder channels 0, 1, and 2. Since these strip charts are always calibrated to linearly display 0 volt on the left and 10 volts on the right, the strip chart will linearly display SPEED in rpm. Strip chart 0 will display the full range of possible speeds from -15 000 to +15 000 rpm, negative speed being defined as opposite rotation from positive speed. Strip chart 1 will display only positive speeds, and strip chart 2 only negative speeds. Notice that we have flipped the axes on this chart so that negative speed dynamics will look the same as positive speed dynamics.

For this particular example, an area of interest might be the dynamic behavior around 10 500 rpm. To observe this region in detail, we have chosen to assign Dac 0 of unit '50 to DATAO channel 1. This Dac is physically connected to an X-Y recorder whose X axis is synchronized to a transient generator being used to evaluate the dynamics of our system. Since this display will overwrite itself, the change in transient behavior as system parameters are changed can be seen immediately. Using INFORM as shown above, this device was calibrated to display only values between 10 000 and 11 000 rpm, thus greatly improving the resolution and readability in the area of interest. Should interest later shift to some other area, such as the transition through zero, the X-Y recorder could be recalibrated by simply changing channel 1 to

```
≥C1 SPEED,-100,100,'60,0
```

Since everything except the calibration remains the same, extensive use of defaults could simplify the entry to

```
≥C1 ,-100,100
```

This shortened entry saves time and helps keep our concentration on the area of interest, namely, control performance, not Dacs and displays.

Notice that if time was available inside the computer as an integer-valued variable, this time parameter could be used to drive the X axis of the X-Y recorder. In that case we could connect a system Dac to the X axis, calibrate the recorder X axis for 0- to 10-volt operation, and define an appropriate DATAO channel.

There are some restrictions and considerations associated with the selection of EU values to calibrate the display. Generally, they are of sufficient latitude to be of no concern to the user, but an explanation of the process used by DATAO to produce a display may prove enlightening. For the ensuing discussion, define a maximum EU value (MAX) as the biggest EU value that can be scaled into a 16-bit, binary word; that is,

$$\text{MAX} = \text{Sf} \times 2^{15}$$

Let W be the EU value of a 0-volt output and X the EU value of a 10-volt or full-scale output.

Restrictions are necessary to prevent overflow of the integer arithmetic used when producing the display. These restrictions are

$$\frac{MAX}{2^{15}} < |W - X| < 32 \times MAX$$

and

$$MAX < |W|$$

Failure to meet these restrictions will result in the error message "SPOF" and a termination of the command.

A point to consider when selecting W and X is the accuracy of the display. Because of the integer arithmetic used, accuracy may suffer, depending on the scaled values of W and W - X. The operation performed by the program is as follows:

$$\text{Number output to the Dac} = \left[\left(\frac{V \times K1}{4}\right) + K2\right] 2^{K3}$$

where V is the scaled value of the output variable and K1 and K2 are defined as

$$K1 = \frac{10\ V}{X - W}\ Sf \times 4 \times 2^{-K3}$$

$$K2 = -\frac{K1 \times W}{4 \times Sf}$$

The K3 is selected by the program such that

$$2^{14} \leq |K| < 2^{15}$$

and has the restriction that

$$-15 < K3 < 15$$

This mathematics produces the desired output which is

$$\text{Output as a fraction of } 10\ V = \frac{(V - W)}{(X - W)}$$

The $V \cdot K1$ multiplication is 16- by 16-bit integer for a 31-bit product. The K2 is a 31-bit constant, and the 2^{K3}-multiplication is performed by a full arithmetic shift followed by a truncation to 15 bits. A more detailed analysis of this arithmetic is given in the PROGRAM STRUCTURE section.

Since K1 is held between 2^{14} and 2^{15}, the accuracy of the calculation depends mostly on the values of K2 and 2^{K3}. The 2^{K3} shift will effectively truncate when

K3 is less than 0. Hence, the accuracy of K2, or the offset voltage, will be poor if W/4Sf becomes too much smaller than $2|K3|$. The absolute value of K3 will become significantly negative as $|W - X|$ approaches $32 \times MAX$. Therefore, the poorest accuracy occurs for small magnifications (large W - X) of the signal around zero. Because of this, recorders having 1 volt as full scale should not be used directly by setting $|X|$ at 10 times the desired value. Instead, a times-10 attenuator should be used ahead of the recorder.

SAMPLE Subroutine Control

The SAMPLE subroutine operates in a fashion similar to DATAO; that is, the main program and operating system determine when samples are taken, but what is sampled is controlled by the operator. The operator, however, can only select items for sampling that the system has stored in core; that is, he cannot control analog-to-digital converters or similar I/O devices.

The command structure that controls SAMPLE is identical to that for DATAO. The SAMPLE control-command processor is entered from the normal INFORM control-edit command processor by using the < INFORM control-edit command. The program will return a < sign instead of the underscore to indicate a readiness to accept SAMPLE control commands. A return from this routine to normal INFORM control-edit commands is accomplished using the underscore SAMPLE control command. The SAMPLE control commands are listed in appendix E.

Variables sampled by the SAMPLE subroutine are assigned channel numbers from 0 to 79. Also assigned to each channel is a starting location for the core storage block to be used to save the samples for that channel. The length of the storage block is the same for all channels. This length must be set the first time the SAMPLE control-command processor is entered after a program load. The operator can accomplish this by using the B block-length definition command. (See appendix C.) The operator must also convey to the program whether his system is programmed to use SAMPLE in the averaging or nonaveraging mode. He does this by using the B command and specifying an averaging parameter in addition to the block storage length. If this averaging parameter is 0 or 1, SAMPLE will operate in the nonaveraging mode. If the system does use averaging, the operator must indicate the number of averages for each reading the system takes.

Programmers will want to refer to the programming section for details on how to interface SAMPLE to his system and how an advanced system can automatically switch from averaging to nonaveraging operation or vice-versa, depending on the averaging parameter entered by the operator.

The operator is also responsible for defining the storage area for the samples. He must do this for each channel by using the A or C commands. The size of the storage area needed will depend on (1) the number of samples taken for each reading, (2) the system averaging mode, and (3) the number of channels defined (i.e., the number of named locations being sampled). Items (1) and (2) affect the block storage length that the operator should have defined with the previous B command. The block length should equal the number of samples per reading for nonaveraging and twice this number for averaging.

For example, consider a system in which a reading consists of a single transient. If we wish to sample the transient data by taking 100 samples spaced 0.1 second apart, we would define a block length of 100 and the number of averages as zero. The 0.1-second spacing is beyond the control of the INFORM software, as this value would be determined by the system established by the programmer and the manner in which SAMPLE was called; that is, the system would have to provide independent means to initiate transients and change timing.

If the system is intended for averaging operation and produces 16 identical transients that are to be averaged to produce a mean transient, we must double the block length to 200 and define the number of averages as 16. The extra block length is needed by the program to store the 16 values received for each channel as a double-precision integer sum.

The following is an example of how to define a SAMPLE channel for a transient that is to take 100 points of SPEED in a system not using averaging:

```
_<
 <B BLOCK SIZE = 100
 # AVERAGES = 0
<A 00  SPEED,'40000,  '40144
<"_"
_
```

Note that the B command is used to define a block size and number of averages before we proceed. If we decide to change these numbers at a later time, the B SAMPLE command will have to be reissued. This B command should not be used carelessly, however, since changing the block length may invalidate the starting storage location defined for the channels. Also, changing the number of averages may affect the $1 and $2 INFORM commands used to dump the sampled data. The user should carefully read the descriptions and warnings listed for this command in appendix E.

To add more variables, say, ANGLE and TACH, to the list of SAMPLE channels, proceed as follows:

```
_<
<A 01__ANGLE,,__'40310
<A 02__TACH,,__'40454
<"_"
_
```

Note the good use of the starting storage location default option to keep track of storage.

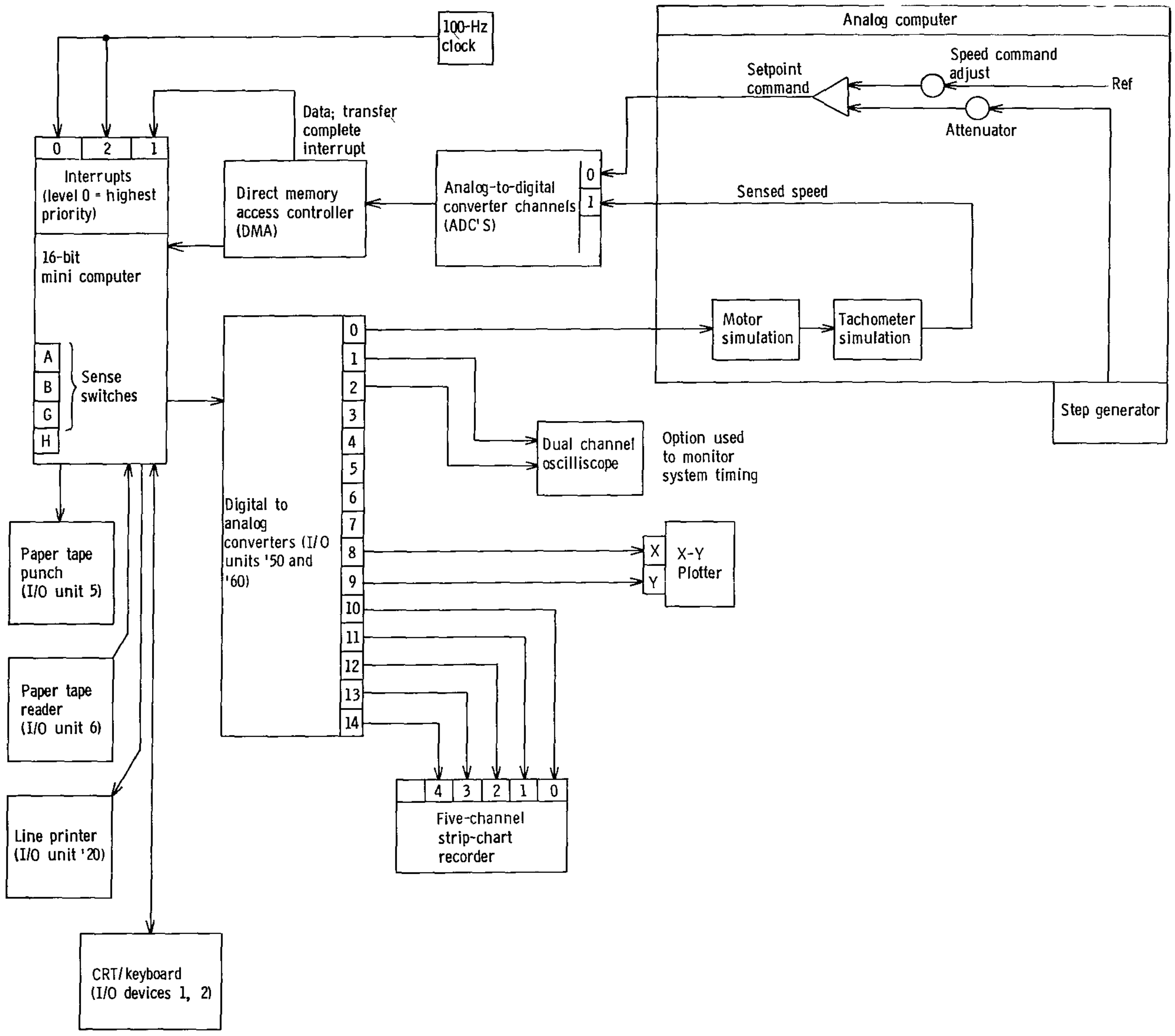
To repeat the above example for a system that uses averaging and repeats the transient 16 times, the procedure is

```
_<
 <B BLOCK SIZE = 200
 # AVERAGES = 16
<A 00__SPEED,'40000,,__'40310
<A 01__ANGLE,,__'40620
<A 02__TACH,,__'41130
<"_"
_
```

Note that, since an average is being computed instead of a simple store, each sample requires two storage locations. We, therefore, defined the block size as 100 × 2 or 200. The effect of the block length increase is seen whenever the program displays the next available location. These differences must be taken into account when using the B command to change block lengths and number of averages, as issuing the B command will not automatically redefine the storage location of all channels.

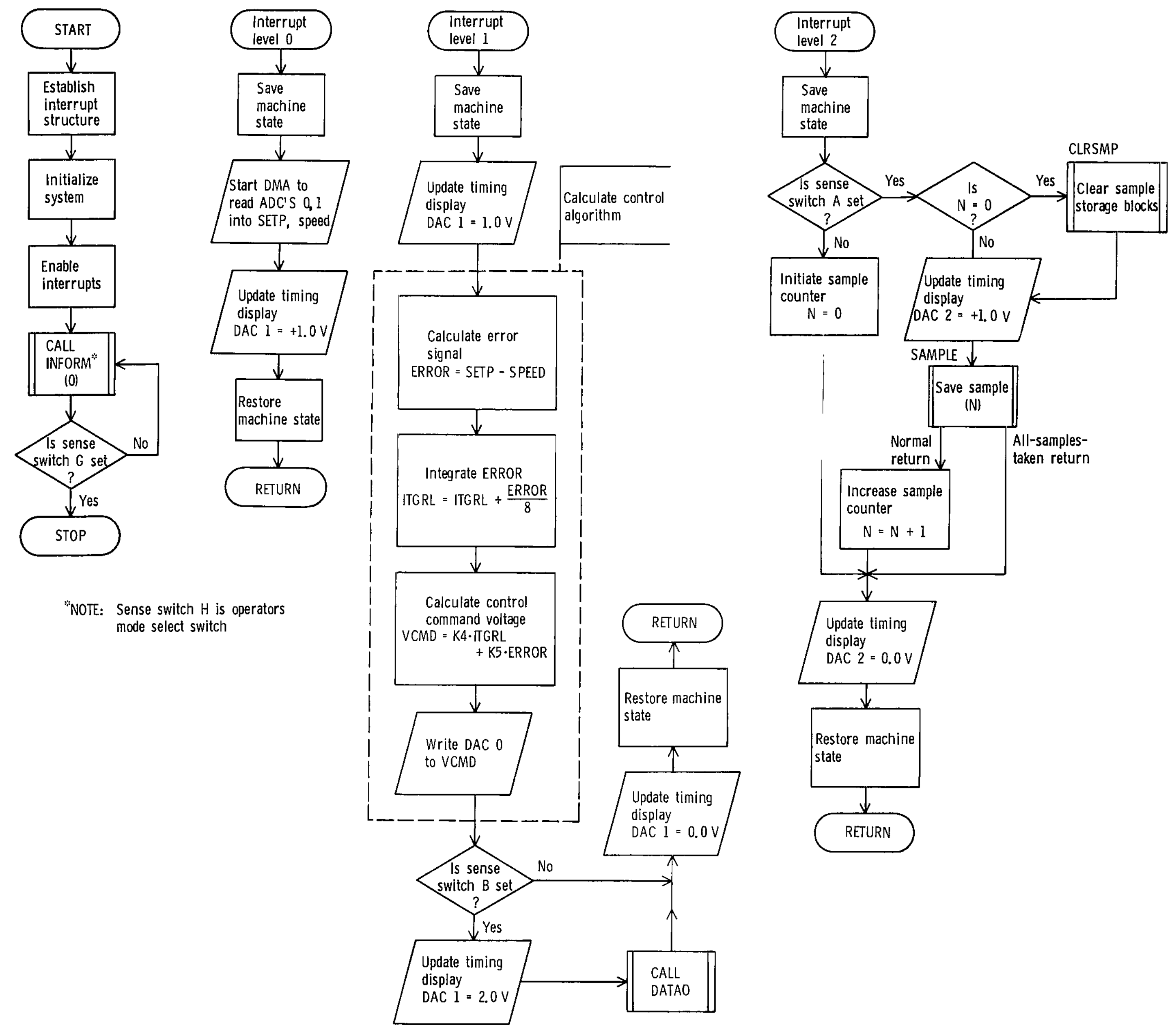
APPLICATIONS AND EXAMPLES

Potential application for the INFORM software is limited only by the imagination of the user. The following simple example illustrates a typical manner in which INFORM is used. Assume that the system consists of an analog simulation of a motor and tachometer pair whose speed is controlled digitally by a 16-bit minicomputer. (A block diagram of the system is given in fig. 8, and the flow chart of the digital program is given in fig. 9.) A dual-channel oscilloscope is included in the system to demonstrate the speed of execution of the program. By writing Dacs 1 and 2 to different levels at various points in the program, computer operation can be observed as it



100-Hz clock
0
2
1
Interrupts (level 0 = highest priority)
16-bit mini computer
A
B
G
H
Sense switches
Data; transfer complete interrupt
Direct memory access controller (DMA)
Analog-to-digital converter channels (ADC'S)
0
1
Analog computer
Setpoint command
Speed command adjust
Ref
Attenuator
Sensed speed
Motor simulation
Tachometer simulation
Step generator
Digital to analog converters (I/O units '50 and '60)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Dual channel oscilliscope
Option used to monitor system timing
X
Y
X-Y Plotter
4
3
2
1
0
Five-channel strip-chart recorder
Paper tape punch (I/O unit 5)
Paper tape reader (I/O unit 6)
Line printer (I/O unit '20)
CRT/keyboard (I/O devices 1, 2)

Figure 8. - Motor/tachometer speed control system of example problem.



START
Establish interrupt structure
Initialize system
Enable interrupts
CALL INFORM* (0)
Is sense switch G set ?
No
Yes
STOP
Interrupt level 0
Save machine state
Start DMA to read ADC'S 0, 1 into SETP, speed
Update timing display DAC 1 = +1.0 V
Restore machine state
RETURN
Interrupt level 1
Save machine state
Update timing display DAC 1 = 1.0 V
Calculate control algorithm
Calculate error signal ERROR = SETP - SPEED
Integrate ERROR ITGRL = ITGRL + ERROR/8
Calculate control command voltage VCMD = K4·ITGRL + K5·ERROR
Write DAC 0 to VCMD
Is sense switch B set ?
No
Yes
Update timing display DAC 1 = 2.0 V
CALL DATAO
Update timing display DAC 1 = 0.0 V
Restore machine state
RETURN
Interrupt level 2
Save machine state
Is sense switch A set ?
Yes
No
Is N = 0 ?
Yes
No
CLRSMP
Clear sample storage blocks
Initiate sample counter N = 0
Update timing display DAC 2 = +1.0 V
SAMPLE
Save sample (N)
Normal return
All-samples-taken return
Increase sample counter N = N + 1
Update timing display DAC 2 = 0.0 V
Restore machine state
RETURN

*NOTE: Sense switch H is operators mode select switch

Figure 9. - Flow chart for computer program of example problem.

steps through a typical update cycle. Figure 10 shows a typical trace where DATAO and SAMPLE are setup for five channels.

In this example, much of the INFORM software is exercised. Hence, in addition to acquainting new user's with the program, this example can be used as a test procedure for those wishing to validate their software. To eliminate confusion, operator-entered carriage returns are shown herein as "J". The commands we will be using are listed in detail in appendixes C to E and are summarized in appendix I.

The first step is to define the digital program variables. This is done using the INFORM space command:

```
_ERROR **'614,15000/32768J
       --
_SETP **J
      --
_'000615
--------
_SPEED **,_'000616J
       --_--------
_ITGRL **,_'000617,1/1B3J
       --_--------
_VCMD **,_'000620,10/1SJ
      --_--------
_!J
```

Notice that various default options have been used to indicate that ERROR, SETP, and SPEED have the same scale factor and are located in successive memory locations. Since these variables are in the common core and will not change as the program is updated, we have produced a binary-dump on the paper-tape unit (DF9) so that the next time the program is run the definition process will not have to be repeated.

As a result of issuing the ! command, the paper tape was punched and the machine responded:

```
VERIFY?_YJ
--------
```

We replied, "Y," after mounting the tape just produced onto the paper tape reader (DF10).

We should now like to observe the problem in operation. To do so, the strip-chart recorders will be used. Before they are used, they should be calibrated to operate between 0 and 10 volts. To do this a calibration voltage is needed. There is no better source for such a voltage than the DATAO display system. In fact, if DATAO is used, the user need not remember that the system uses 0 and 10 volts, nor worry about Dac counts per volt, etc. He need only consider zero and full scale. To generate a calibration signal, dummy variable X is defined, and DATAO is set up to display X on all channels. By using the INFORM scale and store command, the value of X can be

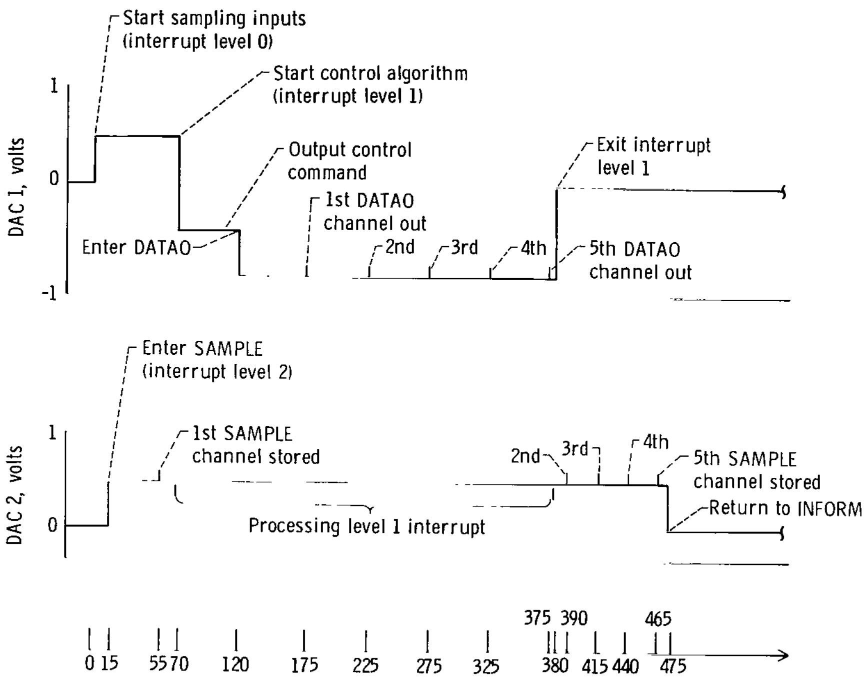


Figure 10. - Scope display showing digital program timing for example problem.

changed and thus cause DATAO to generate a known display anywhere between zero and full scale.

The procedure is as follows: Location '500 is unused by the program, so it can be used for X.

```
_X **'500,1/32000ȷ
```

Now set X = 0.

```
_"_'000500 WAS=_'143772_0,NOW=_'00000
```

Notice that we have chosen full scale of X = 1 to correspond to 32 000 counts. Using 32 000 gives sufficient resolution in setting X (i.e., 1 part in 32 000), yet is not so large that 1 will not fit into our 16-bit machine (32767 being the maximum integer value). Any value could have been chosen, except that, if we had defined X as being 1/1S, a problem would have arisen because 1S does not fit the machine; that is, X would always have to be a fraction. Thus, an error message would be received if an attempt were made to set X = 1.0.

The DATAO channels will now be defined. The Dac unit connected to the display is unit '60.

```
_>ȷ
>A 00_X,0,1,'60,10ȷ
>A 01_,0,1,'60,11ȷ
>A 02_,0,1,'60,12ȷ
>A 03_,0,1,'60,13ȷ
>A 04_,0,1,'60,14ȷ
```

We now adjust the recorder's offset to display zero, then return to the INFORM command mode so we can set it for full scale and adjust the recorder gain:

```
>"_"ȷ
_X"_'000500 WAS=_'0000000_1,NOW=_'076400
```

The recorder's sensitivity is now adjusted to display full scale.

With the calibration complete, the recorders can now be set up to display our problem. Using named locations, the variables SETP, ERROR, SPEED, ITGRL, and VCMD will be assigned to Dacs 10 to 14, respectively:

```
_>J
>@J
>A 00_SETP,0,1000J
>A 01_ERROR,-1000,1000J
>A 02_SPEED,0,1000J
>A 03_ITGRL,-4,4J
>A 04_VCMD,-5,5J
>"_"J
```

Note that typing has been saved by defaulting the Dac unit and Dac numbers. This can be done because these values were defined when the calibration procedure was performed.

The first portion of figure 11 shows the resulting display. This display indicates that the system is unstable. Obviously, K4 and K5 were chosen poorly. The system should be stable. To locate the problem we will check K4 and K5 for proper scaling. They should be 3.0 and 2.0/3000.0, respectively. Since K4 and K5 are not in the common core, we will use the power of the sequentially formed arithmetic expression (SFAE) to define their addresses. To do this, the variable X is redefined as having the relocatable load address ('1045) of the control algorithm program. Then the assembly listing address of '100 for K4 can be used as follows:

```
_X REDEF_'1045,0J
_K4 **X+'100,VCMD/ITGRL*1/1B4J
_K5 **,_'001146_VCMD/ERROR*1/1B5J
```

Since K5 follows K4, the K5 address was defaulted. Note also the fancy scale-factor manipulation used for K4 and K5. Since K4 converts ITGRL scaling to VCMD scaling, VCMD/ITGRL (the ratio of the scale factors) was used in the scale-factor definition. To fit reasonable values of K4 into an integer value, it was necessary to further scale K4, hence, the factor 1/1B4. The program compensates for this additional scaling by an arithmetic left shift of four after the multiplication. (See the control equation in fig. 9, p. 29.) Similar reasoning applies to the K5 scale factor.

We will now display K4 and K5.

```
_K4=_10000-3
_K5=_66667-8
```

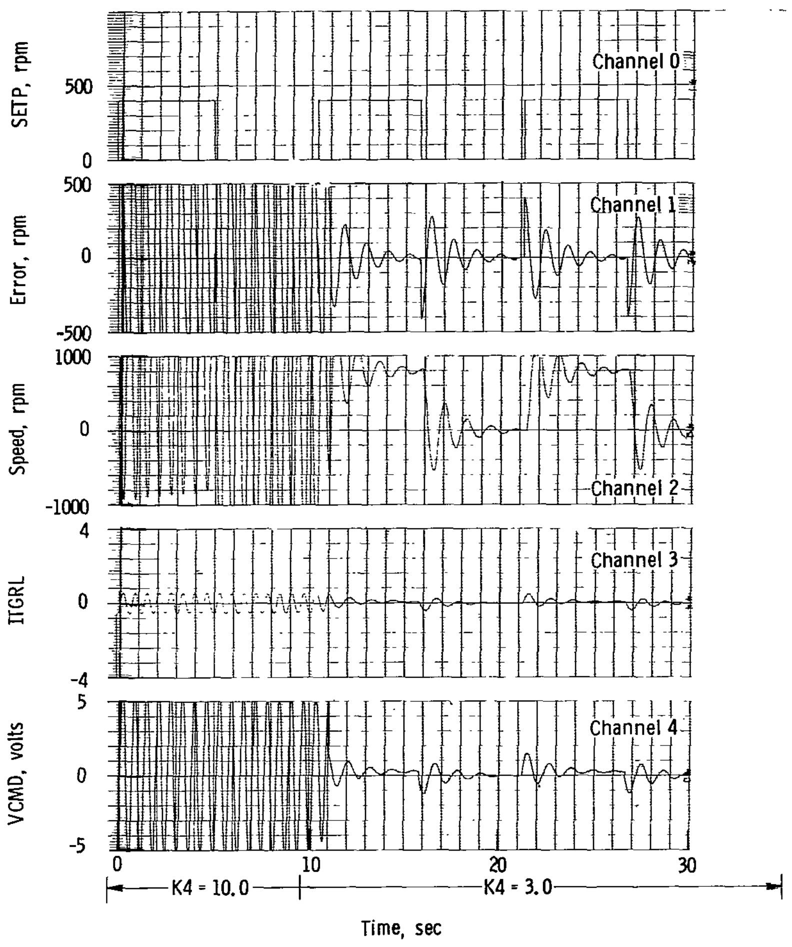


Figure 11. - DATAO traces for example problem showing effect of K4 change on system. K5 = 6.667×10^{-4}.

We had intended K4 to equal 3.0, but it must have been entered improperly when scaling K4 for the program's data statement. The problem can easily be fixed by

```
_K4"_'001145_WAS=_'040000_3.0,NOW=_'011463
```

Figure 11 shows the result of this change. Notice that the control program keeps running while we are making changes.

The resulting system response is nice, but could be better. INFORM and DATAO can be used to find more optimum values. Let's increase K5 by a factor of 10.

```
_K5"_'001146_WAS=_'002000_K5*10.,NOW=_'024000
```

Notice how the SFAE simplifies increasing or decreasing by factors. The resultant trace is shown in figure 12. It is now obvious that the scales for the SPEED (channel 2) and ITGRL (channel 3) traces can be expanded to make them readable. This is accomplished as follows (with fig. 13 as the result):

```
_>J
>C 02,,-500,500J
>C 03,,-.25,.25J
```

We will now get a listing of what the revised channel definitions are and save them by means of a relocatable binary-dump tape. The ? DATAO command is used:

```
>? 1J
00__'_60__10__SETP_____00000_00_____10000-01
01__'_60__11__ERROR___-10001-01_____10000-01
02__'_60__12__SPEED___-50001-02_____50000-02
03__'_60__13__ITGRL___-23001-05_____25000-05
04__'_60__14__VCMD_____00000_00_____10000-03
>'J
VERIFY?_YJ
>"_"J
```

The display for the channel definitions shows us first the channel number, then the DAC unit number, the DAC number, the named location, the EU value at 0 volt, and the EU value at 10 volts, respectively.

Next INFORM is used to generate some steady-state data. First, a data table is built by

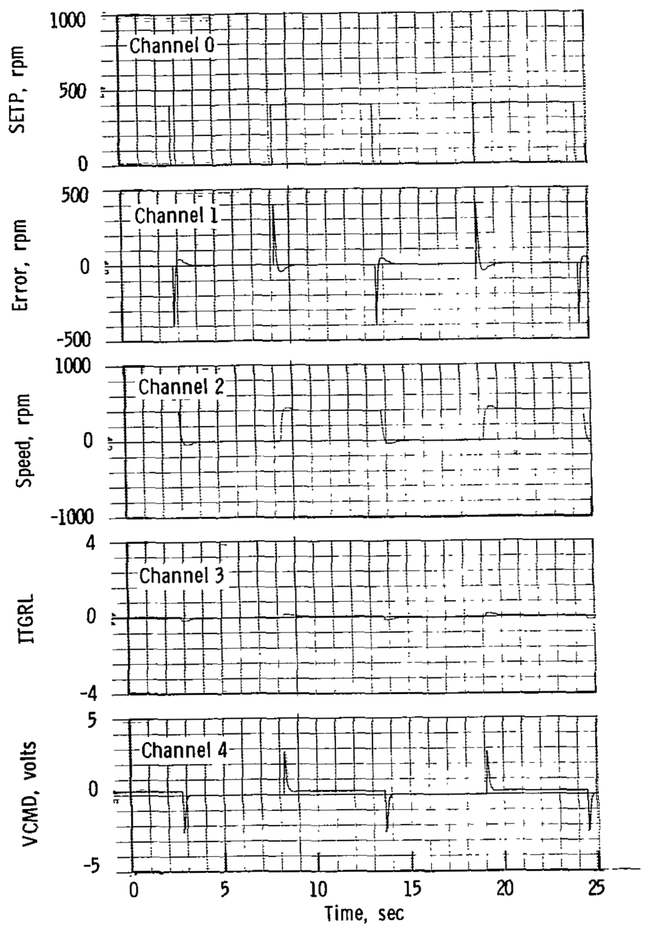


Figure 12. - DATAO traces for example problem with K4 and K5 optimized using INFORM. K4 ≈ 3.0, K5 = 6.667x10^{-3}.

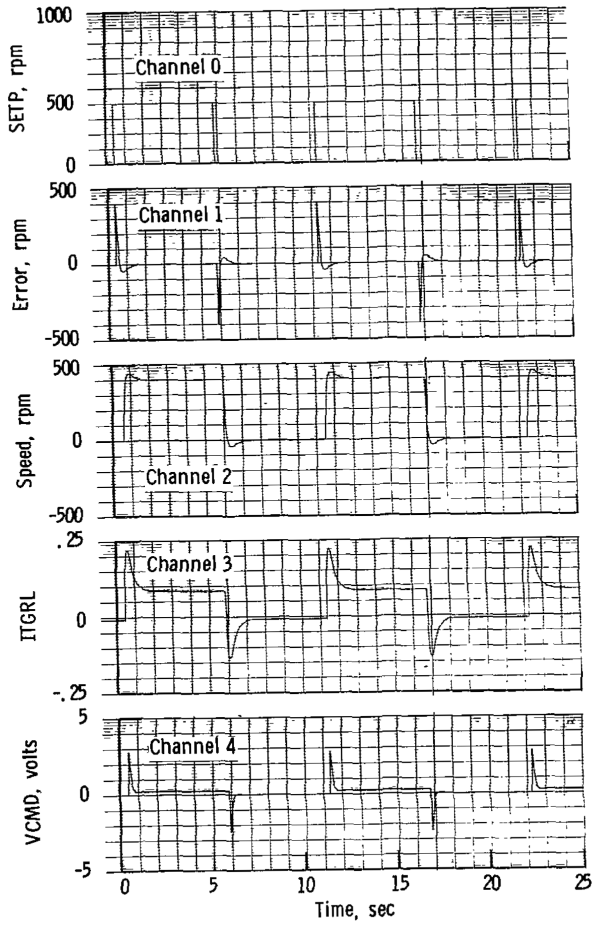


Figure 13. - Scale-expanded DATAO traces for example problem with K4 and K5 optimized using INFORM.

```
_@J
_K4,_01
_K5-_02_00
_ERROR,_01
_SETP,_03
_SPEED,_03
_ITGRL,_04
_VCMD,_05
_)J
```

The form command could now be used to print each update of the table as we change the set points that generate the steady-state data. However, to illustrate the passive mode use of the program, a different procedure will be followed. Set the operator's mode-select switch (sense switch H) for passive mode printouts. Note that the passive mode will not be entered until INFORM is left or a data table is printed (see fig. 6, p. 19). We will now set DF0 through DF2 to set up the passive mode printout to print data table 1 in format 2 on the line printer (unit 5):

```
_% 0,2J
_% 1,1J
_% 2,5J
```

The table is started by printing the first point, complete with heading:

```
_\ 1,1,5J
```

Since the operator's mode select switch was set for passive mode, the INFORM subroutine was exited on completion of this printout. The system sense switch G is now used to control the entry of INFORM, since each time INFORM is entered a new line of data will be printed in the table. Before each printout, we will change the setpoint (SETP), then momentarily pulse sense-switch G to enter INFORM and print one data table line. Figure 14 is the result.

The last line of the table reveals a serious problem: The steady-state error (ERROR) is excessively large. We will re-enter INFORM's interactive mode to examine the problem. First, the operator's mode select switch is set to the interactive mode; then system sense switch G is reset to enter INFORM.

Suspecting that the problem is caused by a temporary overflow when calculating VCMD, we will use INFORM's variable-trip-mode option to attempt to catch an overflow just before it occurs. We will set the trip to occur when VCMD exceeds 9.8.

```
  K1          K2
 ERROR       SETP        SPEED.      ITGRL       VCMD

29999-04   66667-07
18311-04   91553-05  -91554-05  -51271-07  -33570-07

29999-04   66667-07
18311-04   15001-01   15003-01   33008-05   10022-04

29999-04   66667-07
00000 00   30066-01   30066-01   66455-05   19934-04

29999-04   66667-07
27466-04   45081-01   45053-01   99780-05   30115-04

29999-04   66667-07
91553-05   60059-01   60049-01   13303-04   39969-04

29999-04   66667-07
91553-05   75018-01   75009-01   16638-04   49973-04

29999-04   66667-07
18311-04   90079-01   90060-01   19966-04   60016-04

29999-04   66667-07
27466-04   10508 00   10506 00   23311-04   70111-04

29999-04   66667-07
91553-05   12007 00   12006 00   26648-04   79999-04

29999-04   66667-07
11724 00   13508 00   17834-01   62815-04  -29948-04
```

Figure 14. - Printouts produced for the example problem steady-state points generated by INFORM working in passive mode.

```
_[ VCMD>9.8ȷ
HLT?_Nȷ
```

Since the halt option has been declined, a printout of the data table will occur each time trip occurs. For readability we can print the heading on our table by setting DF1 for printout in format 1:

```
_% 1,1ȷ
```

Now all we need do is exit INFORM to initiate the variable trip mode:

```
_.ȷ
```

Note that system sense switch G must be kept on reset so that INFORM is entered repetitively. This is necessary so that INFORM can test each update cycle for the trip condition. After putting some disturbances into the system, figure 15 results. The system has been caught in a malfunction mode. A user could now use the halt option to stop the program; and then use INFORM or a suitable debug procedure to locate the defect. Stopping execution on the cycle that caused malfunction eliminates needless probing of cycles that operated correctly.

Details of program repair are left to the reader. To terminate the variable trip mode, the operator's mode select switch is set to interactive mode. Once the underscore is obtained, the following command is issued:

```
_]ȷ
```

To complete the exercise of the INFORM software, SAMPLE will be used to capture a typical system transient. One hundred points will be collected for a single transient; that is, the averaging mode will not be used. We now enter the SAMPLE command processor:

```
_<ȷ
```

after which the first item should be to define the block size and number of averages:

```
<Bȷ
BLOCK SIZE=_100ȷ
#_AVERAGES=_0ȷ
```

Then, the channels are defined. Core starting from location '40000 will be used to save the samples:

```
 K1          K2
ERROR       SETP        SPEED       ITGRL       VCMD

29980-05   10000-03
10071-03   14595 00   14585 00   32473-04   98026-04

 K1          K2
ERROR       SETP        SPEED       ITGRL       VCMD

29980-05   10000-03
14923-02   13593 00   13444 00   29946-04   99728-04

 K1          K2
ERROR       SETP        SPEED       ITGRL       VCMD

29980-05   10000-03
11219 00   14149 00   29306-01   16887-04   98550-04
```

Figure 15. - Printouts produced for the example problem by INFORM generating in variable trip mode.

```
'040144      99312-02    10001-03    10112-03    10274-03    10555-03
'040151      10446-03    10529-03    10582-03    10658-03    10693-03
'040156      10736-03    10767-03    10803-03    10832-03    10853-03
'040163      10840-03    10810-03    10841-03    10853-03    10822-03
'040170      10802-03    10776-03    10727-03    10730-03    10693-03
'040175      10657-03    10513-03    10099-03    98439-02    94666-02
'040202      91895-02    87158-02    82465-02    79651-02    75969-02
'040207      72418-02    68848-02    65169-02    61983-02    58777-02
'040214      55573-02    52466-02    49536-02    46782-02    44037-02
'040221      41382-02    38818-02    36346-02    34139-02    32043-02
'040226      30029-02    28011-02    26276-02    24445-02    23163-02
'040233      21686-02    20232-02    18901-02    17670-02    16704-02
'040240      15839-02    14923-02    14191-02    13458-02    13000-02
'040245      12543-02    12085-02    11739-02    11536-02    11303-02
'040252      11261-02    11169-02    11169-02    11076-02    11303-02
'040257      11444-02    11627-02    11739-02    11993-02    12360-02
'040264      12726-02    14282-02    17304-02    20782-02    24445-02
'040271      28290-02    32135-02    35889-02    39642-02    43396-02
'040276      46875-02    50446-02    53925-02    57312-02    60516-02
'040303      63721-02    66833-02    69855-02    72150-02    75256-02
'040310      10638-03
```

Figure 16. - Display of data collected by subroutine SAMPLE. Display was produced using the INFORM "/" list core command.

```
<A 00_ERROR,'40000,_'040144
<A 01_SPEED,,_'040310
<A 02_ITGRL,,_'040454
<A 03_VCMD,,_'040620
<"_"J
```

System sense switch A is used (fig. 9, p. 29) to start collecting data. Since the 100-hertz system clock patched into interrupt level 2 is also the system update clock, a sample will be taken after each update cycle. It will then take 1 second to sample the 100 points. We can use timing DAC2 to initiate a transient so that the disturbance can be synchronized with the samples. Since we do not want DATAO to interfere with the taking of the samples (fig. 10), system sense switch B is reset to suppress the DATAO display. To display the data collected on the line printer (unit 5) do

```
_$1 ,,5J
_$2 ,,5J
```

However, as explained under the $1 and $2 command descriptions in appendix C, these printouts are not really intended for operator reading. (The $1 and $2 printouts are also shown in appendix C.) The INFORM / list core command can be used to see the sampled data. To use this command to view the samples from SAMPLE channel 1, we first note that these samples start at location '40144. If one forgets where the SPEED samples were stored, the ? display SAMPLE definitions command can be used to supply this information:

```
_>J
>? 1J
BLOCK_SIZE=_00100
#_AVERAGES=_00001
 00___ERROR__'040000___'040144
 01___SPEED__'040310___'040310
 02___ITGRL__'040620___'040454
 03___VCMD___'041130___'040620
```

Furthermore, it is usually desired to descale the values before they are listed. The SPEED scale factor should be used. Hence, the command issued is

```
_/ '40144,'40144+100,1,SPEED,5J
```

The display is shown in figure 16.

Had we been using SAMPLE in the averaging mode, we would note that the data would be stored by SAMPLE as double-precision integers. Furthermore, it would represent the sum of the number of averages taken. Therefore, the number of averages should be included in the scale factor to divide it out. For example, assuming that channel 1 starts at '40310 and that 16 averages are taken, the command to duplicate the above printout would look like this:

```
_/ '40310,'40310+200,6,SPEED/16,5ȷ
```

PROGRAM STRUCTURE

The program contains several subroutines that perform specific functions required by the various commands. Some of these subprograms are part of external supportive software packages required to implement the program. These external routines consist of floating-point arithmetic routines, I/O routines which input and display octal/decimal or integer numbers and, I/O routines that produce and read relocatable binary-dump tapes. The math routines are usually supplied with most machines as part of the Fortran library. The special I/O routines are described in appendix F.

INFORM Structure

The program retains the information defining the named locations in parallel tables. The table position of the named location in use by the command is called the "indexing variable." This indexing variable is always maintained in the INDX location.

The address of the named location is stored in the LOC table, and the name itself is stored in two tables called NAM1 and NAM2. The name consists of five, 6-bit truncated ASCII characters which constitute a 30-bit word length. The least significant 16 bits are contained in NAM2, and the most significant 14 bits are in the least significant 14 bits of NAM1. The most significant two bits of NAM1 are maintained as zeros.

The scale factor assigned to the named location is saved in table SF as a 31-bit, single-precision, floating-point, real value. The format used is consistent with the standard arithmetic software supplied with the SEL 810B computer on which this program was developed. To avoid the overflow and underflow problems of integer arithmetic, calculations which involve EU values are performed using floating-point, real-arithmetic software.

Information for the three different INFORM data-table sequences is stored in TAB1, TAB2, TAB3, and LGTH, where TAB1, TAB2, and TAB3 are format tables which correspond to data tables 1, 2, and 3, respectively. By being matrices of indexing variables, they indicate which named locations are to be displayed and in what position. TAB1 is 100 locations long and can contain 200 indexing variables at two per word. TAB2 and TAB3 each contain 43 locations and can contain 86 indexing variables. An indexing variable of 255 ('377) denotes a line terminator for a line of the data table, and an indexing variable of zero indicates that a blank space is to be placed in the data table output.

Operator errors are processed by branching to a common routine called EROR. This routine cancels the command in progress and returns to the beginning of the current command processor by using the RTRN location. RTRN contains a program address and is set at various places in the program to establish the point where this error processor should return. RTRN also serves as the return address for the command processing routines, thus making them independent of the command decode sequence used to enter them. RTRN is set by executing a store place and branch (jump to subroutine) to RTRN followed by an immediate "branch indirect" to RTRN (return from subroutine) instruction. This method was used because it is the only way to save the program counter on the SEL 810B, a necessity when implementing a dynamic unconditional jump instruction. The EROR and command processing routines then simply execute a branch to the address in RTRN.

Program Flow Charts

The structure of the INFORM software can be easily understood by referring to the flow charts of appendix G. Subroutines contained with the INFORM software have a separate section for their flow charts in this appendix. Descriptions of the functions of all the subroutines used are contained in appendix H. When subroutines are encountered in program flow, a brief description of the essential functions being performed is given in the block to ease reading of the flow chart. It should be pointed out, however, that other essential functions may also be performed by the subroutines. The reader should refer to the particular subroutine flow chart to ascertain its complete operation. The variable arguments supplied to and received from a subroutine are shown in parentheses by the ENTER and RETURN statements. The definition of these arguments is clearly spelled out within the subroutine flow chart. What these arguments are equivalent to for any particular use of a subroutine by a calling program should be clear from the statement of subroutine function in the calling program's flow chart.

Portions of the flow charts are contained within dashed blocks. An overall description of what occurs within the dashed block is given at the point where the program flow initially enters the dashed block. Inside the block is more detailed information to aid an assembly-language programmer in achieving the required operation without violating program conventions or assumptions. A quick overview of program operation can be obtained by only reading the descriptions for the dashed in blocks of the flow chart.

The starting point for each command is located by finding the offpage connector symbol (○) containing the mnemonic listed under the Programmer's Flow Chart Reference section of each command description. The < and > commands of INFORM, however, are special, as they transfer the program to a different command interpreter section; that is, they change the value of the RTRN address so that when commands are completed, commands used to control the DATAO or SAMPLE subroutines and not INFORM's control-edit commands are expected. These two commands, taken together, may be treated as a completely different program that merely uses the INFORM-defined, named-location tables and some of the INFORM subroutines. This was done so that the program can be easily shortened by eliminating either SAMPLE, DATAO, or both, together with their somewhat common command processor. A more detailed description of this processor is given in the next section.

This knowledge, together with the command and subroutine description sections should eliminate the need for any further explanation of the INFORM flow chart. The flow charts for the DATAO and SAMPLE subroutines and command processors (< and > commands) are listed in the final section of appendix G.

DATAO and SAMPLE Command Processor

DATAO and SAMPLE control commands are processed by a common command interpreter, the CMD subroutine. This routine interprets the command key character and branches to the appropriate command processing routine. The routines are common for the @, C, D, *, !, and _ commands as only some operands differ. The ?, A, and B commands require separate processors, as the operations to be performed vary considerably.

The ? and B commands are separated by having different returns from the CMD subroutine. Hence, the routines to process these commands occur under the SPL (< command) and ODAT (> command) INFORM command processing sequences because these sequences call the CMD subroutine.

The "A" command has some processing functions in common for both DATAO and SAMPLE. The different portions are effected by having the common portion (the ADDO processor) execute a dynamic jump to subroutine. This jump is changed when the operators for the common commands are changed in the SPL or ODAT sequences. The names given these dynamic operators are X01, DA1, NCH, CHIX, and NBUF.

SAMPLE Structure

The channel definitions for the SAMPLE program are stored in the TORG and SIDX matrices. The position of the values in these matrices is equal to the channel number they define. The matrix of indexing variables, SIDX, is used to obtain names and locations of the variables sampled in a similar fashion as TAB1, TAB2, and TAB3 are used to obtain names and locations for INFORM data tables. The matrix TORG contains the starting addresses of the storage blocks used to save the samples. The operation of SAMPLE is straightforward and requires no further explanation.

DATAO Structure

The DATAO program is considerably more complex. As was done for SAMPLE and the data tables of INFORM, a matrix of indexing variables is saved in DIDX to point to the proper named locations for each channel. In order to produce an output, DATAO must make a calculation of the form

$$\text{Voltage output as a fraction of full scale} = \frac{V - W}{X}$$

where V is the current value in EU's, W is the EU value at the origin, and X is the EU value at full scale. It is this calculation plus the need for a fast output that causes the complexity.

If the machine were to perform this calculation using software floating-point, real-arithmetic routines and the EU values at zero and full scale as supplied by the operator, the calculation would take too long to be useful as a dynamic display. Therefore, this calculation must be translated into one composed solely of arithmetic, which can be speedily handled by the machine's hardware.

One might think that this is simply a matter of scaling the EU values and performing integer arithmetic. However, this method creates problems because of possible subtraction overflows. Some sort of scaling would be necessary to prevent this. If this scaling had the effect of reducing the scaled magnitude of $|X - W|$, small values of $|X - W|$ (corresponding to large magnifications of V), would produce inaccurate values because of the truncation of a small number to its nearest integer. This is significant because it may be desirable to use DATAO to distinguish a changing bit pattern among a few least-significant bits of a large word. If this word represents something such as a program address, the scale factor will be unity, and $|X - W|$ will surely be small. If the scaling were to increase the magnitude of $|X - W|$, a corresponding increase in the $|V - W|$ calculation would have to occur. But V is unknown, making it impossible to insure that the V - W calculation will not overflow without testing it.

All this complication adds is unnecessary calculation time, for, if the required calculation is in the form

$$Y = A \times V + B$$

A and B can be scaled such that proper operation will be assured if W and $|X - W|$ meet modest requirements. The computation is performed as

$$Y = \left(V \times K\frac{1}{4} + K2\right) 2^{K3}$$

The $V \times K1$ multiplication is done first to yield a 31-bit product. This is then shifted right by a full right arithmetic shift two places to divide by 4. This shift will also insure that the absolute value of the result is less than $(1/4) \times 2^{30}$. A double-precision integer whose absolute value is less than $(3/4) \times 2^{30}$, K2, may then be added to the shifted product without fear of an overflow. The multiplication by 2^{K3} is then accomplished by means of a full right or left arithmetic shift, depending on a dynamic shift instruction that has been stored in the DEXP matrix. This shift instruction, or K3, is selected such that $2^{14} \leq |K1| < 2^{15}$ where K1 is an integer defined as

$$K1 = \left(\frac{10 \text{ volts}}{X - W}\right) 4 \times Sf \times 2^{-K3}$$

truncated to 16 bits.

Then K2 becomes equal to $-K1 \times W/4Sf$. If MAX is defined as the maximum scaled value of V that can be stored in a 16-bit word, the restrictions that $|K3|$ be less than 16 (i.e., a single shift instruction) and that K2 be less than $(3/4)2^{30}$ translate into

$$\frac{MAX}{2^{25}} < |W - X| < MAX \times 2^5$$

and

$$|W| < 3MAX$$

It is doubtful that it would ever be desired to exceed these bounds since they lie outside the possible values of V.

The required coefficients of K1 and K2 are stored in the three matrices DK1, DK2, and DK2B. The shift instruction stored in the DEXP matrix is actually picked out of the matrix, stored in the program path, and executed for each channel. The output device number and Dac channel number are selected in a similar fashion by executing an output instruction that was built by the machine at channel-definition time and stored in the DUNT and DCHN matrices. The details of this operation obviously depend on the I/O structure of the machine on which the program is to be run.

PROGRAMMING NOTES

An attempt has been made to divorce the flow chart from dependency on any particular machine, yet retain sufficient detail to facilitate implementation on machines other than the SEL 810B. Since considerable programming effort and debugging time has gone into the program structure to make it both maintainable and free of unexpected operations, it is highly recommended that those operations outlined in the flow chart be strictly adhered to.

Handling of Input and Output

It may be noted that many of the commands contain the capability to change the I/O device on which the input or display is to occur. This versatility is derived from the special I/O routines listed in appendix F. These routines are part of a CIPHER I/O library developed at Lewis, and are used for all input and output operations. They are designed to make all devices operate like a teletypewriter using an ASCII code. This capability was included since the required code is minimal because of the way that I/O devices are interfaced to the SEL 810B computer. Persons using machines with less capability may wish to delete the unit change option to possibly reduce the core requirements of the I/O routines. Note however, that the I/O device independency does enable multiple use of each I/O routine and that the entire set of routines is thus very compact. Eliminating this device independency may so specialize the use of each routine that more routines will be required to perform the complete computing job. The additional routines may actually use more core than is saved. Since these routines are very versatile, it may be more beneficial to save core by applying these routines to tasks outside of INFORM and tnus shorten the existing main program instead of vice-versa.
stead of vice-versa.

Modifications to the INFORM program to accomplish this change are minimal, as all I/O device numbers are obtained through the UNIT subroutine and are thus easily identified for exclusion. To also ease the I/O change, the subroutine through which a particular I/O occurs is listed on the flow chart.

The IN$ and OUT$ locations contain the device number to be used by the I/O subroutines. They are kept pointing to the on-line keyboard by being frequently set to its device number. Those routines that might output through a modified IN$ or OUT$ unit number have all possible exits from these routines to points that reset IN$ and OUT$ to the operator's console. Since these exit points are not always easily identified, because of intricate program interleaving, it is recommended that anyone wishing to change the I/O formats do so by making the I/O subroutines compatible with the program and not vice versa. Also, any required device switching should be done using IN$ and OUT$ as identified in the flow chart.

Problems may arise when devices that need special commands to initialize them (such as, "top of form") are used. Since these functions could be performed manually for I/O devices on the development computer, no reference for performing these tasks is indicated. Since INFORM is an interactive program, it is extremely difficult to predict the operator's intent when issuing each command. Hence, it is much easier to let the operator manually manipulate the I/O devices. If manual manipulation is impossible, such as for magnetic tape, additional commands will be needed to allow the operator to manipulate such devices.

The program uses ASCII characters exclusively. In addition, inputs for which no reference to a CIPHER I/O library subroutine is given are done through the IOIN subprogram, which uses the INPT subroutine. The IOIN subprogram returns the ASCII code and filters out all ASCII codes used for control, such as rubout, line feed, XOFF, etc., by not recognizing them as inputs. If the machine being used recognizes lowercase letters, IOIN should also convert such codes to caps. Thus, these codes are prevented from entering the program. This is an important consideration, since failure to filter these codes may bypass the normal error protection devices and thus cause unpredictable behavior. The INPT subroutine recognizes carriage returns, and forces a line feed if the input device is the operator's console. This is an important feature, necessary to prevent overprinting on the operator's console, because the INFORM software assumes that the operator's console is always ready to receive messages or start a new command line. Commands that are self-terminating supply their own carriage return line feed.

Numerical Conventions

All values used by the program are assumed to be integers unless they are represented by AA, BB, CC, N, SVSF, VOLT, or numbers including a decimal point. The

AA, BB, etc., represent single-precision, floating-point real numbers. No mixed-mode arithmetic is shown, and all places where conversions are required are outlined explicitly. Places where double-precision integer arithmetic is required are also identified. Any integer arithmetic, when shown, should be easily performed on a 16-bit machine without overflows occurring.

The author would like to caution programmers about converting floating-point values to integer values. To maintain the operator's intent and the validity of commands by means of a maximum error recognition capability, it is important to cancel the instruction in progress by a branch to EROR if the floating-point number to be converted is greater than the integer value the machine can handle. A classic example is when an overflow occurs when calculating an address. If the instruction proceeds with an invalid address, unpredictable operation may result. The C$21 routine supplied in most Fortran packages may be used only if it can be modified to return to EROR if this error is detected.

Common Changes

Modifying the command structure. - The filtering of ASCII control and lowercase codes from the input stream yields a particularly efficient method of performing the decoding of the command key character. All INFORM control-edit commands are assumed to start with a name for a named location; hence, the NAME subprogram is executed first. When the return from NAME occurs, it is known that the six-bit truncated ASCII code for the command key character entered must be either 0 or '33 to '77. The zero code is eliminated first by the machine's zero test. Of the codes remaining, if the code is less than '72, '25 is subtracted from it. If it is greater than or equal to '72, '72 is subtracted from it. The codes are thus translated into numbers 0 to '44. Numbers greater than '32 come from ASCII codes for numbers and may be eliminated. Hence, numbers less than '33 may be used as an index for an address array that contains the addresses of the start of the command processing routines. By executing an indexed, indirect-jump instruction on this address array (where the machine performs indexing before the indirect), a jump to the proper command sequence is effected using a minimum amount of the core for the decoding process.

This procedure also makes it easy to add or drop commands by simply changing the address in the array. Key characters that have no corresponding command have the address of EROR in the array. The "#" should not be used for a command because the operator may confuse it with the "#" that terminates a command in progress.

If more key characters are required, it is recommended that individual key characters be expanded into multiple commands by appending numerical character 0, 1, 2, or 3, to the appropriate key character. This will create three commands where there was previously only one. This procedure was done for the : command, which became the four commands :0, :1, :2, and :3. This procedure is recommended because the routine to input and verify the numerals 0, 1, 2, 3 already exists with the program. By executing this routine followed by an indexed, indirect-jump instruction using the number input as an index, two additional commands will be decoded in less code than would be required to properly decode only one additional command using conventional compare and test procedures. This additional decode procedure is added to the start of the already existing command processing sequence (i.e., processor CALC for the : command example).

Once one becomes familiar with the basic functions performed by the subroutines, adding commands to suit one's needs should be easily accomplished in a minimum of steps without having to seriously compromise the ideal command format or function. The possibility of taking portions of existing command processors and converting them to subroutines if portions of their functions are needed for new commands should not be overlooked.

Changing defaults. - The *initial* values for the modifiable defaults DF0 to DF12 are contained in an array called DF0. To permanently change the initial values of these defaults, simply change their initial values in the array. To add a modifiable default, simply add an initial value to the array and place the new length of the array in the DFMT data location contained at the end of the array. Likewise, to *eliminate* a default, shorten the array and change DFMT accordingly. Of course, the commands using the new or deleted defaults will have to be changed to accept or reject said defaults. The % change-default-value command will automatically adjust to the new array length, as it uses CFMT as a guide.

Changing length of storage tables. - A common change to the program will be to shorten or expand the length of the storage tables. Given below are instructions on how to change the length of these tables. When making these changes, be sure to observe any initial values as given in the definitions section.

(1) To change the length of the defined named-location tables and thereby change the maximum number of named locations which may be defined

(a) Set MAXL = X, the new length of these tables

(b) Reserve locations for X values of matrices LOC, SF, NAM1, and NAM2. (Note: SF is floating-point real and requires two locations per value.)

(2) To change the length of data tables 1, 2, or 3, to X, Y, and Z, respectively,

(a) Set the LIM matrix to LIM(1) = X, LIM(2) = Y, and LIM(3) = Z

(b) Reserve X/2, Y/2, and Z/2 locations for TAB1, TAB2, and TAB3, respectively

(c) Reserve the greater of X, Y, or Z locations for STOR.

(3) To change the maximum number of SAMPLE channels to X

  (a) Set NS = X

  (b) Reserve X locations for the TORG and SIDX matrices.

(4) To change the maximum number of DATAO channels to X

  (a) Set ND = X

  (b) Reserve X locations for each of the following matrices: DK1, DEXP, DK2, DK2B, DIDX, DCHN, DUNT.

If the passing of addresses to the PUNCH, LOAD, and VERIFY subroutines was properly programmed as being fully relocatable, changing the length of any of these tables will not affect the operation of commands that produce or load dump tapes. Likewise, all references to these tables should be independent of their length.

It is for this reason, as well as for simplifying the logic that depends on the currently used length of these tables, that all indexing is done in the forward direction (i.e., the algebraic value of $I + 1$ is greater than I) using positive indices and that all loops are terminated by a compare-with-a-stop length. Failure to observe this convention may cause many errors and may actually make the implementation more complex. This is because the portion of these tables in use is constantly varying in length. In addition, most of the program operation depends on the value of the invariant indexing variables. Hence, much of the compare logic and indexing variable computations would need changing. It must be remembered that rolling of the defined named-location tables is not permitted, as this will change the indexing variables and thus alter the fundamental operation of the program.

Core reduction. - A considerable reduction in the program core requirements can be made by eliminating the STOR temporary storage array. Its sole function is to increase the speed at which the values of the named locations are obtained. This is done to minimize data skew among variables that are not quite stationary because of noise. This problem is frequently encountered in on-line control systems where updating of the control calculation must be maintained on a priority basis while the INFORM display is taking place. If this feature is not necessary, the STOR buffer can be eliminated with a corresponding change in the PTBL through EXIT portions of the flow chart.

Some additional core (approximately 80 locations) can be saved by eliminating the default options of the commands. The savings come mainly from the elimination of the DFMT routine, the DF0 array, and a shortening of each command in the appropriate manner. Some care will be required to eliminate this feature, however, and it is not recommended since it greatly improves program operation.

Expansion to incorporate noninteger data types. - The elimination of the speed requirement will also simplify the expansion of the INFORM program to accommodate variables other than those stored as integers. The two most significant bits of the NAM1 location are free to contain a data-type code, say 00 for integer, 10 for double-precision integer, 11 for floating-point, real numbers, etc. By creating a subroutine to obtain the current value of the variable and assuming that all values are floating-point numbers instead of integers, the modifications should reduce to a simple matter of converting all nonaddress arithmetic to that of floating point. Of course, the FIND subroutine would also have to be modified to ignore the first two bits of NAM1 when doing a search of the NAM1 and NAM2 tables.

Expanding error messages. - Some persons may object to the ambiguity of the single ? error message issued whenever a fault is detected. Whenever a branch to EROR occurs, the flow chart lists an appropriate comment that explains the particular error occurring. To expand the message file, simply modify the error routine to accept an argument to indicate which comment should be printed instead of the ?, and create a storage array of comments. The available assembly-language listing also has appropriate comments next to each branch to EROR.

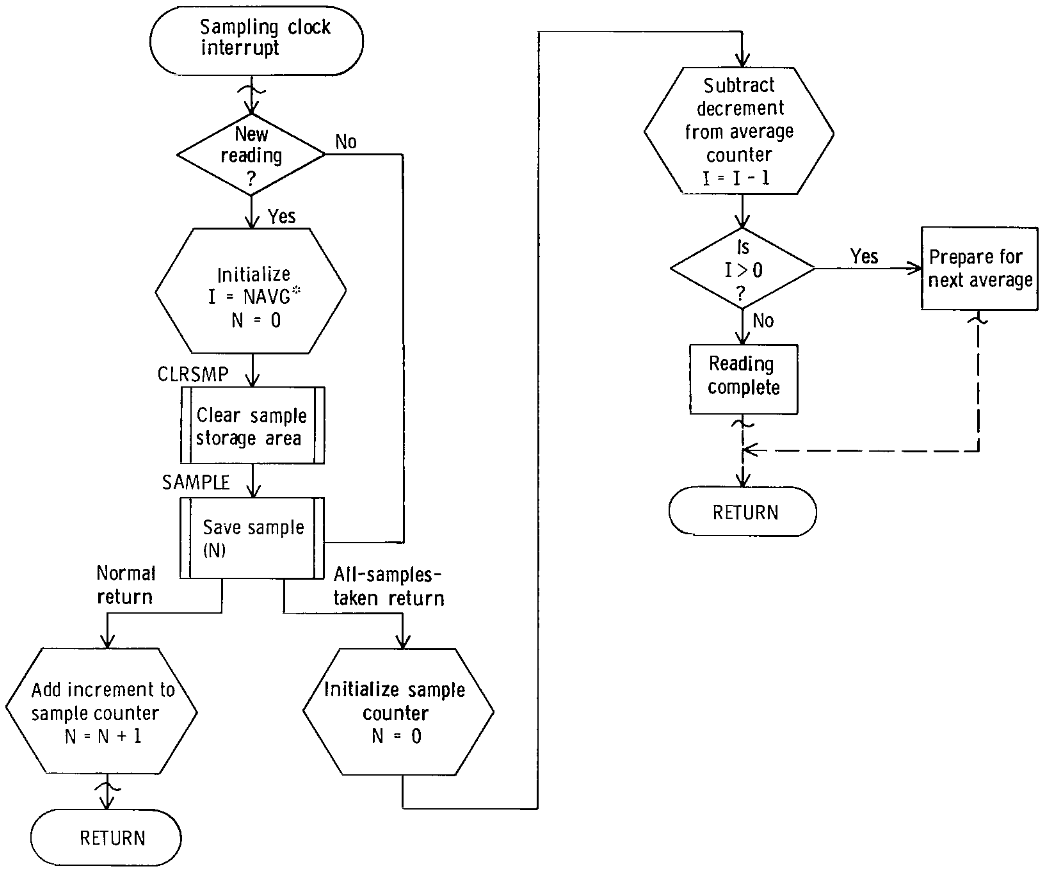
Distinguishing Averaging from Nonaveraging SAMPLE Mode

The number-of-averages (NAVG) parameter entered by the operator is available from INFORM. It is an integer whose value may be obtained by the main calling program through external referencing procedures. It is greater than zero at all times. If SAMPLE is in the averaging mode, NAVG will be greater than 1 and equal to the number of averages. Hence, the calling program can use NAVG as the initial value for a do-loop that increases each time SAMPLE returns to the abort location, indicating that all sample points needed for a single average have been taken. The start of this loop should initiate or restart any required transients and initialize (re-initialize) the sample number argument of SAMPLE to zero. The operator will also require a sense switch or INFORM changeable, logical variable to indicate when to start the first average of a reading. This initialization is done by calling CLRSMP and setting the do-loop variable equal to NAVG. Figure 17 is the flow chart for such a system.

If the main program is set up to handle averaging in this manner, it will automatically work when no averages are being taken, as NAVG will equal 1, indicating that only one pass through the do-loop should occur.

CONCLUSIONS

In this report a versatile program for generating and collecting data for display and information purposes has been described in detail. It has been shown that the software is sufficiently versatile to permit program debugging as well as to permit



*Uses an external reference to the NAVG INFORM variable.

Figure 17. - Flow chart for automatic control of sampling when SAMPLE is operating in either the averaging or nonaveraging mode.

parameter optimization for software development. The fact that these manipulations may occur in a complete interactive or real-time environment increases the program's power. The example problem barely taps the resources of the INFORM software. By using INFORM and becoming more familiar with the commands as outlined in appendices C, D, E, and I, the user should find many opportunities to benefit from it. In fact, users may find that they need never produce a program to perform man-machine communications again.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, March 26, 1979,
505-05.

APPENDIX A

SYMBOLS

A	mnemonic used to indicate single word or integer values held temporarily (usually in an accumulator)
AA	same as A, except represents double-word values such as numbers in floating-point-real format
ACUM	pointer indicating the next available location in the LOC, NAM1, NAM2, EXP and SF tables equal to the number of named locations defined plus one. Initial value: 1 + N, where N is the number of predefined named locations for the hardware registers
ADC	analog-to-digital converter
ADDO	start of command processing routine that adds channel to DATAO or SAMPLE (See flow chart note 6, appendix G, p. 108.)
ADDR	address of defined name being tested for variable-trip-mode triggering
ADD1	start of the "add to data table" command
Ad	octal-or-decimal address entered by operator (See appendix B for acceptable formats.)
ANGLE	dummy name operand used in examples
AT	start of the INFORM clear and open data-table command-processing routine
ASCII	american standard code for information interchange
ASNX	indexing variable to be assigned to the named-location definition being created
AVLH	latch to classify the current value of NAVG; equals -NAVG if and only if NAVG > 1, equals 1 if and only if NAVG $= 1$. If AVLH equals zero, it is an indication to the SAMPLE subroutine to store a value of zero instead of the normal value. Initial value: 1
B	mnemonic used to indicate single word or integer values held temporarily (usually in an accumulator)
BB	same as B, except represents double-word values such as numbers in floating-point-real format

BUF	common core buffer used by NAME subprogram to store the name input in same format used by NAM1 and NAM2 tables. It contains all spaces if the name was defaulted. If the name contains less than three alphanumeric characters, a space is added at the front. Initial value: unspecified
CALC	start of the :1, :2, and :3 INFORM calculate-and-display command processing routines
CD11	address pointer used by the INFORM list core command processing routine. (See flow chart note 4.)
CHIX	equivalent to SIDX or DIDX (See flow chart note 5, appendix G, p. 107.)
CHNG	start of the command processing routine which changes a channel definition of DATAO or SAMPLE (See flow chart note 6, appendix G, p. 108.)
CIPHER	input/output library described in ref. 1
CLOP	flow chart connecting symbol
CMDC	channel incrementer for DATAO and SAMPLE A and C control commands; equals 1 for A commands and 0 for C commands
CMDS	start of the INFORM control command interpretation process
CNDF	number of modifiable defaults contained in the program
CR	ASCII code for carriage return
CRLF	used to designate ASCII codes for carriage return and line feed
CRTB	start of INFORM add name and/or start new line on open data-table command processing routine
CTBL	start of the INFORM close or open data-table command processing routine
DAC	direct-address constant equaling the starting address of a currently referenced data-table sequence array (TAB1, TAB2, or TAB3), with postindexing affixed
Dac	digital-to-analog converter
DACW	location in the DATAO subroutine where a dynamic output accumulator instruction is executed
DATA	flow-chart symbol for portion of PTBL table printout routine where data are printed

DA1	address to be used for the starting location of SAMPLE or DATAO memory dump tapes (See flow chart note 6, appendix G, p. 108.)
DCHN	matrix of digital-to-analog converter numbers on which DATAO outputs occur. Position in matrix corresponds to DATAO channel number. Initial value: zeros or some illegitimate converter number
DEFN	start of INFORM define-a-name or redefine-a-name command processing routine
DEXP	matrix of shift instructions for scaling DATAO outputs. Position in matrix corresponds to DATAO channel number
DFIX ✓	temporary location used to save default indexing variable
DFMT	start of the set-default-forms command processing routine
DFX	array of default values DF0 to DF12
DF0	number of format type to be used by passive-mode data-table printouts when the format specified in the calling program word (FORMAT) is zero
DF1	table number to be used for passive-mode data-table printouts when the data-table number specified in the calling program word (FORMAT) is zero
DF2	unit number to be used for passive-mode data-table printouts when the unit number specified in the calling program word (FORMAT) is zero
DF3	default format type of INFORM print-data-table command
DF4	default data-table number for INFORM print-data-table command
DF5	default output unit number for INFORM print-data-table command
DF6	default starting channel number for INFORM display-or-dump data collected by SAMPLE program command
DF7	default ending channel number for INFORM display-or-dump data collected by SAMPLE program command
DF8	default output unit number for INFORM display-or-dump data collected by SAMPLE program command
DF9	default output unit number for saving definition dumps
DF10	default input unit number for verifying or loading definition dumps
DF11	default output unit number for INFORM list core command
DF12	default output unit number for INFORM display all definitions command

DIDX	matrix of indexing variables to point to named location operand used by each channel of DATAO; order in the matrix corresponds to the DATAO channel number
DISP	start of INFORM command processor to display single INFORM definition
DK1	matrix of multipliers for scaling DATAO outputs. The position in the matrix corresponds to the DATAO channel number
DK2, DK2B	two parallel matrices representing a double-precision integer used for offset voltage calculations on DATAO outputs. The position in the matrix corresponds to the DATAO channel number.
DLM2	location where the command keyword or last operand delimiter is saved
DROP	start of command processing routine that deletes a channel from DATAO or SAMPLE (See flow chart note 5, appendix G, p. 107.)
DTLH	data table latch. A negative number for DTLH indicates that no tables are open; 1, 2, or 3 indicates that that numbered data table is open. Initial value: -2
DUMP	start of INFORM display-or-dump data collected by SAMPLE command processing routine
DUNT	matrix of write-Dac-unit-number instruction on which DATAO Dacs (DCHN matrix) are located. Initial value: zeros or some illegitimate output unit coded instruction
DU1	flow chart connector symbol
EADR	last core address whose contents are listed by the INFORM list core command. Initial value: 0
EQTY	value used to implement dynamic compare logic required for variable-trip-mode operation. Value will usually be 0, 1, or 2, but the meaning may vary depending on the hardware compare instruction of the machine used. For the SEL 810B; 0 means trip if the memory tested is greater than TRIP, 1 means trip if it equals TRIP, and 2 means trip if it is less than TRIP
EQU	start of the INFORM command processor to display current EU values
EROR	start of error processing routine
ERROR	name operand used in example problem; it represents the difference between the commanded and measured SPEEDS.

EU	engineering units
EVT	start of the INFORM end-variable-trip-mode command processing routine
EXIT	program point where operator-set, mode-select flag is tested before returning from INFORM
FMTY	format type used by the INFORM list core command. Initial value: 0
FMT0	flow chart converting symbol directing start of routine to output a data table in format type 0
FORM	start of the INFORM print-data-table command processing routine
FORMAT	argument passed to INFORM to describe the format of the passive mode display. The least significant three bits are the output unit number; the next three are the data table number; and the remaining are the format type. A zero for any of these numbers implies the use of DF0, DF1, or DF2, respectively
I	mnemonic used to indicate indexes and counters
ID	identification
IDXO	number of the SAMPLE or DATAO channel being defined or redefined
INC1	number to be added to the potential default address for the DEFN INFORM command processor to determine the actual default address
INC2	value to be added to ACUM when the named location definition is complete; equals 0 if and only if an old definition is being replaced, 1 if a new definition is in process
INDX	current index used to reference the defined portion of the named-location definition tables LOC, SF, NAM1, and NAM2. INDX is referred to as the "indexing variable." Initial value: 0
INST	machine-generated instruction for variable-trip-mode implementation. It is either a no-operation or stop instruction.
IN$	pseudoregister or common-core location that contains the unit number of the current input device. IN$ is used by the CIPHER I/O library described in ref. 1.
I/O	input or output
J,K	mnemonic used to indicate indexes and counters

K1, K2, K3, K4 K5	constants
LGTH	three-word array containing the current length of data tables 1, 2, and 3. Initial values: LTGH(0) to LGTH(2), 0
LIM	three-location data matrix used to indicate the maximum length the data-table sequence buffers TAB1, TAB2, and TAB3 may achieve. Values: 172, 86, 86
LIST	start of the INFORM list core command processing routine
LOADS	start of the INFORM restore INFORM definitions command processing routine
LOC	matrix that contains the location for the definition of named locations. Initial values: LOC(0) = 0, the rest are arbitrary
LODO	start of the command processing routine that loads channel-definition dump tapes for DATAO or SAMPLE (See flow chart note 6.)
MAX	maximum integer value the computer's arithmetic unit can process; usually equals +32767 (or -32768) for a 16-bit machine
MAXL	location containing the maximum length of the storage matrices for the definitions of the named locations
Max	operand for DATAO control command indicating the displays full-scale EU value
Min	operand for DATAO control commands indicating the display's minimum EU value
N	memory locations where the floating-point-real-number input through the TYR subprogram is saved
Na	name operand (See appendix B for acceptable formats.)
NAM1, NAM2,	parallel tables that contain the five-character name for the definition of named-locations. Initial values: NAM1(0) = '20202, NAM2(0) = '4040 (five spaces for the no-name parameter), and the remainder are arbitrary
NAVG	number of points added together to implement the averaging filter of SAMPLE; must be greater than zero at all times. Initial value: 1
NBUF	maximum number of SAMPLE or DATAO channels that the ADDO command processor will permit

NCH	equivalent to NSCH or NDCH. Used to indicate the size of the SAMPLE or DATAO channel storage tables. It also serves as the address used for the last location of the SAMPLE or DATAO memory dump tapes. (See flow chart notes 6 and 7, appendix G, p. 108.)
NDCH	value in NDCH is one greater than the number of channels currently defined for DATAO. Initial value: 1
NGLH	latch used by subroutines MATH and NANU. If equal to -1, no negative sign preceded the name or number received from the operator
NLTH	name latch set by the NAME subprogram to indicate whether name was before the delimiter. Its value is -1 if and only if a name was entered. All other values imply no name was entered
No	octal-or-decimal number used as operand (See appendix B for acceptable formats.)
NSCH	one greater than the number of channels currently defined for SAMPLE. Initial value: 1
ODAT	start of the DATAO control command interpretation process
OLDX	old indexing variable, that is, the value of the indexing variable used by the last command. It is used by commands that permit a default of the name operand
OLSF	scale factor applied to number displayed by the list core INFORM command. Initial value: 1.0
OTBL	start of the INFORM open-data-table command processing routine
OUT$	pseudoregister or common-core location that contains the number of the current output device. OUT$ is used by the CIPHER I/O library described in ref. 1
PCHO	start of the command processing routine that punches definition dump tapes for DATAO or SAMPLE. (See flow chart note 6, appendix G, p. 108.)
PTBL	start of the routine that prints a data table whose number is in TABN, on the device whose number is in OUT$, in the format contained in TFMT.
QURY	start of INFORM command processing sequence to display all INFORM definitions
QUTE	start of the INFORM scale and store command processing sequence

RTRN	return address for the EROR error processing subroutine, and for command execution sequences. It points to one of three possible program locations: CMDS, SPL, and ODAT. Initial value: the address of CMDS
SADR	first core address whose contents are listed by the INFORM list core command. Initial value: 0
SF	matrix that contains the first word of the single-precision, floating-point-real scale factor of the definition of the named locations. Initial values SF(0) = 0; the rest are arbitrary
Sf	Scale factor
SFAE	sequentially formed arithmetic expression (See appendix B for acceptable formats.)
SFTO	location in DATAO where a dynamic shift instruction is executed
SIDX	matrix of indexing variables to point to the named-location operand used by each channel of SAMPLE. The order in the matrix corresponds to the SAMPLE channel number
SIZE	single value that equals the length of each SAMPLE storage block
SVLH	same as AVLH except that it is never zero. It is used to save the original value of AVLH when CLRSMP sets AVLH to zero. It is also used instead of AVLH whenever it is necessary to protect program operation from zero AVLH values. This is necessary since CLRSMP, DATAO, SAMPLE, and INFORM may be running concurrently on different interrupt levels. Initial value: 1
SOUT	flow chart connector symbol used to indicate exit from the VT routine
SVSF	scale factor obtained from the GSF or GSF2 subroutines floating-point-real value
TABN	data-table number to be used by the PTBL data table printout routine
TAB1, TAB2, TAB3	matrices of indexing variables, which serve as format tables for data tables 1, 2, and 3. A value of '377 denotes a line terminator, and zero that a blank space is to be placed in the data table output
TACH	dummy name operand used in examples
TADR	address to be assigned to named location definition being created
TDKB	temporary location used to save value to go in DK2B matrix until all operator entries are received

TDK1	temporary location used to save DK1 DATAO integer-multiply constant until all operator entries are received
TDK2	temporary location used to save value to go in DK2 matrix constant until all operator entries are received
TEX	temporary location used to save shift instruction to go in DEXP matrix until all operator entries are received
TFMT	format to be used by the PTBL data table printout routine
Tn	data table number operand
TNA	two-location buffer that holds the packed name to be assigned to the named-location definition being created
TORG	matrix of starting addresses of the sample storage block of each SAMPLE channel. The order in the matrix corresponds to the SAMPLE channel number
TRIP	scaled engineering unit value for the defined name being tested against variable-trip-mode requirements. TRIP is compared with the current value of the named location to determine if trip is to occur
TUN	temporary location to save instruction to go into DONT matrix until all operator entries are received
UARW	start of the INFORM delete-data-table-line command processing routine
VCMD	name operand used in examples to represent a commanded output voltage
VOLT	floating-point real constant that equals the number of counts produced by the analog-to-digital converters for a 1.0-volt input. Value: 3200.0 for the SEL 810B system
VT	start of the INFORM enable-variable-trip-mode command processing routine
VTLH	variable-trip-mode latch. A zero sets the latch that indicates that the variable-trip option is to be used when in passive mode. Initial value: Reset or not equal to zero
X	used to represent single-word-variable arguments passed to or from subroutines
XO1	address pointer pointing to either the S1 or O1 subroutines (See flow chart note 6, appendix G, p. 108.)

XX	used to represent two-word-variable arguments passed to or from subroutines
X2	flow chart connector symbol
X3	flow chart connector symbol
x	used to represent an arbitrary ASCII character; may appear strung together, as xxxxx, to represent more than one (in this case, 5) ASCII characters
Y	used to represent single-word-variable arguments passed to or from subroutines
Yn	single ASCII character to indicate operator response; it is "Y" for yes, and "N" for no
YY	used to represent two-word-variable arguments passed to or from subroutines
↲	symbol used to indicate an operator entered carriage-return

APPENDIX B

CONVENTIONS AND DEFINITIONS

Certain definitions and conventions are used when describing the operation of the program. These conventions are listed below.

Values printed by the machine: When it is necessary for an example to indicate values printed by the machine, these values are underlined unless specified otherwise.

Representation of ASCII or keyboard characters: Keyboard entries are shown within quotation marks. Where the character entered is nonprintable or ambiguous, such as "space", its description appears within the quotation marks.

Representation of octal (base eight) numbers: Numbers displayed or entered in octal, or base eight, representation are preceded by an apostrophy. For example, '100 is equivalent to decimal 64.

Machine size: Where references to machine registers are made or where maximum values or limits are given, they are for a 16-bit machine using two's complement arithmetic and having a maximum integer capacity of +32767 and -32768. Single-precision, floating-point numbers are assumed to be 32 bit plus sign with a nine-bit, signed exponent field.

Octal-decimal numbers: An octal-decimal number is any number entered in a format acceptable to the TTYR subroutine of the CIPHER I/O library. Acceptable formats are octal or free-format decimal.

Octal numbers are preceded by an optional plus or minus sign followed by a mandatory apostrophy and must consist of one to six base-eight digits. Numerical length must not exceed 16 binary bits. The number is terminated with a comma or carriage-return.

Decimal numbers may be optionally preceded by a plus or minus sign followed by any number of decimal digits. The decimal point may occur anywhere or be omitted. The absolute value of any entry must not exceed 10^{77}. An exponent to represent either decimal, binary, or unity (scaled fraction) scaling may be entered. Decimal scaling is represented by an "E", "E+", or "E-" entry followed by a maximum of two decimal digits. Binary scaling is represented by a "B", "B+", or "B-" entry followed by a maximum of two decimal digits. Binary scaling is such that B0 places the decimal point before the most significant bit of the machine's integer accumulator with $B + I$ moving the decimal point right and $B - I$ left I places. Binary scaled numbers may not be converted to integers unless X, the decimal numerical portion of the entry is

$$-2^I \le X < 2^I$$

Unity scaling is a special case of binary scaling equivalent to B0. All numbers are entered as fractions and scaled such that 1.0 is equal to the maximum integer value of the machine. To indicate unity scaling, an S follows the fractional entry. No digits are allowed after the S. The absolute value of a unity scaled value must be greater than or equal to -1.0 and less than +1.0.

Numbers may be defaulted (omitted) by simply entering a comma (or carriage return if at end of line). In this event the default value for the command in progress will be used. An erroneous number may be deleted without canceling the whole command line by entering a rubout (ASCII '377) before the comma. The error message E! indicates that the start of a new number is expected. A comma immediately following the error message will result in the use of the default value. Entry of "8", "9", "B", "S", "E", or "." when entering an octal number or entry of an illegal sequence (i.e., two exponents) will have the same effect as entering a rubout. Entry of any character not recognizable by the TTYR subroutine will be treated as a delimiter, thus causing a return to the INFORM software. Since INFORM only accepts commas, carriage returns, or spaces for delimiters, INFORM will cancel the command line and issue the error message ? when such entries occur. Examples for decimal 100.0 are

```
100,              100.0         1.0E2,        +1.0E+2,
+1000E-1,         1000.0E-1     '144,
11"rubout"_E!_    100,          25B13         .00305176S
```

For a -100.0 simply precede all the preceding examples by a minus sign. An entry of '177634 is also equal to -100.0.

Sequentially formed arithmetic expression (SFAE): A sequentially formed arithmetic expression is any string of octal or decimal numbers or defined names separated by +, -, *, or / characters. The sequence terminates with either a comma or carriage return, depending on the command using the SFAE as an operand. The numerical value to be used for a name also depends on the command using the SFAE. For example, when defining an address, the location of the named location as currently defined is used wherever the name appears. Other commands may use the currently defined scale factor or the current EU value of the named location. Appendixes C, D, and E list the default value appropriate for each command.

The SFAE is evaluated as each operation (+ for addition, - for subtraction, * for multiplication, / for division) and number or name is received. Once an operation is entered, it must be followed by a numerical value or name. Failure to do so, or use of an undefined name, causes the typout of ? and cancellation of the command in progress. Assuming the values used for SPEED and TACH to be identical, some examples for the numerical value of 100 are

```
1.0E2      1.0E+2      100,      '144,      '100+'44, 90+10,
50*2,      200/2.,      100.0*3.0/6.0*2.0,      SPEED/TACH*100.,
150+"rubout"_E!_-50,
```

Note: 50-50"rubout"_E!_+ 50, is incorrect since the rubout only cancels the octal or decimal number 50 and not the minus sign, which is a numerical operator of the SFAE indicating subtraction. The correct procedure would be to enter: 50--50, as it is interpreted as 50-(-50), where the second minus sign is part of the octal or decimal number 50.

Octal or decimal address: An octal or decimal address may consist of any sequentially formed arithmetic expression. If a name occurs within the expression, the location as currently defined for the named location is used as the numerical value. The comma or carriage return that terminates the SFAE also terminates the address entry. The only restriction on the SFAE is that the calculated value must be a legitimate core location. Warning: Since addresses are integers, intermediate values formed when producing the SFAE that exceed 24 binary bits or that produce fractions which cannot be represented exactly by a 32-bit floating-point quantity may result in calculated addresses that are incorrect due to truncation when the final value is converted to an integer. For example, 13/3*3 yields an address of 12 and not 13! Usually, one is safe if only addition, subtraction, and multiplication by small integers are used when entering addresses.

This flexibility in defining the address allows the rapid determination of addresses that are located within arrays or relocatable subprograms, or both, and frees the operator from tedious base-eight or mixed-base calculations. By defining named locations whose address is the relocation base of subprograms, the addresses listed by the assembler may be used directly by simply inserting the defined name before the comma or carriage return terminating the SFAE. Examples for a decimal 100.0 address are given below. The named location "Na" is assumed to have a defined address of decimal 50.

```
50+Na,        Na+50.,       Na+6"rubout"_E!_50,
Na+Na,        Na+'62,       Na+100.-50., Na*2,
```

Scale factor (Sf): A scale factor (Sf) is the number to be used to multiply by a machine-stored number to produce an engineering unit value. When being entered by the operator, it may consist of any sequentially formed arithmetic expression. If a defined name occurs within the expression, the scale factor as currently defined for the named location is used as the numerical value within the expression. The comma or carriage return that terminates the SFAE also terminates the scale-factor entry. A scale factor can be either positive or negative. A scale factor of zero is interpreted as being equal to 1.0, but that the named location represents an octal number rather than an EU quantity. For more information see the Defining named locations section (p. 16).

NAME: A NAME consists of one to five characters for a named location. The first character must be one of the 26 possible alphabetic characters. Since truncated ASCII is used, all letters must be capitals. The remaining four characters are optional and may be any of the 36 possible alphanumeric characters.

APPENDIX C

INFORM COMMANDS

The general structure and use of the commands listed below is described in the section Use of Commands (p. 15). Unless the command has a prefix operand, the key character should be the first character on the command line. For commands with a prefix operand, the operand should precede the key character, since failure to do so will automatically substitute the default value for the prefix operand. The start of a command line is indicated by the machine output of an underscore and a bell.

Any operand that appears within braces may be defaulted if the default value as listed for that operand is desired. Operands DF0 to DF12 may have their default values changed through use of the % command. Machine messages that occur as part of a command sequence are underlined and should not be entered by the operator. Although not shown, the operator may terminate formats with either a comma or a carriage return. If a carriage return occurs before all operands are entered, the default value for the unentered operands is used. If an unentered operand has no default value, an error will occur. Some commands that have prefix operands (such as =) need no terminator, as they are terminated by the machine. A table containing a summary of all the commands is given in appendix I.

Mode Control Commands

Enable variable trip mode

Key character	Operands
[	{Na}x{SFAE} <u>HLT?_</u>Yn

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.) Default value: The last used name operand of previous commands.

x: One of three possible characters (">", "=", or "<") to indicate that the trip condition will be met when Na either is greater than, equal to, or less than the SFAE.

SFAE: A sequentially formed arithmetic expression for an EU value for Na at which trip will occur. If a name appears within the SFAE, the current EU value for that name is used as the numerical value within the SFAE. Default value: 0.0.

Yn: One of two possible characters - "Y" to indicate the desirability of a machine halt when the trip condition is met, or "N" to decline the halt option.

Description: This command is used to set the variable-trip-mode latch (VTLH) and define the trip condition. Note: A mode change will not occur on executing this command, but the path of the passive mode will be altered (see section INFORM Mode Selection).

Programmer's flow-chart reference: VT.

Reenable previous variable trip mode

Key character	Operands
[	None

Description: Envoking this command is the same as reentering the previous enable variable-trip-mode command as described above. If no previous enable variable-trip-mode command was issued, an error will occur. Variable-trip-mode conditions are not saved on relocatable binary dump tapes; hence, loading such tapes will not affect operation of this command.

Programmer's flow-chart reference: VT.

Disable variable trip mode

Key character	Operands
]	None

Description: This command resets the variable-trip-mode latch (VTLH). No operands are required.

Programmer's flow-chart reference: EVT.

Accept DATAO control commands

Key character	Operands
>	None

Description: This command is used to change the input command string from INFORM control-edit commands to DATAO control commands.

Programmer's flow-chart reference: ODAT.

Accept SAMPLE control commands

Key character	Operands
<	None

Description: This command is used to change the input command string from INFORM control-edit commands to SAMPLE control commands.

Programmer's flow-chart reference: SPL.

Exit INFORM subroutine

Key character	Operands
	None

Data Table Manipulation Commands

Clear and open data table

Key character	Operands
@	{Tn}

Tn: A single-digit table number (1, 2, or 3). Default value: Data table 1.

Description: This command clears data table number Tn and opens the table for entries to build a new table.

Programmer's flow-chart reference: AT.

Open data table

Key character	Operands
(	{Tn}

Tn: A single-digit table number (1, 2, or 3).

Description: This command opens table number Tn.

Programmer's flow-chart reference: OTBL.

Close an open data table

Key character	Operands
)	None

Description: This command closes any open data table.
Programmer's flow-chart reference: CTBL.

Add name to open table

Prefix operand	Key character	Additional operands
{Na}	,	None

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.) Default value: A blank space equal to the length of one variable printout is added to an open table. If no table is open, see "Description" below.

Description: This command adds the named location Na to the rear of an open table. If no table is open, no action is taken, but the name referenced becomes the default value of the name operand for later commands that use the last-referenced name operand as their default value. If no name operand was entered, the default value remains unchanged. After the entry of this command, the machine prints a two-digit decimal number, indicating the number of entries since the last line terminator (i.e., the number of names on the current line).

Programmer's flow-chart reference: ADD1.

Add name and/or start new line on open data table

Prefix operand	Key character	Additional operands
{Na}	-	None

Na: A one- to five-character name for named location. (See appendix B for acceptable formats.) Default value: The line is terminated without adding a name to the table.

Description: Variable Na, if supplied, and a line terminator are added to an open table. This will end the previous line of a currently open table and start a new

blank line. Following the entry of this command, the machine will print a two-digit decimal number, 00, indicating the start of the new line. If no table is open, an error message is issued, and no action is taken.
Programmer's flow-chart reference: CRTB.

Delete last line from open table

Key character	Operands
↑	None

Description: This command is used to delete the last line of an open data table up to but not including the last line terminator in the table. If a line terminator was the last table entry, that line terminator as well as the entire previous line up to but not including the line immediately before the terminator is deleted from the table. If no table is open, an error message is issued, and no action is taken.
Programmer's flow-chart reference: UARW.

Print data table

Key character	Operands
\	DF3 , DF4

DF3: A format number (0, 1, 2, or 3). Default value: Format type 1.
DF4: A data table number (1, 2, or 3). Default value: Data table 1.
DF5: An octal or decimal number for an output device. Default value: Line printer (unit 5).

Description: This command is used to print the prestored data-table-sequence number, DF4 (data table DF4) in data-table-format number DF3 on I/O-device DF5, while in the interactive mode. Any combination of DF3, DF4, and DF5 is permitted, but care must be exercised to insure that data table DF4 has line terminators occurring sufficiently often to prevent truncation of a line by the I/O device. If data table DF4 is of zero length (i.e., undefined), an error message is issued, and no action is taken. Examples of data formats 1, 2, and 3 are given in figure 7 (p. 20). Data-table-format 0 is designed for computer to computer communication. It is identical to a $2 SAMPLE-block-dump format for only one data block. The number of variables in the data table becomes the block-length number in this dump format. See the $2 command description below for a detailed description of the $2 dump format.
Programmer's flow-chart reference: FORM.

Miscellaneous Display Commands

Display current EU value

Prefix operand	Key character	Additional operands
{Na}	=	None

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.) Default value: The last-used name operand of previous commands.

Description: This command is used to display the current EU value for the named location Na.

Programmer's flow-chart reference: EQU

Calculate and display address

Key character	Operands
:0	SFAE

SFAE: A sequentially formed arithmetic expression for an address. If a name appears within the SFAE, the address as currently defined for that name is used as the numerical value within the SFAE.

Description: The octal representation of the value calculated by the SFAE is displayed on the operator's console.

Programmer's flow-chart reference: CALC.

Calculate and display scale factor

Key character	Operands
:1	SFAE

SFAE: A sequentially formed arithmetic expression for a scale factor. If a name appears within the SFAE, the scale factor as currently defined for that name is used as the numerical value with the SFAE.

Description: The value calculated by the SFAE is displayed on the operator's console.

Programmer's flow-chart reference: CALC.

Calculate and display engineering units (EU)

Key character	Operands
:2	SFAE

SFAE: A sequentially formed arithmetic expression for an EU value. If a name appears within the SFAE, the current EU value as located and descaled according to the name's definition is used as the numerical value within the SFAE.

Description: The value calculated by the SFAE is displayed on the operator's console.

Programmer's flow-chart reference: CALC.

Calculate and display octal words of floating-point EU value

Key character	Operands
:3	SFAE

SFAE: A sequentially formed arithmetic expression for an EU value. If a name appears within the SFAE, the current EU value as located and descaled according to the name's definition is used as the numerical value within the SFAE.

Description: This command is identical to the :2 command, except that the display consists of two octal values instead of the floating-point number.

Programmer's flow-chart reference: CALC.

List core

Key character	Operands
/	{Ad1},{Ad2},{x},{Sf},{DF11}

Ad1: An SFAE for the starting octal or decimal address. (See appendix B for acceptable formats.) Default value: Last nondefaulted Ad1 used by this command. Initial value: 0.

Ad2: An SFAE for the ending octal or decimal address. Default value: Last nondefaulted Ad2 used by this command, if and only if Ad1 was defaulted. Ad2 defaults to Ad1 if Ad1 was entered. Initial value: 0.

x: Data or display type. This ASCII character should be "0" for an octal display of integer data: "1" for a decimal display of integer data: "2" for a floating-point display of single-precision, floating-point real data; "4" for a decimal display of scaled-fraction data; "6" for a floating-point display of double-precision integer data; "7" for a decimal integer display of double-precision integer data; and "8" for an ASCII dump of core, the core being interpreted as containing 2 ASCII characters per word. Default value: Last nondefaulted x used by this command. Initial value: 0.

Sf: An SFAE for an octal or decimal scale factor to be applied to the data. See appendix B for acceptable formats. Default value: Last nondefaulted Sf used by this command. The initial value assumed is 1.0. Note: This value is ignored for display formats 0, 7, and 8.

DF11: Octal or decimal number for an output device. Default value: The operator's console (unit 1).

Description: The core starting at location Ad1 and up to and including Ad2, is displayed on the device selected by DF11. The value for x determines how the data are to be interpreted (integer, floating-point real, scaled fraction, or ASCII) and what kind of display is to be produced. Before the display, the data are multiplied by Sf. Sf is ignored for octal, double-precision integers, and ASCH data.

Programmer's flow-chart reference: LIST.

Data Storage Commands

Scale and store integer

Prefix operand	Key character	Additional operands
{Na}	"	{SFAE}

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.) Default value: The last-used name operand of previous commands.

SFAE: A sequentially formed arithmetic expression for an EU value to be stored. If a name appears within the SFAE, the current EU value, as located and descaled from its definition, is used as the numerical value within the SFAE. Default value: 0.0.

Description: The EU value as calculated by the SFAE is scaled and stored as an integer using the scale factor and location of the definition for Na. The octal representation of the location for Na, which is to be modified, is displayed before acceptance of the SFAE. The message issued is "LOCATION WAS = xxxxxx ".

After this message the operator should enter the SFAE. After the SFAE, the machine will print the octal representation of the value just stored. The message is "NOW = xxxxxx".

Programmer's flow-chart reference: QUTE.

Commands for Definition Control

Define a named location

Prefix operand	Key character	Additional operands
{Na}	"space"	{Ad},{Sf}

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.) Default value: the no-name parameter.

Ad: An SFAE for an octal or decimal address. (See appendix B for acceptable formats.) Default value: One greater than the address for that of the last-used name operand of previous commands. If Ad is defaulted, the machine will display the default value before accepting the Sf entry. (See note below.)

Sf: An SFAE for an octal or decimal scale factor. (See appendix B for acceptable formats.) Default value: The same scale factor as that of the last-used name operand of previous commands.

Note: If Na is defaulted, the no-name parameter becomes the default name operand for following commands. For example,

```
_"space"**1,0
        --
_=_'000000
 --------
```

The "=" command displays the no-name parameter (i.e., core location 1) using a scale factor of 0. Continuing,

```
_SPEED"space"**,_'000002__ ,
             --  ----------
```

Since we defaulted both Ad and Sf for the space command, a value of 1 + 1, or 2, became the default location (as demonstrated by the machine display of '000002). The default Sf became 0, since the definition for the no-name parameter we just defined was used to determine the Ad and Sf values.

Description: This command is used when it is desired to define a named location. The default option for Na, or the no-name parameter, is especially useful when making program patches or doing random searches during debugging. It allows the operator to manipulate the core with all the power associated with named variables, but without cluttering the defined-name list with temporary definitions. The no-name variable is overwritten each time this command is envoked, but the ensuing definition is used to determine the default values of subsequent commands; that is, the no-name parameter becomes the last-used name operand for these commands until a predefined name is used. The use of this command to make program patches is illustrated by the following sequence used to patch machine instructions in locations '400 through '405.

```
_"space"**'400,0
_" '000400 WAS = '000000 '22, NOW = '000022
_"space"**, '000401 ,
_" '000401 WAS = '000001 '032404, NOW = '032404
_"space"**, '000402 ,
_" '000402 WAS = '111010 '10405, NOW = '010405
_"space"**, '000403 ,
_"space"**, '000404 ,
_= '007776
_"space"**, '000405 1/32768
_" '000405 WAS = '123456 0.5, NOW = '040000
```

Programmer's flow-chart reference: DEFN.

Redefine a named location

Prefix operand	Key character	Additional operands
{Na}	"space"	{xxxxx},{Ad},{Sf}

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.)

xxxxx: A one- to five-character name for a named location to replace name Na. It (i.e., xxxxx) may be the same as Na but otherwise must be unique. (See appendix B for acceptable formats.) Default value: The name used for Na (i.e., the name stays the same).

Ad: An SFAE for an octal or decimal address. (See appendix B for acceptable formats.) Default value: The address currently defined for Na (i.e., the address remains unchanged).

Sf: An SFAE for an octal or decimal scale factor. (See appendix B for acceptable formats.) Default value: The scale factor currently defined for Na (i.e., Sf will remain unchanged).

Description: This command is used to redefine a named location. All occurrences of variable Na in any tables will now be replaced with variable xxxxx, and the new location and scale factor, as defined, will be used. The only exception to this rule is when Na occurs in a DATAO table. Here, the new name and location will be used, but because additional values are derived from the scale factor when building DATAO tables, the old scaling for Na will be applied to xxxxx in DATAO outputs. Hence, the DATAO table entry must be re-entered. (See appendix D for commands to revise the DATAO tables.)

Programmer's flow-chart reference: DEFN(OVLY).

Change a default value

Key character	Operands
%	DFn,{No}

DFn: Integer number of the default value to be changed, that is, "0" for DF0, "1" for DF1, etc.

No: Positive octal or decimal number of the value to which the default is to be changed. Default value: The current value remains unchanged and is displayed on the operator's console.

Description: Default values for commands that are given numbers such as DF1, DF2, . . ., DF12 can be changed using this command. INFORM definition dump tapes will automatically contain the modified defaults. All default values must be positive integers. The machine will display the current status of any default value if No is defaulted.

Programmer's flow-chart reference: DFMT.

Display all INFORM definitions

Key character	Operands
?	{DF12}

DF12: Output device number. Default value: Line printer (unit 5).

Description: The location and scale factor for each named location currently defined is displayed on output device DF12. Since users may define their scale factors as ratios, the single numerical value of the ratio may be of little concern; hence, an EU value corresponding to the maximum value a single integer can contain and the EU value corresponding to a 1-volt Dac output are also displayed. Each definition is numbered so that the operator can determine how many definitions remain before the name learing tables are filled.

Programmer's flow-chart reference: QURY.

Display a single INFORM definition

Key character	Operands
&	{Na}

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.) Default value: The no-name parameter.

Description: The location and scale factor as currently defined for named-location Na is displayed on the operator's console. An Eu value corresponding to the maximum value a single integer can contain and the EU value corresponding to a 1-volt Dac output are also displayed. The indexing variable for the name is also displayed preceding these numbers.

Programmer's flow-chart reference: DISP.

Save INFORM definitions

Key character	Operands

!	{DF9}
	VERIFY? Yn,{DF10}

DF9: The unit number of the output device on which the dump is to occur. Note: Either a comma or carriage return may follow DF9. Default value: Paper tape punch (unit 2).

Yn: A single ASCII character, "Y" or "N", to indicate if the dump just produced is to be verified.

DF10: The unit number of the input device from which the dump is to be read. DF10 should only be entered if Yn is the ASCII "Y". Default value: Paper tape reader (unit 2).

Description: All the defined named-location definitions, the three prestored data-table sequences, and the default forms are output on device DF9 in a relocatable binary-dump format. This dump may be saved and used to restore these operator-defined values at a later date by using the * (restore definitions) command. The VERIFY? query occurs after completion of the dump. If it is desired to check the dump for bit errors, the dump should first be mounted onto device DF10, then an ASCII "Y" should be entered for the Yn mnemonic, followed by a value for DF10. The dump will then be read and checked against the values in core. If and only if an error occurs, CK is printed on the operator's input device, and the command is terminated before the entire dump is read. If no verification is desired, an ASCII "N" should be entered for the Yn mnemonic, and no DF10 should be entered. This option is especially useful when INFORM is used for programs that have their named locations defined as being in the common core. The program may be updated and reloaded without having to redefine the INFORM names. As long as the common locations remain fixed, the dump tape may be used to teach a new load the old definitions.

Programmer's flow-chart reference: PNCH.

Restore INFORM definitions

Key character	Operands
*	{DF10}

DF10: The unit number of the input device from which the definition dump is to be read. Default value: Paper tape reader (unit 2).

Description: A dump produced by the ! (save definitions) command is read from device DF10. The values read from this dump replace those values and become the currently defined values. As a result of executing this command, all tables are closed and the variable-trip mode is disabled. If a read error occurs during the load process, CK is printed on the operator's input device, all table sequences, definitions, and default values are cleared to their initial values, and the command is terminated before the remainder of the dump is read. Since all program intelligence is lost if a bad dump is read, it is good practice to verify all dumps before using them.

Note: If, after a fresh load of INFORM, a dump is produced before any definitions are made, this dump will clear the program memory whenever loaded. Loading this dump will allow the operator to start with a fresh set of definitions whenever desired.

Programmer's flow-chart reference: LOADS.

Display/Dump Data Collected by SAMPLE

Output SAMPLE channels in format type 1

Key character	Operands
$1	{DF6},{DF7},{DF8}

DF6: An octal or decimal number representing the first channel data block to be output. Default value: zero.

DF7: An octal or decimal number representing the last channel data block to be output. Default value: 80. Note: If DF7 exceeds the number of channels currently defined, only as many channels as are defined will be dumped.

DF8: Number of the output device on which the data from the SAMPLE subroutine channels is to be output. Default value: Floppy disk (unit 3).

Description: The data gathered by the SAMPLE subroutine for channels DF6 to DF7 is output on device DF8. If DF7 exceeds the number of channels currently defined, the last channel number defined is used for DF7. The data are taken from the storage blocks reserved for the SAMPLE channels. The entire block length, as currently defined, is output for each channel. If averages were taken, only the single-precision value of the average is output. The data for each channel are output in ASCII characters representing the decimal equivalent of the scaled binary data. Each line of output is terminated with the carriage return, line feed, and "XOFF" (ASCII '223) characters. This format was established for automated reading of the output into a larger time-sharing computer system using a Fortran program. A typical $1 dump is shown in figure 18.

The data are grouped in blocks of lines, one block for each SAMPLE channel. Preceding all of these blocks are three lines: The first line contains the number of data points within each block of data that is to follow. This number occurs twice in the Fortran format of 2I6. The second line of output is a number derived by multiplying the first channel number by 100 and adding the last channel number. This value occurs in the Fortran format of I6. The third line duplicates this number in the Fortran format of I8. The data blocks follow these three lines. The format for each line within a channel's data block is as follows:

First line: The channel number occurs twice in the Fortran format of 2I6.

Second line: The scale factor of the named-location sampled for this channel appears. The scale factor is in the Fortran format of I6, I3, where the I3 value represents a power of 10 to be multiplied times the I6 value.

Third line: A checksum in the Fortran format of I7 is given. This value equals the algebraic sum of the I6 and I3 values of the second line above.

```
 00100 00100
 00003
 0000003
 00000 00000
 45776-05
 045771
-00172-00198-00222-00244-00264-00282-00300-00316-00332-00340-00348-00356-00366-00368-00368-00374-00374-00374-005610
-00372-00366-00366-00360-00356-00348-00338-01700-01666-01602-01522-01442-01362-01278-01196-01114-01034-00956-017378
-00878-00800-00728-00656-00586-00522-00456-00396-00336-00278-00222-00172-00120-00072-00030 00014 00052 00092-006094
 00122 00156 00182 00214 00240 00262 00282 00300 00314 00330 00342 00352 00364 00370 00372 00380 00380 00380 005342
 00380 00380 00376 00376 00370 00370 00364 00358 01712 01678 01614 01538 01460 01374 01292 01210 01128 01046 017026
 00970 00892 00816 00742 00674 00602 00534 00466 00410 00350
 006456
 00001 00001
 45776-05
 045771
 02166 02192 02218 02240 02262 02282 02300 02312 02324 02336 02344 02352 02360 02362 02366 02368 02368 02368 041520
 02366 02364 02360 02354 02350 02344 02336 02328 02294 02228 02150 02068 01990 01904 01822 01740 01660 01582 038240
 01504 01428 01354 01284 01214 01146 01082 01022 00962 00904 00848 00794 00746 00700 00656 00612 00574 00534 017364
 00506 00472 00442 00414 00386 00366 00346 00326 00310 00294 00284 00274 00264 00256 00252 00248 00246 00244 005930
 00244 00242 00248 00250 00254 00256 00262 00270 00278 00312 00378 00454 00534 00618 00702 00784 00866 00948 007900
 01024 01102 01178 01252 01322 01392 01460 01526 01584 01644
 013484
 00002 00002
 24414-08
 024406
 02324 02299 02271 02240 02207 02171 02132 02092 02050 02007 01963 01918 01872 01826 01780 01733 01686 01639 036210
 01592 01546 01500 01455 01410 01366 01323 01110 00901 00700 00509 00328 00157-00003-00153-00293-00423-00543 012482
-00653-00753-00844-00926-01000-01066-01123-01173-01215-01250-01278-01300-01315-01324-01328-01327-01321-01310-020506
-01295-01276-01254-01228-01198-01166-01131-01094-01055-01014-00972-00928-00883-00837-00791-00744-00697-00650-018213
-00603-00556-00509-00462-00416-00370-00325-00281-00067 00142 00343 00535 00717 00888 01049 01200 01341 01471 004097
 01592 01703 01805 01897 01981 02056 02122 02180 02231 02274
 019841
 00003 00003
 30518-08
 030510
 04717 04527 04340 04155 03976 03770 03586 03446 03259 03116 02971 02823 02662 02542 02431 02289 02176 02063 058843
 01960 01880 01769 01691 01603 01538 01485-06857-06168-06331-06389-06423-06434-06398-06348-06274-06186-06084-056946
-05958-05808-05666-05503-05330-05169-04976-04796-04596-04390-04178-03980-03756-03538-03338-03115-02911-02684-079692
-02498-02283-02100-01878-01676-01489-01305-01126-00962-00784-00623-00468-00300-00159-00039 00114 00227 00340-017009
 00452 00565 00658 00771 00851 00962 01040 01115 08399 08730 08893 08973 09020 09001 08977 08929 08858 08760 094954
 08670 08547 08411 08262 08124 07944 07762 07561 07404 07207
 079892
```

Figure 18. - Display of data collected by subroutine SAMPLE. Display was produced using INFORM $/ command. System of example problem was used to obtain data.

Fourth line: The fourth through the next to the last line of the data block contain the scaled data. Each line consists of 18 data points plus a checksum. The Fortran format is 18I6, I7. The checksum in format I7 represents the algebraic sum of the 18 I6 values.

Next to last line of block: If less than 18 values remain to complete the number of points in the block, only as many I6 values as are needed are present, and no checksum appears on this line.

Last line of block: This line is the checksum in the Fortran format of I7 for the short line produced above. If the number of points in the block is an integer multiple of 18, no short line was produced above, hence the checksum appears on the line with the data and this line is omitted.

Note: EU values are obtained by multiplying the data that start on the third line by the scale factor derived from the first line.

Programmer's flow-chart reference: DUMP.

Output SAMPLE channels in format type 2

Key character	Operands
$2	{DF6},{DF7},{DF8}

DF6: An octal or decimal number representing the first channel data block to be output. Default value: zero.

DF7: An octal or decimal number representing the last channel to be output. Default value: 80. Note: If DF7 exceeds the number of channels currently defined, only as many channels as are defined will be dumped.

DF8: Number of the output device on which the data from the SAMPLE subroutine channels are to be output. Default value: Floppy disk (unit 3).

Description: This command is similar to the $1 command except that the data are descaled by multiplying by the scale factor for the named location associated with the SAMPLE-channel number. The product is a floating-point number. A typical $2 dump is shown in figure 19. The format for each line within a channel's data block differs from that for a $1 dump. It is as follows:

First line: The first through next to the last line of the data block contains the scaled data. Each line consists of 12 data points plus a checksum. The Fortran format is 12(I6,I3),I8. The I3 value represents a power of 10 to which the I6 value should be raised to obtain the EU data value; that is, -100.0 would appear as -10000-02. The checksum in format I8 represents the algebraic sum of the 12 I6 values plus the 12 I3 values.

```
 [illegible]
 000063
 0000001
-[illegible]-14009-02-14466-02-15199-02-15661-02-15931-02-16297-02-0307471
-16[illegible]-02-16[illegible]-02-[illegible]-02-17[illegible]-02-16755-02-16480-02-16297-02-15931-02-0201084
-15473-02-[illegible]-02-[illegible]-02-[illegible]-02-69673-02-66011-02-62348-02-58583-02-54750-02-50996-02-47334-02-43763-02-0696295
-40193-02-[illegible]-02-[illegible]-02-[illegible]-02-26876-02-23896-02-20870-02-18178-02-15382-02-12727-02-10163-02-78736-03-0346929
-54953-03-[illegible]-03-1[illegible]-03 64067-04 23804-03 42114-03 55847-03 71411-03 83313-03 97961-03 10986-02 11993-02 0359854
 12909-02 1[illegible]-02 14574-02 15166-02 15656-02 16113-02 16663-02 16927-02 17029-02 17395-02 17395-02 17395-02 0190681
 17395-02 17395-02 17212-02 17212-02 16937-02 16937-02 16663-02 16388-02 78369-02 76813-02 73883-02 70404-02 0435584
 66833-02 62897-02 59141-02 55[illegible]-02 51636-02 47887-02 44403-02 40[illegible]-02 37354-02 33966-02 30853-02 27557-02 0558722
 24445-02 21332-02 18766-02 [illegible]-02 99152-02 10034-01 10115-01 10204-01 10355-01 10446-01 10529-01 10583-01 0252056
 10638-01 10693-01 10[illegible]-01 10[illegible]-01 10803-01 10812-01 10831-01 10840-01 10840-01 10840-01 10831-01 10822-01 0129435
 10803-01 10776-01 10[illegible]-01 10[illegible]-01 10693-01 10657-01 10601-01 10199-01 98419-02 94666-02 91095-02 87158-02 0456438
 83405-02 79611-02 [illegible]-02 72418-02 68848-02 65369-02 61981-02 58777-02 55573-02 52460-02 49530-02 46783-02 0770760
 44037-02 41382-02 38818-02 36346-02 34149-02 32043-02 30025-02 28015-02 26276-02 24445-02 23163-02 21606-02 0380285
 20033-02 18951-02 17670-02 16704-02 15839-02 14923-02 14191-02 13458-02 13000-02 12543-02 12085-02 11719-02 0181342
 11536-02 11353-02 11261-02 11169-02 11169-02 11078-02 11353-02 11444-02 11627-02 11719-02 11993-02 12360-02 0138038
 12726-02 14282-02 17305-02 20[illegible]-02 24445-02 28290-02 32131-02 35889-02 39642-02 43396-02 46875-02 50446-02 0366187
 53925-02 57317-02 60016-02 63[illegible]-02 66853-02 69855-02 72510-02 75256-02 56738-05 56128-05 55444-05 54688-05 0742890
 53882-05 53003-05 52[illegible]-05 51074-05 50049-05 48939-05 47925-05 46826-05 45703-05 44[illegible]-05 43457-05 42310-05 0579799
 41162-05 40015-05 38867-05 37744-05 36621-05 35522-05 34424-05 33[illegible]-05 32300-05 27100-05 21997-05 17090-05 0396132
 12427-05 80078-06 38330-06-73243-08-37350-06-71534-06-10328-05-13258-05-15943-05-18380-05-20606-05-22608-05-0152492
-24415-05-26026-05-27418-05-28639-05-29664-05-30519-05-31202-05-31739-05-32105-05-32325-05-32423-05-32398-05-0358933
-32252-05-31983-05-31617-05-31153-05-30616-05-29981-05-29249-05-28468-05-27613-05-26710-05-25758-05-24757-05-0350217
-23731-05-22657-05-21559-05-20436-05-19313-05-18165-05-17018-05-15870-05-14723-05-13575-05-12428-05-11280-05-0210815
-10157-05-90033-06-79347-06-68605-06-16358-06 34668-06 83740-06 13062-05 17505-05 21680-05 25610-05 29297-05-0039304
 32739-05 35913-05 38867-05 41577-05 44067-05 46313-05 48364-05 50195-05 51807-05 53223-05 54468-05 55518-05 0552991
 14395-04 13815-04 13245-04 12680-04 12134-04 11505-04 10944-04 10498-04 99457-05 95093-05 90668-05 86151-05 0470533
 81238-05 77576-05 74188-05 69855-05 66406-05 62958-05 59814-05 57373-05 53986-05 51605-05 48920-05 46936-05 0750795
 45319-05-17814-04-18824-04-19322-04-19499-04-19602-04-19636-04-19526-04-19274-04-19148-04-18879-04-18568-04-0164922
-18183-04-17726-04-17292-04-16795-04-16267-04-15776-04-15187-04-14637-04-14027-04-13398-04-12751-04-12147-04-0184234
-11463-04-10798-04-10188-04-95063-05-88838-05-81910-05-76234-05-69673-05-64088-05-57313-05-51148-05-45442-05-0662215
-39826-05-34364-05-29359-05-23927-05-19013-05-14283-05-91504-06-48524-06-11903-06 34790-06 69275-06 10376-05-0198377
 13794-05 17242-05 20081-05 23529-05 25970-05 29358-05 31738-05 34027-05 25632-04 26642-04 27139-04 27383-04 0302479
 27527-04 27469-04 27396-04 27249-04 27032-04 26733-04 26459-04 26083-04 25668-04 25214-04 24792-04 24243-04 0315817
 23688-04 23074-04 22595-04 21994-04
 0091335
```

Figure 19. - Display of data collected by subroutine SAMPLE. Display was produced using INFORM $2 command. System of example problem was used to collect data.

Next to last line of block: If less than 12 values remain to complete the number of points in the block, only as many I6/I3 pairs as are needed are present, and no checksum appears on this line.

Last line of block: This line is the checksum in the Fortran format of I8 for the short line produced above. If the number of points in the block is an integer multiple of 12, no short line was produced above; hence, the checksum appears on the line with the data, and this line is omitted.

Programmer's flow-chart reference: DUMP.

APPENDIX D

COMMANDS TO CONTROL THE DATAO SUBROUTINE

Displays from the DATAO subroutine are referenced by channel number. The 24 channels available are labeled 0 to 23. Assigned to each channel is a single named location, two EU values representing zero and full-scale outputs, an output-unit number, and a Dac channel number for the output unit. These values for each channel must be defined by the operator before a display of that channel will occur. Only unit and channel numbers corresponding to system Dacs are accepted. Channel numbers must be defined sequentially, but once defined may be modified, but not deleted, at any time. Only the highest number channel may be deleted.

The DATAO control command processor is entered by using the > INFORM control-edit command. The computer returns a > sign instead of the underscore when the DATAO control command processor is in effect. The DATAO control commands are listed in this appendix. The underscore DATAO command will return control to the INFORM control-edit command processor.

Add a new channel definition

Key character	Operands
A	{Na},SFAE1,SFAE2,{Dn},{Dnn}

Na: A one- to five-character name for a named location. See appendix B for acceptable formats. Default value: The last-used name operand of previous commands.

SFAE1: An SFAE for an EU value to represent the 0-volt output of the display. If a name occurs within the SFAE, the current EU value as located and descaled from the named-location definition is used as the numerical value within the SFAE.

SFAE2: An SFAE for an EU value to represent the 10-volt output of the display. If a name occurs within the SFAE, the current EU value, as located and descaled from the named location definition, is used as the numerical value within the SFAE.

Dn: An octal or decimal unit number for one of the Dac units. Default value: The last value of Dn used for this DATAO channel. Channels that have not been previously defined since program load will generate an error if Dn is defaulted. The DATAO @ command does not destroy this default value.

Dnn: An octal or decimal Dac number of unit Dn. Default value: The last value of Dnn used for this DATAO channel. Channels that have not been previously defined since program load will generate an error if Dnn is defaulted. The DATAO @ command does not destroy this default value.

Description: The next sequential DATAO channel number supplied by the machine is defined to display the value of Na on a linear scale, outputing 0 volt when Na = SFAE1 and 10 volts when Na = SFAE2. The channel number being defined is displayed by the machine immediately after the delimiter that separates the "A" keyword from the operands. To protect recording equipment, the program limits the output to the -0.1 and 10.24 volt levels. No restrictions exist on the relationship of SFAE1 to SFAE2; that is, SFAE1 may be less than, greater than, or equal to SFAE2. SFAE1 and SFAE2 may also be negative to produce reversed displays. (For further information, see DATAO Displays section (p. 21).)

Programmer's flow-chart reference: ADDO, 01.

Change a channel definition

Key character	Operands
C	xx,{Na},SFAE1,SFAE2,{Dn},{Dnn}

xx: The number of the DATAO channel to be redefined.

Na: A one- to five-character name for a named location. See appendix B for acceptable formats. Default value: The named location currently defined for channel xx.

SFAE1: An SFAE for an EU value to represent the 0-volt output of the display. If a name occurs within the SFAE, the current EU value, as located and descaled from the named-location definition, is used as the numerical value within the SFAE.

SFAE2: An SFAE for an EU value to represent the 10-volt output of the display. If a name occurs within the SFAE, the current EU value, as located and descaled from the named-location definition, is used as the numerical value within the SFAE.

Dn: An octal or decimal unit number for one of the Dac units. Default value: Unit number as currently defined for channel xx.

Dnn: An octal or decimal Dac number of unit Dn. Default value: Dac number as currently defined for channel xx.

Description: This command is similar to the A command, except that the channel being redefined must already have been defined using the A command.

Programmer's flow-chart reference: CHNG.

Delete the last channel definition

Key character	Operands
D	None

Description: The last sequentially defined channel definition is deleted. Note: This command may be used any number of times in succession, as an automatic stop occurs when no channel definitions remain.
Programmer's flow-chart reference: DROP.

Delete all channel definitions

Key character	Operands
@	None

Description: All channel definitions are deleted.
Programmer's flow-chart reference: ATO.

Display all channel definitions

Key character	Operands
?	{DF12}

DF12: Output device number. Default value: Line printer (unit 5).
Description: The definitions for all the currently defined channels are displayed on device DF12. The order of the values on each line of output is the channel number, the output unit number in octal, the Dac channel number, the name of the named location being displayed, the EU value for 0-volt output, and the EU value for a 10-volt output.
Programmer's flow-chart reference: ODAT+6.

Save channel definitions

Key character	Operands

!	{DF9}
	VERIFY? Yn,{DF10}

DF9: The unit number of the output device on which the dump is to occur. Note: Either a comma or carriage return may follow DF9. Default value: Paper tape punch (unit 2).

Yn: A single ASCII character, "Y" or "N", to indicate whether the dump just produced is to be verified.

DF10: The unit number of the input device from which the dump is to be read. DF10 should only be entered if Yn is the ASCII character "Y". Default value: Paper tape reader (unit 2).

Description: All the channel definitions are output on device DF9 in a relocatable binary-dump format. This dump may be saved and used to restore these operator-defined values at a later date by using the * (restore channel definitions) command. The VERIFY? query occurs after completion of the dump. If it is desired to check the dump for bit errors, the dump should first be mounted onto device DF10, then an ASCII "Y" should be entered for the Yn mnemonic, followed by a value for DF10. The dump will then be read and checked against the values in core. If and only if an error occurs, CK is printed on the operator's input device, and the command is terminated before the entire dump is read. If no verification is desired, an ASCII "N" should be entered for the Yn mnemonic and no DF10 should be entered.

Programmer's flow-chart reference: PCHO.

Restore channel definitions

Key character	Operands
*	{DF10}

DF10: Input unit number. Default value: Paper tape reader (unit 2).

Description: A dump produced by the DATAO ! (save channel definitions) command is read from device DF10. The channel definitions, as read from this dump, replace those channels as currently defined. If a read error occurs during the load process, CK is printed on the operator's input device, all definitions and default values are cleared to their initial values, and the command is terminated before the remainder of the dump is read.

Programmer's flow-chart reference: LODO.

Accept INFORM commands

Key character	Operands
-	None

Description: This command is used to change the input command string back to INFORM control-edit commands.

Programmer's flow-chart reference: CMDS.

APPENDIX E

COMMANDS TO CONTROL SUBROUTINE SAMPLE

Variables sampled by subroutine SAMPLE are assigned channel numbers 0 to 79. Also assigned to each channel is a starting location for the core storage block to be used to save the samples. The length of the storage block is the same for all channels. This length must be set the first time the SAMPLE-control command processor is entered after a program load. If averaging is being used, that is, if the number of averages specified is two or more, the defined block length must be at least two times the number of samples taken. The block length may be changed at any time by using the B command. The number of points to be averaged to determine a sample must also be defined at this time. If this number equals zero or one, averaging is not be used.

The SAMPLE-control command processor is entered by using the < INFORM control-edit command. The computer returns a < sign instead of the underscore when the SAMPLE control command processor is in effect. The SAMPLE control commands are listed in this appendix. The underscore SAMPLE command will return control to the INFORM control-edit command processor.

Add a new channel definition

Key character	Operands
A	{Na},{Ad}

Na: A one- to five-character name for a named location. (See appendix B for acceptable formats.) Default value: The last-used name operand of previous commands.

Ad: An octal or decimal address for the start of the storage block to be used to save variable Na's history. (See appendix B for acceptable formats.) Default value: Address of the highest numbered channel defined plus the current block size. An error will occur if Ad is defaulted when no SAMPLE channels have been defined.

Description: The next sequential SAMPLE channel number, as supplied by the machine, is defined to store variable Na's history. The number of the channel being defined is displayed immediately after the delimiter separating the "A" key character from its operands. The storage space to be used starts at address Ad. The next available storage location, which will not be used by this channel, is displayed after the command line. This value will only be applicable for the currently defined storage block length and averaging mode. For further details see the SAMPLE Subroutine Control section.

Programmer's flow-chart reference: ADDO,S1.

Change a channel definition

Key character	Operands
C	xx,{Na},{Ad}

xx: The number of the SAMPLE channel to be redefined.

Na: A one- to five-character name for a named location. See appendix B for acceptable formats. Default value: The named location currently defined for channel xx.

Ad: An octal or decimal address for the start of the storage block to be used to save Na's history. Default value: The address currently assigned to channel xx.

Description: This command is similar to the A command preceding except that the channel being redefined must already have been defined using the A command.

Programmer's flow-chart reference: CHNG.

Delete the last channel definition

Key character	Operands
D	None

Description: The last sequentially defined channel definition is deleted. Note: This command may be used any number of times in succession, as an automatic stop occurs when no channel definitions remain.

Programmer's flow-chart reference: DROP.

Delete all channel definitions

Key character	Operands
@	None

Description: All channel definitions are deleted.

Programmer's flow-chart reference: ATO.

Display channel definitions

Key character	Operands
?	{DF12}

DF12: Output device number. Default value: Line printer (unit 5).

Description: The definitions for all the currently defined channels are displayed on device DF12. The order of the values on each line of output is the channel number, the output-unit number in octal, the Dac channel number, the name of the named location being displayed, the EU value for 0-volt output, and the EU value for a 10-volt output.

Programmer's flow-chart reference: ODAT+6.

Define block length

Key character	Operands

B	BLOCK SIZE={No}
	# AVERAGES= {Noo}

No: An octal or decimal number to define the new storage block length. Note: No should be terminated with a carriage return.

Noo: A positive octal or decimal number indicating the number of readings being averaged to determine a sample. It is displayed when the ? command is issued. Note: Noo should be terminated with a carriage return.

Description: The storage block length for all channels is immediately changed to No, and Noo becomes the assumed number of averages being taken for each sample point (the actual number is controlled by the program calling SAMPLE). When the data are output using the $1 or $2 commands, the data stored in the storage block are assumed to be the summations of Noo readings at each sample point. This assumption is made regardless of how the main program calls SAMPLE or any previous condition of No and Noo in effect when the data were taken. If Noo is greater than one, the data are treated as double-precision integers, and are divided by Noo to obtain a single-precision integer value before output. Therefore, if Noo is greater than one, No must be twice the number of sample points; otherwise, No should equal the number of samples to be taken.

Warning: In general, No and Noo should not be changed until after any data collected are output using the $1 and $2 commands. Otherwise, the output may be incorrect. It is permissible, however, to change No and Noo at any time provided the operator realizes these consequences. (See the SAMPLE calling sequence for additional information, p. 12.)

Warning: The starting location of each storage block is not changed. Hence, increasing No from its present value may result in overlapping blocks unless sufficient space was originally allocated for each block. The program has no provision to determine whether storage bounds may be exceeded, and it is up to the operator to insure that each block maintains its integrity. If Noo is increased from one to greater than one, the number of samples fitting in a storage block is automatically cut in half. The main program need not be altered if it uses the abort-return location to determine when all samples are in, since the return to this location will automatically occur sooner.

Programmer's flow-chart reference: SZE.

Save channel definitions

Key character	Operands

!	{DF9}
	VERIFY? Yn, {DF10}

DF9: The unit number of the output device on which the dump is to occur. Note: either a comma or carriage return may follow DF9. Default value: Paper tape punch (unit 2).

Yn: A single ASCII character, "Y" or "N", to indicate whether the dump just produced is to be verified.

DF10: The unit number of the input device from which the dump is to be read. DF10 should only be entered if Yn is the ASCII character "Y". Default value: Paper tape reader (unit 2).

Description: All the channel definitions are output on device DF9 in a relocatable binary-dump format. This dump may be saved and used to restore these operator-defined values at a later date by using the * (restore channel definitions) command. The VERIFY? query occurs after completion of the dump. If it is desired to check the dump for bit errors, the dump should first be mounted onto device DF10, then an ASCII "Y" should be entered for the Yn mnemonic, followed by a value for DF10. The dump will then be read and checked against the values in core. If and only if an error occurs, CK is printed on the operator's input device, and the command is terminated before the entire dump is read. If no verification is desired, an ASCII "N" should be entered for the Yn mnemonic and no DF10 should be entered.

Programmer's flow-chart reference: PCHO.

Restore channel definitions

Key character	Operands
*	{DF10}

DF10: Input unit number. Default value: Paper tape reader (unit 2).

Description: A dump produced by the SAMPLE ! (save channel definitions) command is read from device DF10. The channel definitions, as read from this dump, replace all existing channel definitions. If a read error occurs during the load process, CK is printed on the operator's input device, all definitions and default values are cleared to their initial values, and the command is terminated before the remainder of the dump is read.

Programmer's flow-chart reference: LODO.

Accept INFORM commands

Key character	Operands
-	None

Description: This command is used to change the input command string back to INFORM control-edit commands.

Programmer's flow-chart reference: CMDS.

APPENDIX F

DESCRIPTION OF REQUIRED SUPPORTIVE SOFTWARE

Subroutines required:

Standard Fortran: L$22, H$22, M$22, D$22, A$22, S$22, N$22, C$12, C$21, C$42, C$24

CIPHER input-output library: MESAGE, PUNCH, LOAD, VERIFY, INPT, AIP, AOP, OUT00, OUT11, OUT22, TTYR, FDDH, FDD11, FDD22, FDDEND

Since complete descriptions of the functions of all the routines listed here are contained in appendix H, only an overview of the routines' application is given here.

In the program descriptions given in this appendix DAC, as assembly language pseudo-operation, is used to represent a direct address constant produced by the program loader. The value stored in the reserved location is equal to the address of the parameter named in the DAC statement. In addition to the machine's hardware A, B, and X registers, pseudoregisters are defined in map or page zero memory as follows:

Location	Description	Register
0	Common core	V
1	Common core	W
2	First floating-point accumulator	AA
3	Second floating-point accumulator	AA
4	Third floating-point accumulator	BB
5	Fourth floating-point accumulator	BB
6	Return address register	R$TN
7	Output-device number	OUT$
8	Input-device number	IN$

Many arguments passed between subroutines are held in these registers for transfer.

Standard Fortran Routines

L$22

Calling sequence:

```
CALL L$22
DAC operand
```

H$22

Calling sequence:

```
CALL H$22
DAC storage
```

M$22

Calling sequence:

```
Set AA = multiplicand
CALL M$22
DAC  multiplier
AA = product
```

D$22

Calling sequence:

```
Set AA = dividend
CALL D$22
DAC  divisor
AA = quotient
```

A$22

Calling sequence:

```
Set AA = addend
CALL A$22
DAC  addend
AA = sum
```

S$22

Calling sequence:

```
Set AA = minuend
DAC  subtrahend
AA = difference
```

N$22

Calling sequence:

```
Set AA = value to be negated
CALL N$22
AA = -AA
```

C$12

Calling sequence:

```
Set A = the integer value to be converted
CALL C$12
AA = A
```

C$21

Calling sequence:

Set AA = floating-point value to be converted
CALL C$21
A = the integer of AA

C$42

Calling sequence:

Set (A, B) = double-precision integer
CALL C$42
AA = (A, B)

C$24

Calling sequence:

Set AA = floating-point value to be converted
CALL C$24
(A, B) = the double-precision integer of AA

CIPHER Library Input-Output Routines

MESAGE

Calling sequence:

```
        Set OUT$ = the output-device number
        CALL MESAGE
    (X) Data, ASCII characters, stored 2 per word
        Data, 0 - to signify end of message.  Note: These data are not needed if
            the last eight bits of last data statement above are zero
```

Description: MESAGE provides the assembly-language programmer a means of printing table headings or messages with all the ease and convenience of Fortran format statements, but with a vastly reduced core requirement. Device flexibility is achieved through the use of the AOP subroutine. In addition, since the message to be printed appears as data statements representing the eight-bit ASCII code following the CALL statement, a convenient comment in the form of the actual message is automatically inserted in the assembly listing. This message aids in identifying parts of the program.

Subroutines required: AOP

PUNCH

Calling sequence:

```
    (W) Set A = start address
    (X) Set B = last address punched
    (Y) Set X = output-device number
        CALL PUNCH
```

(Z) Data, number of spaces in leader

Subroutines required: Set AOPX, LOAD, VERIFY

LOAD

Calling sequence:

(W) Set A = start address
(X) Set B = end address
(Y) Set X = input-device number
CALL LOAD
Error-in-reading return point
No-error, or normal-return point

Subroutines required: Set AIPX, AOP, VERIFY, MESAGE

VERIFY

Calling sequence:

(W) Set A = start address
(X) Set B = end address
(Y) Set X = input-device number
CALL VERIFY
Error-in-verification return point
No-error or normal-return point

Subroutines required: Set AIPX, LOAD, MESAGE

Description for PUNCH, LOAD, and VERIFY: These routines are a packaged set that generates dumps designed to be read by a person or by Fortran-coded programs.

AIP

Calling sequence:

Set IN$ = input-device number
CALL AIP
A = 16 bits from the IN$ device

AIPR

Calling sequence:

Set IN$ = input-device number
CALL AIPR
A = A + 8 bits from device X

AOP

Calling sequence:

OUT$ = output-device number
Set A = 16 bits for the OUT$ device
CALL AOP

AIPX

Calling sequence:

Set X = input-device number
CALL AIPX
A = 16 bits from device X

AOPX

Calling sequence:

Set X = output-device number
Set A = 16 bits for device X
CALL AOPX

AIPRX

Calling sequence:

Set X = input-device number
CALL AIPRX
A = A + 8 bits from device X

INPT

Calling sequence:

Set IN$ = input-device number
CALL INPT
A = 8 bits from the IN$ device

Description for AIP, AOP, AIPX, AOPX, AIPRX, and INPT: These surboutines are designed to provide a moderate level of device-free I/O capability to the assembly-language programmer. By using this software package, the assembly-language programmer frees himself of the highly specialized and machine-dependent nature of many I/O functions. By having at least two channels of device-free I/O, one immediately gains the flexibility of software and ease of programming normally only encountered on large computing systems, such as the IBM TSS 360/67 systems. The device flexibility that INFORM enjoys is directly attributable to use of these routines.

OUT00

Calling sequence:

(X) Set A = an octal value to be printed
CALL OUT00
Set OUT$ = output-device number
DAC, number of least-significant digits printed

Subroutines required: Set AOP

OUT10: See OUT00

OUT11

Calling sequence:

Set OUT$ = output-device number
Set A = an integer to be printed
CALL OUT11
DAC number of significant digits printed

Subroutines required: OUT44

OUT22

Calling sequence:

Set OUT$ = output-device number
(X) Set AA = floating-point real value printed
CALL OUT22
(Y) DAC format word. The mantissa length is the least-significant six bits of the format word. The exponent length is the next three bits. The remainder of the format word is used to indicate the position of the decimal point and whether it is printed.

Subroutines required: OUT44

OUT12

Calling sequence:

Set OUT$ = output-device number
(X) Set A = integer to be printed
CALL OUT12
(Y) DAC format word (See OUT22 format word.)

Subroutines required: OUT22, C$12

OUT21

Calling sequence:

Set OUT$ = output-device number
(ZZ) Set AA = floating-point real value printed
CALL OUT21
(Y) DAC number of significant digits printed

Subroutines required: C$21, OUT44

OUT44

Calling sequence:

Set OUT$ = output-device number
(ZZ) (A, B) = double-precision integer printed
CALL OUT44
(Y) DAC number of significant digits printed

Subroutines required: AOP

Description of the OUTxx routines: This portion of the CIPHER I/O package is intended to give the assembly-language programmer a moderate amount of format flexibility when numbers need to be displayed in a man-machine interface situation. By developing this set of highly efficient, general purpose routines, the generally annoying and time consuming task of writing code to print numbers in assembly language is eliminated. Although they do not offer the complete flexibility available in Format format statements (10-digit mantissa maximum, no leading zero suppression) they do offer the same ease of programming, without requiring the vast amount of core associated with format-decoding routines. The inclusion of a double-precision integer format and the exclusion of a double-precision, floating-point format should more closely tailor these routines to assembly-language requirements than Fortran format decoders.

The output is sufficiently flexible that it should meet a wide range of aesthetic as well as practical needs. The exclusion of the decimal point and exponent sign from the floating-point format may offend the sensibilities a bit, but this was done to meet the requirements of the FDDxx series of subroutines and thereby simplify them considerably. In addition, a small increase in output speed for slow I/O devices is achieved. If these considerations are not requirements, the decimal point and exponent sign can easily be inserted.

TTYR

Calling sequence:

Set IN$ = input-device number
CALL TTYR
(Y) DAC floating-point storage
(ZZ) AA = number input
(W) X = delimiter character causing return
(Z) B = type of entry information: B > 0 indicates decimal entry; B < 0 indicates octal entry; B = 0 indicates no entry (AA = 0)

Subroutines required: AIP, MESAGE

TTYR2

Calling sequence:

Set IN$ = input-device number
(X) Set A = the first input character
CALL TTYR2
(Y) DAC floating-point storage
(ZZ) AA = number input
(W) X = delimiter character causing return
(Z) B = type of entry information: B > 0 indicates decimal entry; B < 0 indicates octal entry; B = 0 indicates no entry (AA = 0)

Subroutines required: TTYR

Description: TTYR is a versatile routine that finds application whenever the man-machine interface deals with single-precision integers or floating-point numerical values entered by man. Its free-format style of number entry has already been amply described by the description of octal-or-decimal numbers in appendix B. The program can return information to the calling program describing the type-of-number entry, the delimiter that caused branch back, and the value intended. This ability, when coupled with the TTYR2 entry point, which accepts a previously input character, yields virtually limitless use of the program. Input device independency is achieved through use of the AIP subroutine.

FDDH

Calling sequence:

Set OUT$ = output-device number

(W) Set A = identification number

(X) Set B = number of digits for A output

CALL FDDH

(Y) Data, number of data words per line

(Z) Data, coded integer. Least significant six bits, the number of digits in mantissa of the length of integers excluding sign; next three bits, the length of exponent less sign; remaining bits, number of digits in checksum excluding sign.

Subroutines required: OUT11, all the FDDxx type subroutines

FDDEND

Calling sequence:

Set OUT$ = output-device number

CALL FDDEND

Subroutines required: MESAGE, all the FDDxx type subroutines

FDD11

Calling sequence:

Set OUT$ = output-device number

(X) Set A = integer to be output

CALL FDD11

Subroutines required: OUT11, MESAGE

FDD21

Calling sequence:

Set OUT$ = output-device number

(ZZ) Set AA = floating-point value to be output

Subroutines required: OUT21, MESAGE

FDD22

Calling sequence:

```
             Set OUT$ = output-device number
       (ZZ)  Set AA = floating-point value to be output
             CALL FDD22
```

Subroutines required: OUT22, MESAGE

FDD12

Calling sequence:

```
             Set OUT$ = output-device number
        (X)  Set A = integer to be output
             CALL FDD12
```

Subroutines required: FDD22, C$12

Description of the FDDxx subroutines: These subroutines are intended as a general-purpose software package to produce dumps of data that will undergo further processing by being read into another computer. All characters are produced using the ASCII code, and the format of the numbers is that of the OUTxx routines, which is easily interpreted using standard-Fortran integer-read statements. Each line of output is terminated with carriage return, line feed, and XOFF characters. The XOFF character is provided for automatic control of hardware that can read dumps into a time-share system. Persons who wish to use this postprocessing capability should tailor the line terminator characters to their systems.

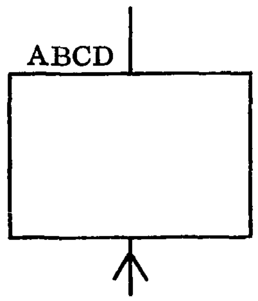
APPENDIX G

FLOW CHARTS

Flow Chart Symbols

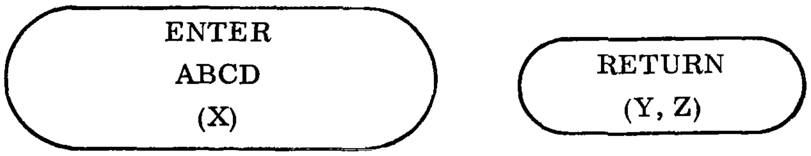
Symbols used in the flow charts are taken from ANSI Standard X3.5-1970 (ref. 2) except as noted below.

Names of subroutines are located above and to the left of the block:



Refers to subroutine ABCD

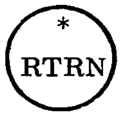
Arguments that are passed to and from subroutines are indicated in parenthesis by the enter and exit symbols; that is,



Indicates that X is received from the calling program and Y, Z are returned. Note: This notation does not refer to arguments held in common core.

The following conventions are used to indicate various addressing modes.

ABCD	Refers to the integer, floating point, etc., contents of memory location(s) ABCD.
[ABCD]	Indicates that memory location ABCD contains the address of the memory cell or cells which contain the value desired.
ABCD(I)	Refers to the I th value in array ABCD.

(A, B)	Is used when it is required to show how two integer words are concatenated to form a double-precision integer, floating-point real, or other two-word value.
-----	Encloses a group of processing functions designed to accomplish a specific task.
	When a connector symbol contains an asterisk, the flow proceeds to the point whose address is in RTRN; that is, RTRN contains a word which points to the proper flow point.

Flow Chart Notes

(1) The actual method used to perform the packing is immaterial, since the NAME subprogram is the only place where packing is done. The PNAM routine, which is used to print a name, is the only other routine in addition to NAME which needs to be modified if packing different from that described below is used. If we label a 16-bit word such that the most significant bit is 1 and the least significant bit is 16, then the method used in the SEL 810B is as follows:

Bits(3 - 8) of BUF(0) go into bits(11 - 16) of BUF(0)
Bits(9 - 14) of BUF(0) go into bits(11 - 16) of BUF(1)
Bits(15 and 16) of BUF(0) go into bits(11 - 12) of BUF(2)
Bits(1 and 4) of BUF(1) go into bits(13-16) of BUF(2)
Bits(5 - 10) of BUF(1) go into bits(11 - 16) of BUF(3)
Bits(11 - 16) of BUF(1) go into bits(11 - 16) of BUF(4)

(2) A particularly efficient means of implementing this block is to do an indexed branch using the truncated ASCII delimiter code passed back from the NAME subroutine as the index. For further details consult the PROGRAMMING NOTES section.

(3) Recall that each element of the TAB1, TAB2, and TAB3 data tables for which this routine is used is only 8 bits long; hence, 2 elements are stored in one 16-bit word. Even indices are stored in the most significant bits, odd the least.

(4) The LI0, LI1, LI2, LI4, LI6, LI7, LI8 program branch points use the same loop (LI15) to format the table output, add an increment to the core address pointer B, and determine when all values have been printed. The CD11 is used to modify the return point of LI15 so that different interpretations to the core values and different printouts can be implemented efficiently. Use of CD11 is thus similar to use of RTRN.

(5) The LOC, NAM1, NAM2, SF , TAB1, TAB2, TAB3, LGTH, and DF0 tables, along with ACUM and CFMT, occur sequentially in core. Therefore, the values of all

these parameters will be saved on the relocatable binary dump produced by the PU subprogram.

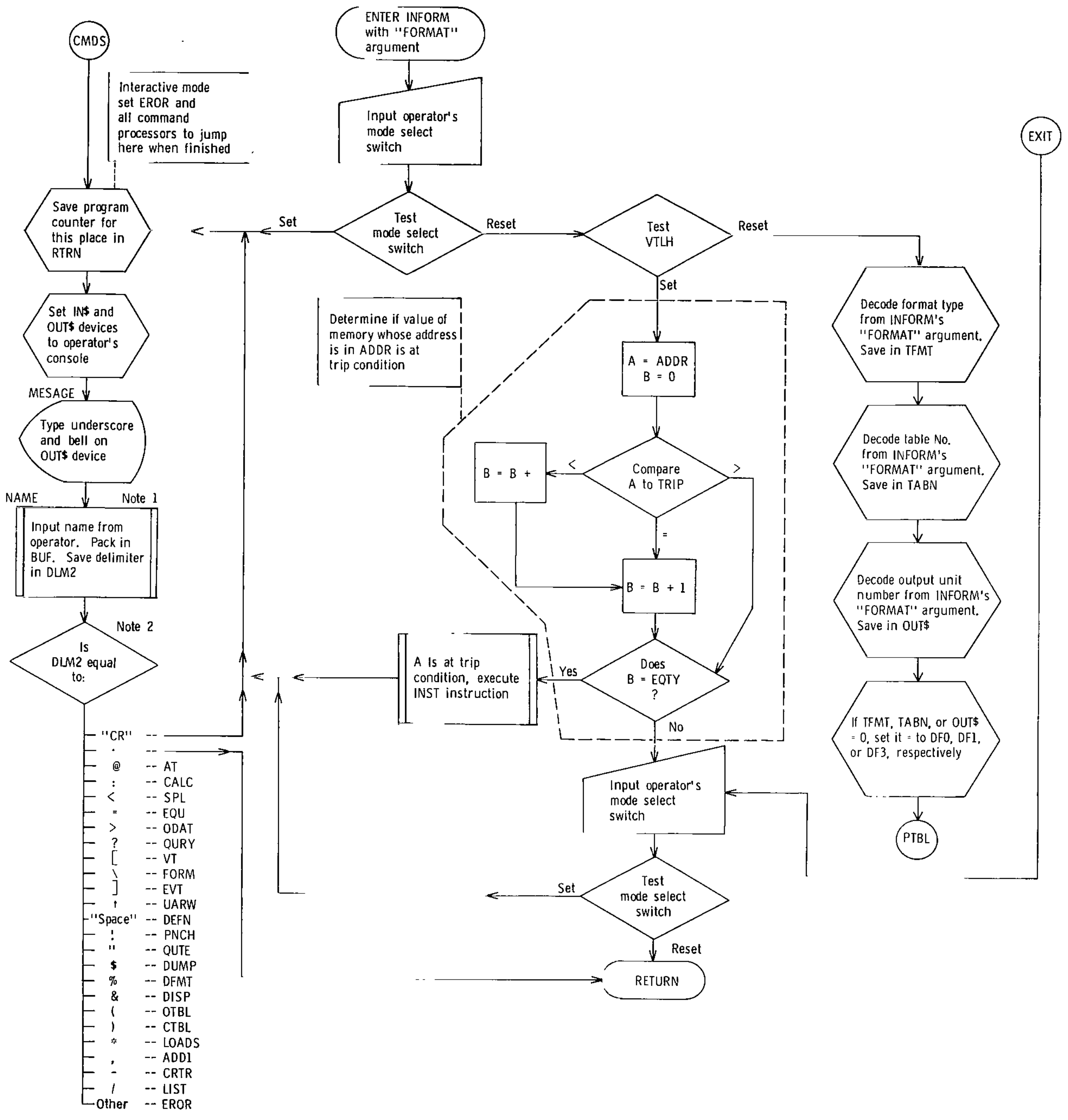
(6) XO1, DA1, NCH, and CHIX contain the addresses of S1, TORG, NSCH, and SIDX (with postindexing affixed) or the addresses of O1, DK1, NDCH, and DIDX (with postindexing affixed). The values depend on whether the command processing routines (ATO, ADDO, CHNG, DROP, LODO, PCHO) are initialized to process SAMPLE or DATAO control commands. All references to XO1, DA1, NCH, and CHIX are done indirectly, thus effectively changing the operand for the above mentioned routines. Also see the DATAO and SAMPLE command processor section under program structure.

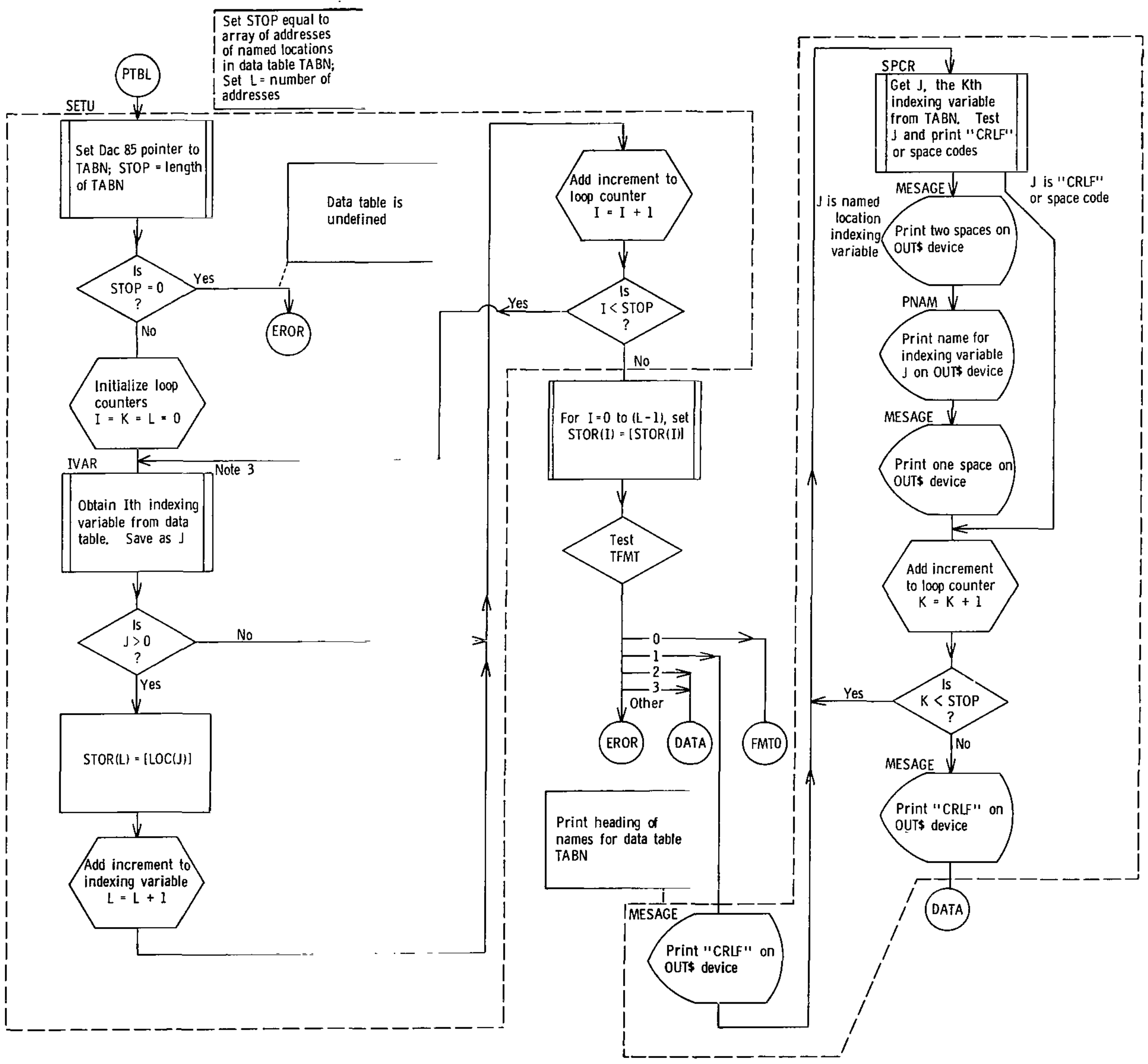
(7) The DK1, DEXP, DK2, DK2B, DIDX, DCHN, and DDAC tables, along with location NDCH, occur sequentially in core. The TORG and SIDX tables, along with locations NAVG, AVLH, SVLH, SIZE, and NSCH, also occur sequentially in core. Therefore, the values of the table set referenced (see note 6) will be saved on the relocatable binary-dump tape produced by the PU subprogram. It is extremely important that the tables and locations mentioned occur in the specified order so that an error occurring when reloading dumps will not adversely affect machine operation; that is, to prevent SAMPLE from erroneously storing on top of program instructions or DATAO from producing extraneous output to random devices.

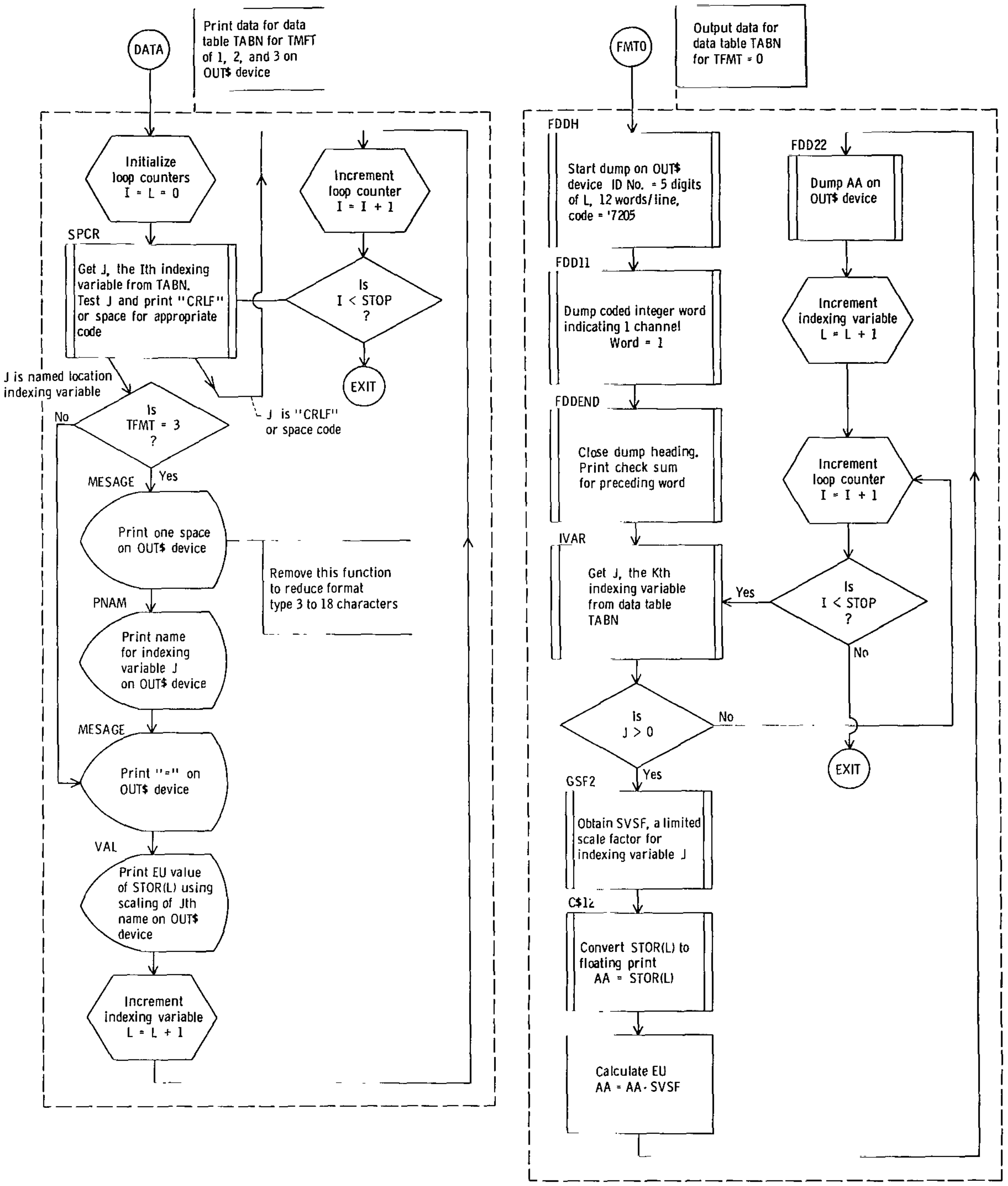
(8) The DATAO subroutine achieves rapid output capability by executing machine-coded instructions generated in the O1 subroutine. These instructions are intended to reduce the required $Y = Ax + B$ type calculation to one of integer arithmetic that can be executed at high speed. This high-speed capability is necessary since DATAO is designed to display the dynamic behavior of the named locations selected.

(9) This test, which may involve several instructions, is actually performed in the GSF subroutine, which passes an argument back to indicate the result of the test.

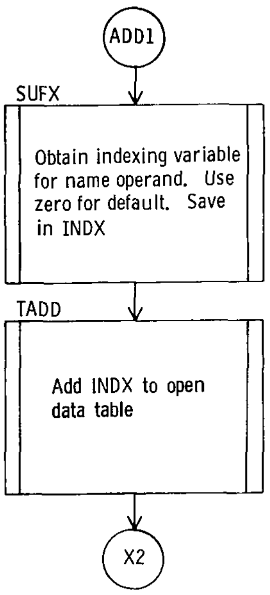
(10) Note that if the default value X, supplied from the calling program, is not 1, 2, or 3, this routine will branch to EROR. This is important, as calling programs depend on GTAB to validate their default values. Also, this routine may be forced to not permit defaults by supplying an erroneous default value.

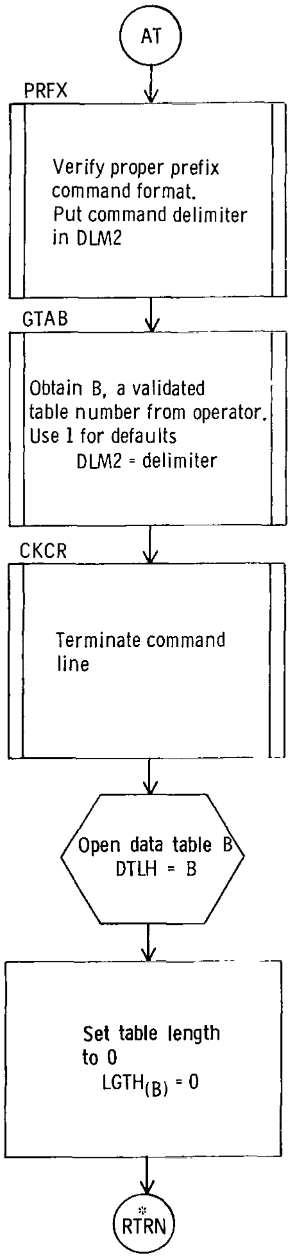

CMDS
Interactive mode set EROR and all command processors to jump here when finished
Save program counter for this place in RTRN
Set IN$ and OUT$ devices to operator's console
MESAGE
Type underscore and bell on OUT$ device
NAME
Note 1
Input name from operator. Pack in BUF. Save delimiter in DLM2
Note 2
Is DLM2 equal to:
"CR" --
. --
@ -- AT
: -- CALC
< -- SPL
= -- EQU
> -- ODAT
? -- QURY
[-- VT
\ -- FORM
] -- EVT
↑ -- UARW
"Space" -- DEFN
! -- PNCH
" -- QUTE
$ -- DUMP
% -- DFMT
& -- DISP
(-- OTBL
) -- CTBL
* -- LOADS
, -- ADD1
- -- CRTR
/ -- LIST
Other -- EROR
ENTER INFORM with "FORMAT" argument
Input operator's mode select switch
Test mode select switch
Set
Reset
Test VTLH
Reset
Set
Determine if value of memory whose address is in ADDR is at trip condition
A = ADDR B = 0
Compare A to TRIP
<
>
=
B = B +
B = B + 1
Does B = EQTY ?
Yes
No
A is at trip condition, execute INST instruction
Input operator's mode select switch
Test mode select switch
Set
Reset
RETURN
EXIT
Decode format type from INFORM's "FORMAT" argument. Save in TFMT
Decode table No. from INFORM's "FORMAT" argument. Save in TABN
Decode output unit number from INFORM's "FORMAT" argument. Save in OUT$
If TFMT, TABN, or OUT$ = 0, set it = to DF0, DF1, or DF3, respectively
PTBL

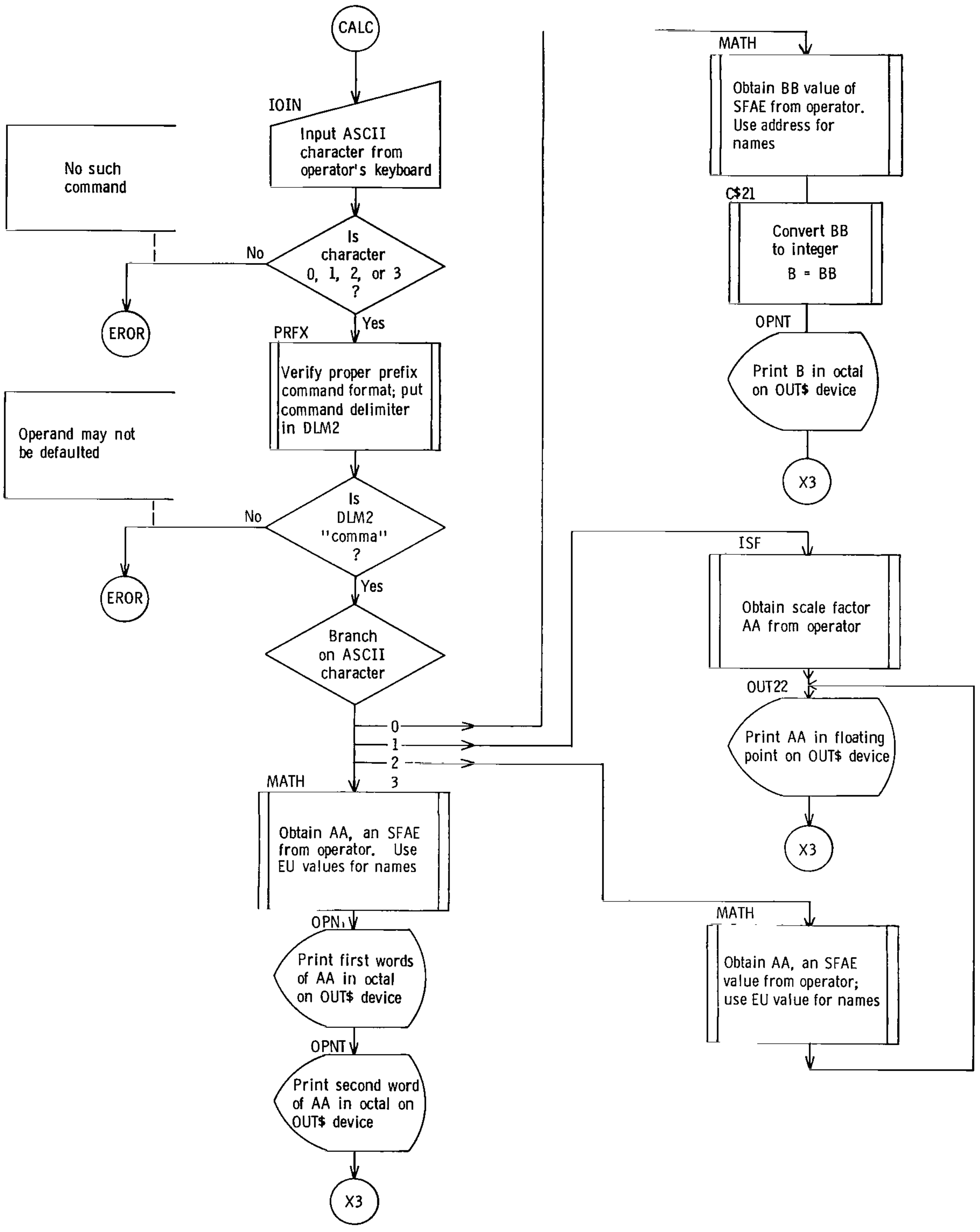

PTBL
SETU
Set STOP equal to array of addresses of named locations in data table TABN; Set L = number of addresses
Set Dac 85 pointer to TABN; STOP = length of TABN
Data table is undefined
Is STOP = 0 ?
Yes
No
EROR
Initialize loop counters I = K = L = 0
IVAR
Note 3
Obtain Ith indexing variable from data table. Save as J
Is J > 0 ?
No
Yes
STOR(L) = [LOC(J)]
Add increment to indexing variable L = L + 1
Add increment to loop counter I = I + 1
Is I < STOP ?
Yes
No
For I = 0 to (L - 1), set STOR(I) = [STOR(I)]
Test TFMT
0
1
2
3
Other
EROR
DATA
FMTO
Print heading of names for data table TABN
MESAGE
Print "CRLF" on OUT$ device
SPCR
Get J, the Kth indexing variable from TABN. Test J and print "CRLF" or space codes
J is named location indexing variable
J is "CRLF" or space code
MESAGE
Print two spaces on OUT$ device
PNAM
Print name for indexing variable J on OUT$ device
MESAGE
Print one space on OUT$ device
Add increment to loop counter K = K + 1
Is K < STOP ?
Yes
No
MESAGE
Print "CRLF" on OUT$ device
DATA

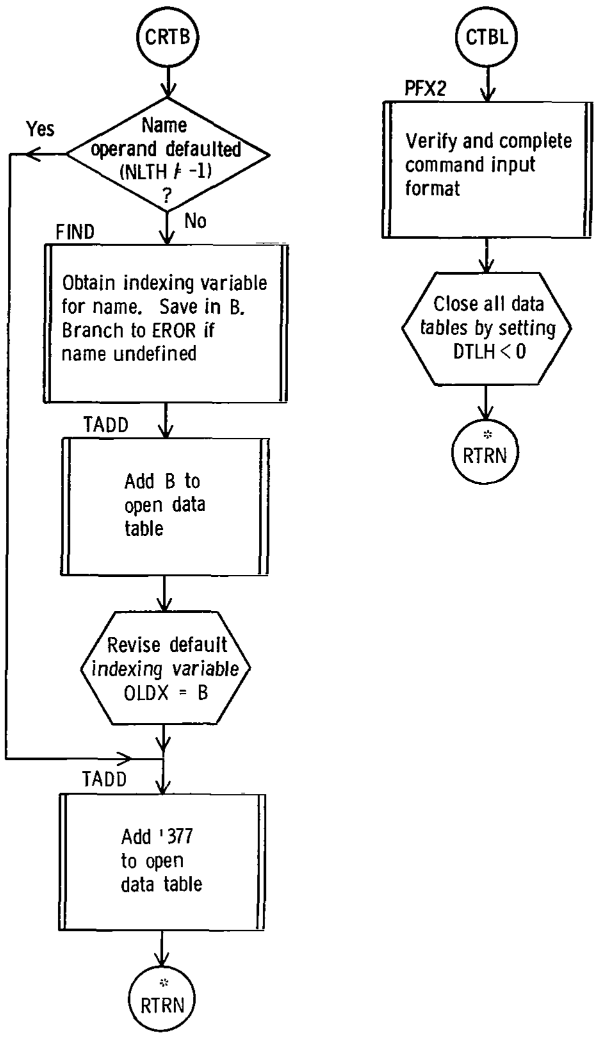

DATA
Print data for data table TABN for TMFT of 1, 2, and 3 on OUT$ device
Initialize loop counters I = L = 0
SPCR
Get J, the Ith indexing variable from TABN. Test J and print "CRLF" or space for appropriate code
J is named location indexing variable
J is "CRLF" or space code
Increment loop counter I = I + 1
Is I < STOP ?
EXIT
Is TFMT = 3 ?
No
Yes
MESAGE
Print one space on OUT$ device
Remove this function to reduce format type 3 to 18 characters
PNAM
Print name for indexing variable J on OUT$ device
MESAGE
Print "=" on OUT$ device
VAL
Print EU value of STOR(L) using scaling of Jth name on OUT$ device
Increment indexing variable L = L + 1
FMTO
Output data for data table TABN for TFMT = 0
FDDH
Start dump on OUT$ device ID No. = 5 digits of L, 12 words/line, code = '7205
FDD11
Dump coded integer word indicating 1 channel Word = 1
FDDEND
Close dump heading. Print check sum for preceding word
IVAR
Get J, the Kth indexing variable from data table TABN
Is J > 0
No
Yes
GSF2
Obtain SVSF, a limited scale factor for indexing variable J
C$12
Convert STOR(L) to floating print AA = STOR(L)
Calculate EU AA = AA · SVSF
FDD22
Dump AA on OUT$ device
Increment indexing variable L = L + 1
Increment loop counter I = I + 1
Is I < STOP ?
Yes
No
EXIT

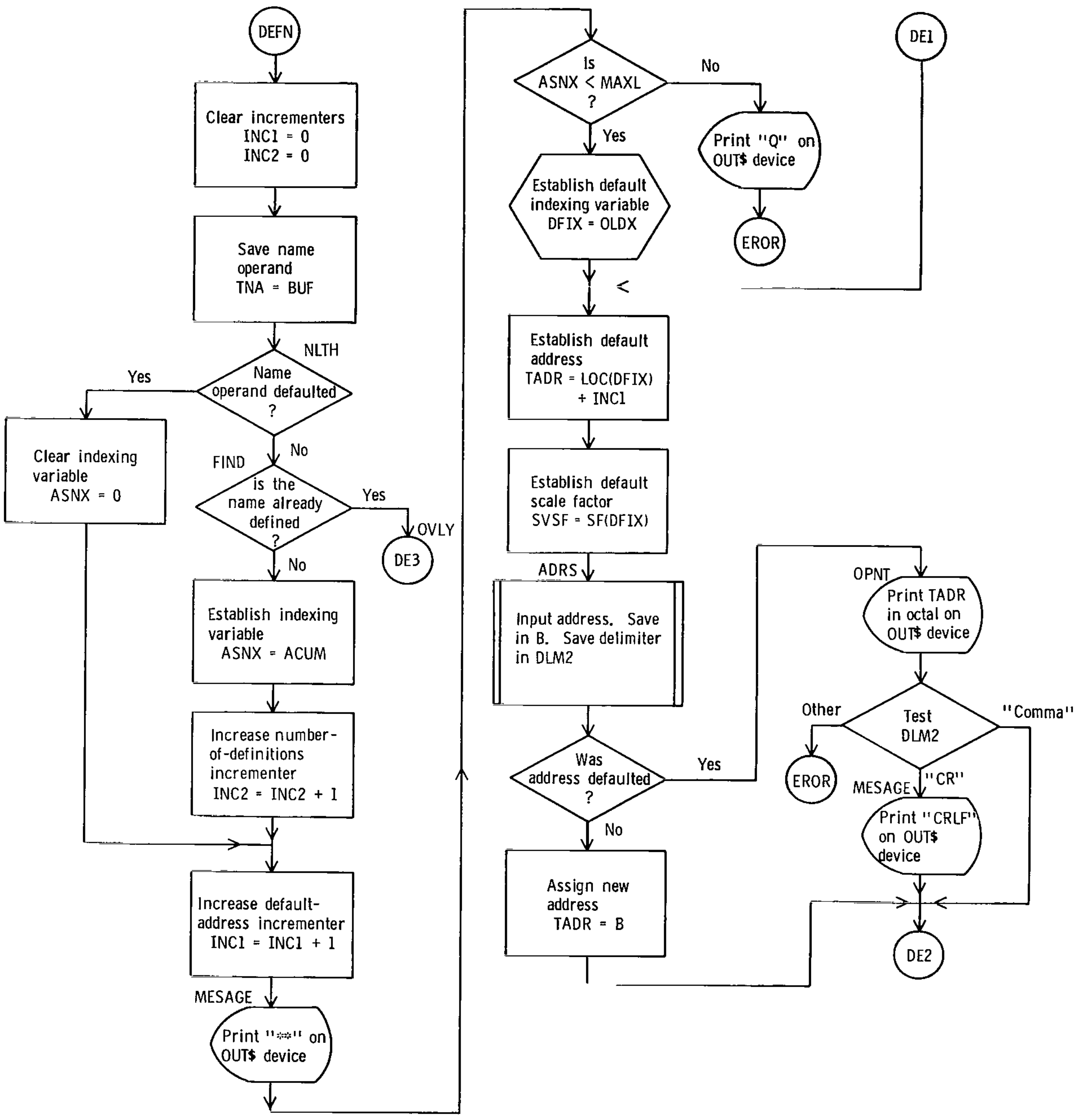
Except for ODAT and SPL, the command processors for the INFORM commands follow in alphabetical order.

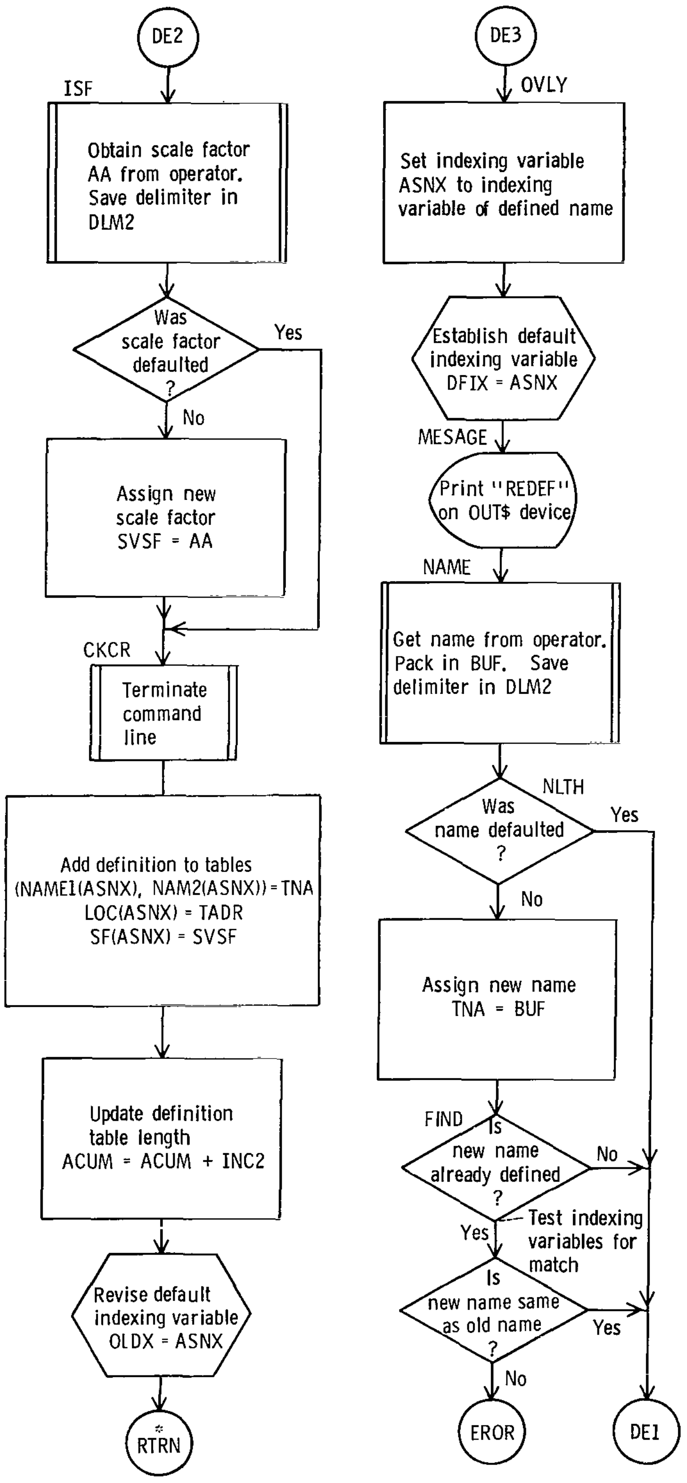

ADD1
SUFX
Obtain indexing variable for name operand. Use zero for default. Save in INDX
TADD
Add INDX to open data table
X2

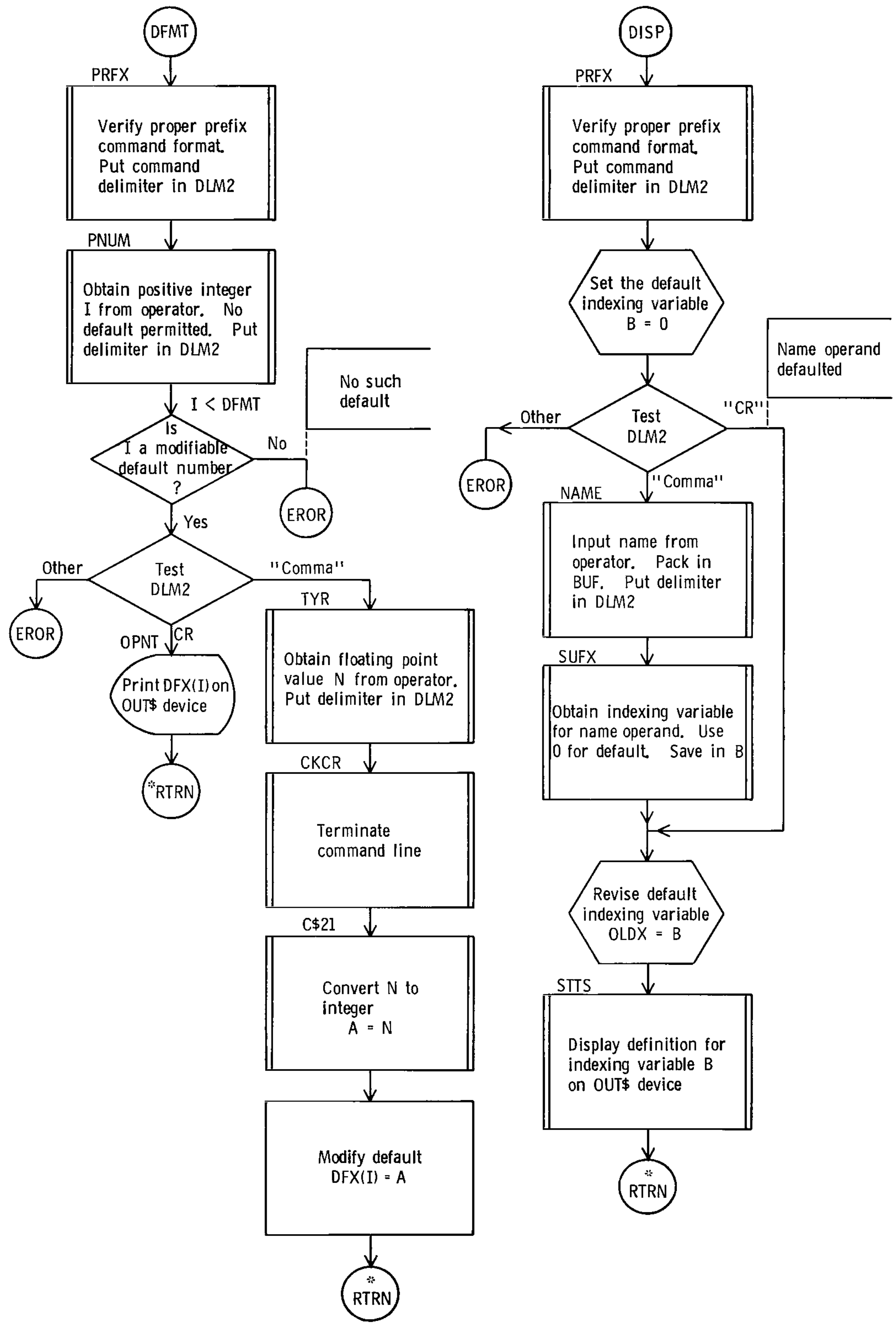

AT
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
GTAB
Obtain B, a validated table number from operator. Use 1 for defaults DLM2 = delimiter
CKCR
Terminate command line
Open data table B DTLH = B
Set table length to 0 LGTH(B) = 0
*RTRN

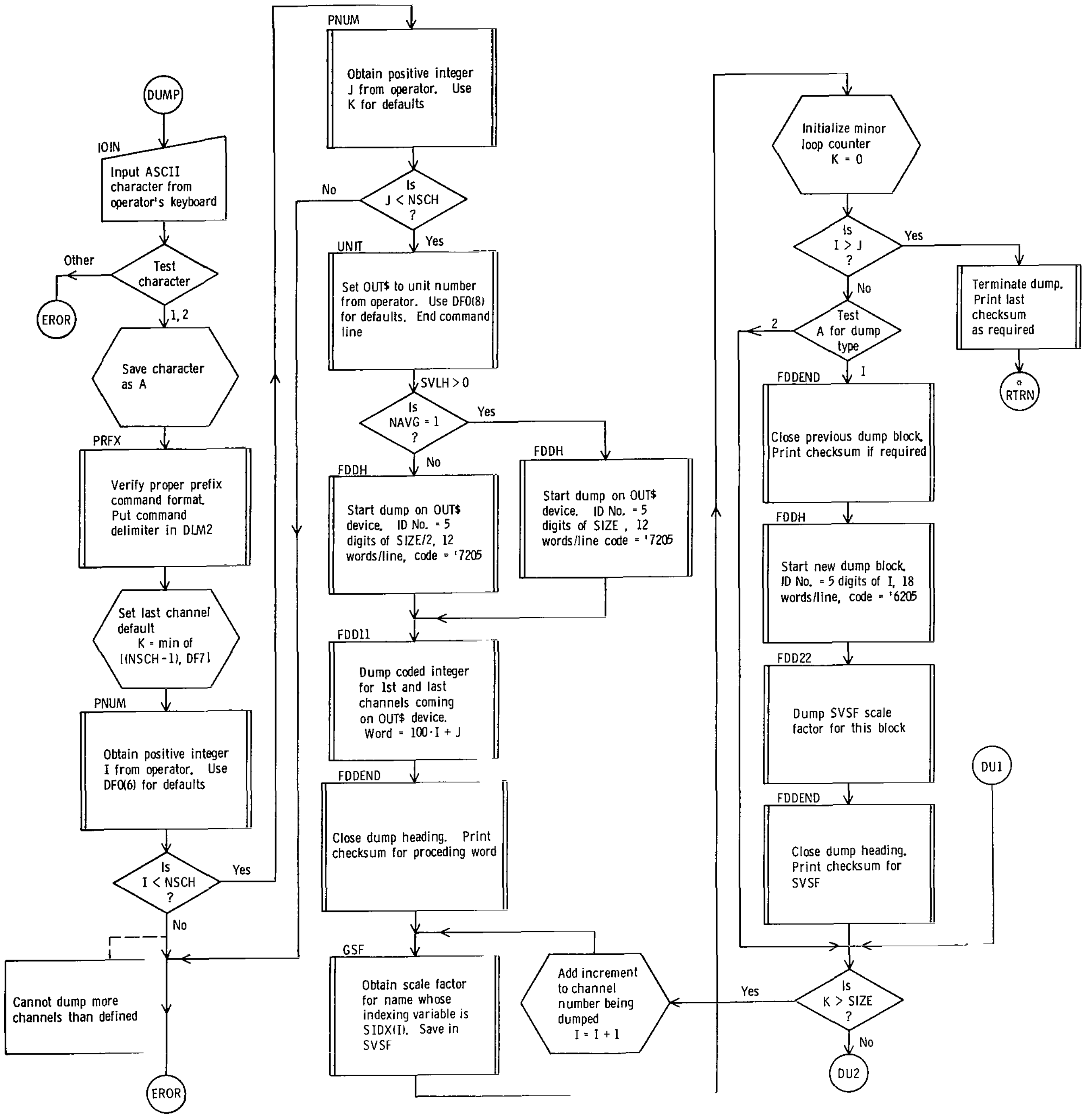

CALC
IOIN
Input ASCII character from operator's keyboard
No such command
Is character 0, 1, 2, or 3 ?
No
EROR
Yes
PRFX
Verify proper prefix command format; put command delimiter in DLM2
Operand may not be defaulted
Is DLM2 "comma" ?
No
EROR
Yes
Branch on ASCII character
0
1
2
3
MATH
Obtain AA, an SFAE from operator. Use EU values for names
OPN1
Print first words of AA in octal on OUT$ device
OPNT
Print second word of AA in octal on OUT$ device
X3
MATH
Obtain BB value of SFAE from operator. Use address for names
C$21
Convert BB to integer B = BB
OPNT
Print B in octal on OUT$ device
X3
ISF
Obtain scale factor AA from operator
OUT22
Print AA in floating point on OUT$ device
X3
MATH
Obtain AA, an SFAE value from operator; use EU value for names

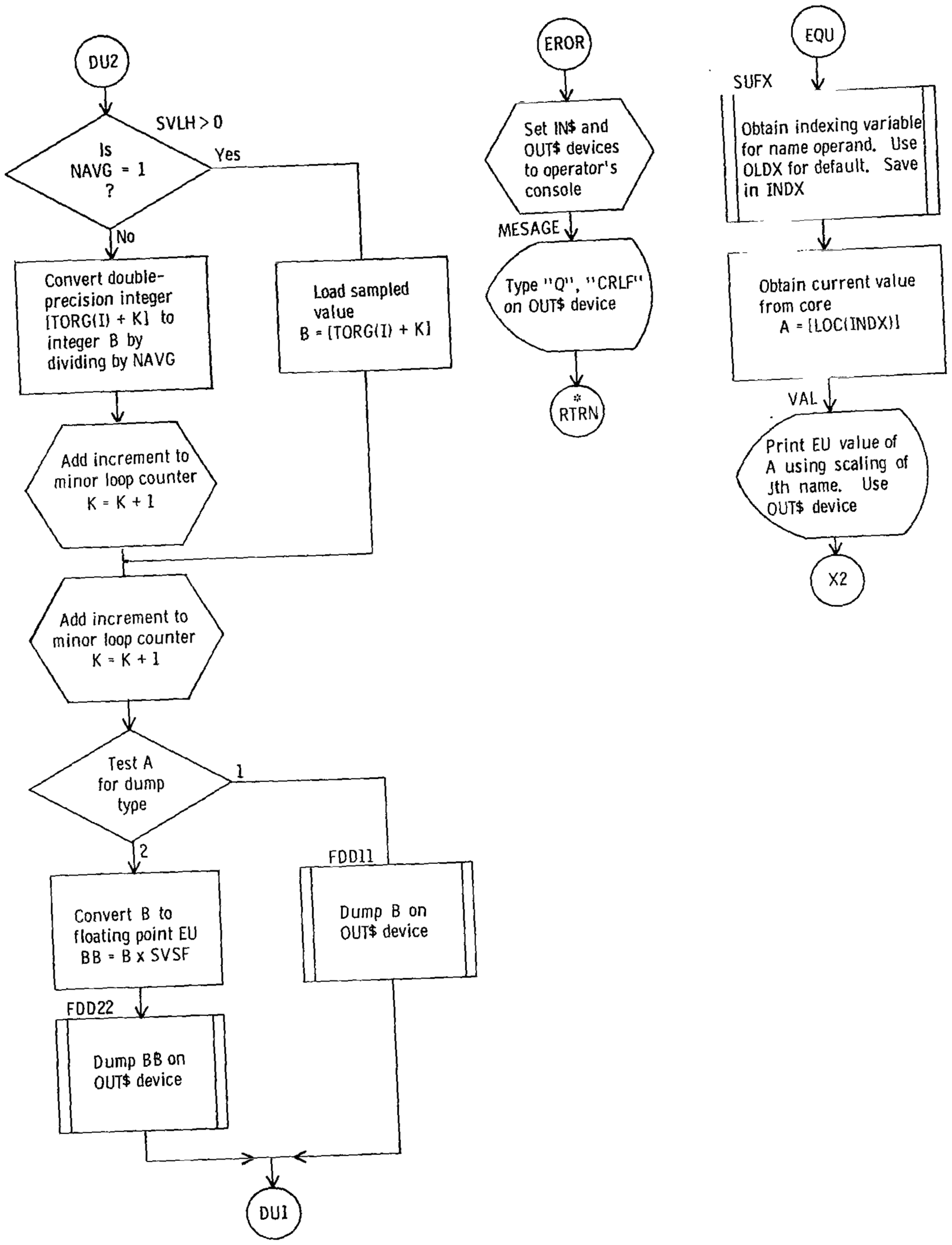

CRTB
Name operand defaulted (NLTH ≠ -1) ?
Yes
No
FIND
Obtain indexing variable for name. Save in B. Branch to EROR if name undefined
TADD
Add B to open data table
Revise default indexing variable OLDX = B
TADD
Add '377 to open data table
* RTRN
CTBL
PFX2
Verify and complete command input format
Close all data tables by setting DTLH < 0
* RTRN


DEFN
Clear incrementers INC1 = 0 INC2 = 0
Save name operand TNA = BUF
NLTH
Name operand defaulted ?
Yes
Clear indexing variable ASNX = 0
No
FIND
Is the name already defined ?
Yes
OVLY
DE3
No
Establish indexing variable ASNX = ACUM
Increase number-of-definitions incrementer INC2 = INC2 + 1
Increase default-address incrementer INC1 = INC1 + 1
MESAGE
Print "**" on OUT$ device
Is ASNX < MAXL ?
No
Print "Q" on OUT$ device
EROR
Yes
Establish default indexing variable DFIX = OLDX
DE1
Establish default address TADR = LOC(DFIX) + INC1
Establish default scale factor SVSF = SF(DFIX)
ADRS
Input address. Save in B. Save delimiter in DLM2
Was address defaulted ?
Yes
No
Assign new address TADR = B
OPNT
Print TADR in octal on OUT$ device
Test DLM2
Other
"Comma"
EROR
"CR"
MESAGE
Print "CRLF" on OUT$ device
DE2

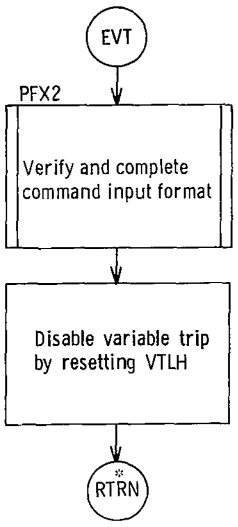

DE2
ISF
Obtain scale factor AA from operator. Save delimiter in DLM2
Was scale factor defaulted ?
Yes
No
Assign new scale factor SVSF = AA
CKCR
Terminate command line
Add definition to tables (NAME1(ASNX), NAM2(ASNX))=TNA LOC(ASNX) = TADR SF(ASNX) = SVSF
Update definition table length ACUM = ACUM + INC2
Revise default indexing variable OLDX = ASNX
* RTRN
DE3
OVLY
Set indexing variable ASNX to indexing variable of defined name
Establish default indexing variable DFIX = ASNX
MESAGE
Print "REDEF" on OUT$ device
NAME
Get name from operator. Pack in BUF. Save delimiter in DLM2
NLTH
Was name defaulted ?
Yes
No
Assign new name TNA = BUF
FIND
Is new name already defined ?
No
Yes
Test indexing variables for match
Is new name same as old name ?
Yes
No
EROR
DE1

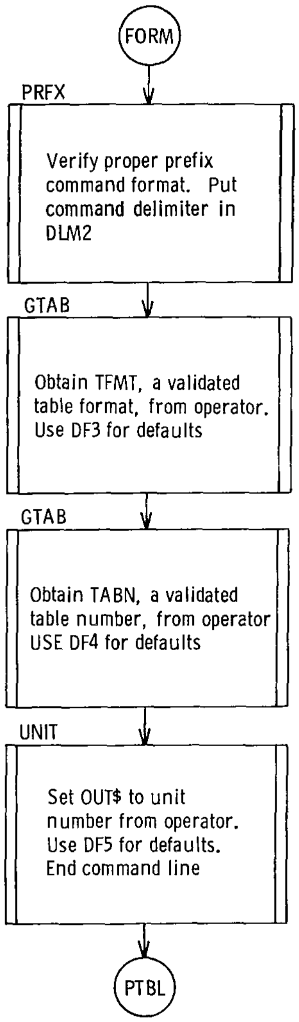

DFMT
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
PNUM
Obtain positive integer I from operator. No default permitted. Put delimiter in DLM2
I < DFMT
Is I a modifiable default number ?
No
No such default
EROR
Yes
Other
Test DLM2
"Comma"
EROR
OPNT
CR
Print DFX(I) on OUT$ device
*RTRN
TYR
Obtain floating point value N from operator. Put delimiter in DLM2
CKCR
Terminate command line
C$21
Convert N to integer A = N
Modify default DFX(I) = A
*RTRN
DISP
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
Set the default indexing variable B = 0
Name operand defaulted
Other
Test DLM2
"CR"
EROR
"Comma"
NAME
Input name from operator. Pack in BUF. Put delimiter in DLM2
SUFX
Obtain indexing variable for name operand. Use 0 for default. Save in B
Revise default indexing variable OLDX = B
STTS
Display definition for indexing variable B on OUT$ device
*RTRN

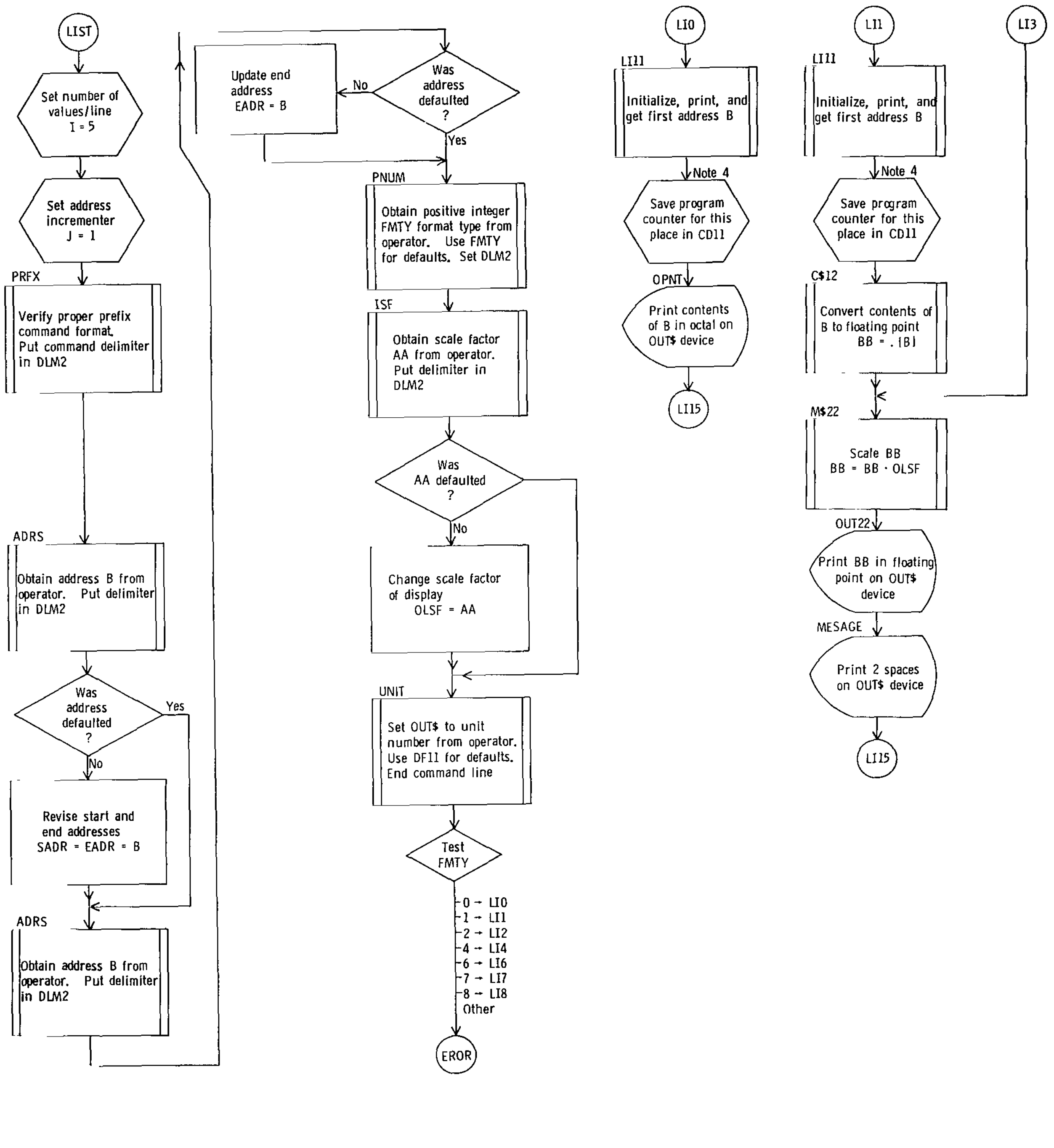

DUMP
IOIN
Input ASCII character from operator's keyboard
Test character
Other
EROR
1, 2
Save character as A
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
Set last channel default K = min of [(NSCH-1), DF7]
PNUM
Obtain positive integer I from operator. Use DF0(6) for defaults
Is I < NSCH ?
Yes
No
Cannot dump more channels than defined
EROR
PNUM
Obtain positive integer J from operator. Use K for defaults
Is J < NSCH ?
No
Yes
UNIT
Set OUT$ to unit number from operator. Use DF0(8) for defaults. End command line
SVLH > 0
Is NAVG = 1 ?
Yes
No
FDDH
Start dump on OUT$ device. ID No. = 5 digits of SIZE/2, 12 words/line, code = '7205
FDDH
Start dump on OUT$ device. ID No. = 5 digits of SIZE , 12 words/line code = '7205
FDD11
Dump coded integer for 1st and last channels coming on OUT$ device. Word = 100·I + J
FDDEND
Close dump heading. Print checksum for proceding word
GSF
Obtain scale factor for name whose indexing variable is SIDX(I). Save in SVSF
Add increment to channel number being dumped I = I + 1
Initialize minor loop counter K = 0
Is I > J ?
Yes
Terminate dump. Print last checksum as required
RTRN
No
Test A for dump type
2
I
FDDEND
Close previous dump block. Print checksum if required
FDDH
Start new dump block. ID No. = 5 digits of I, 18 words/line, code = '6205
FDD22
Dump SVSF scale factor for this block
FDDEND
Close dump heading. Print checksum for SVSF
DU1
Is K > SIZE ?
Yes
No
DU2

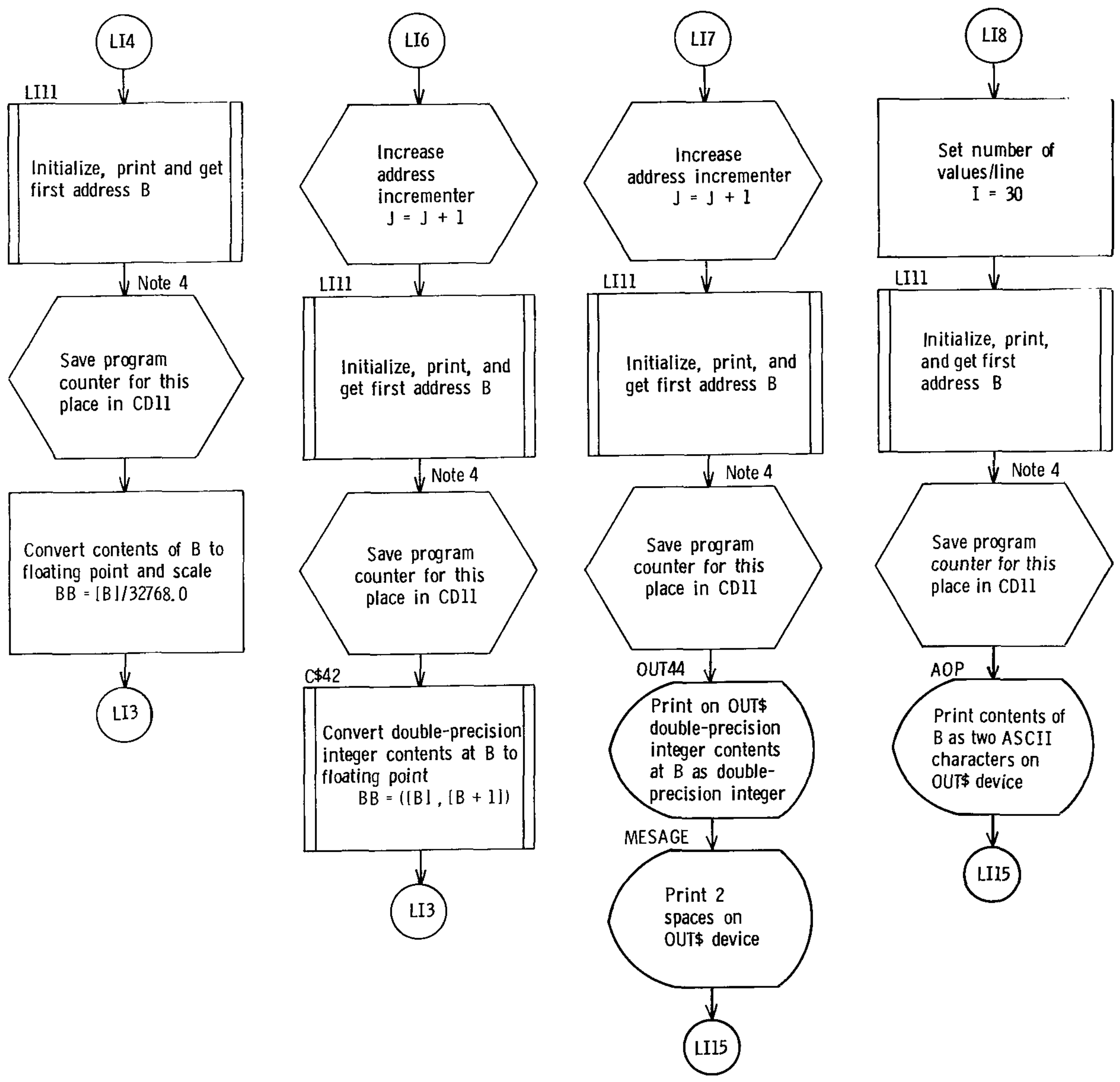


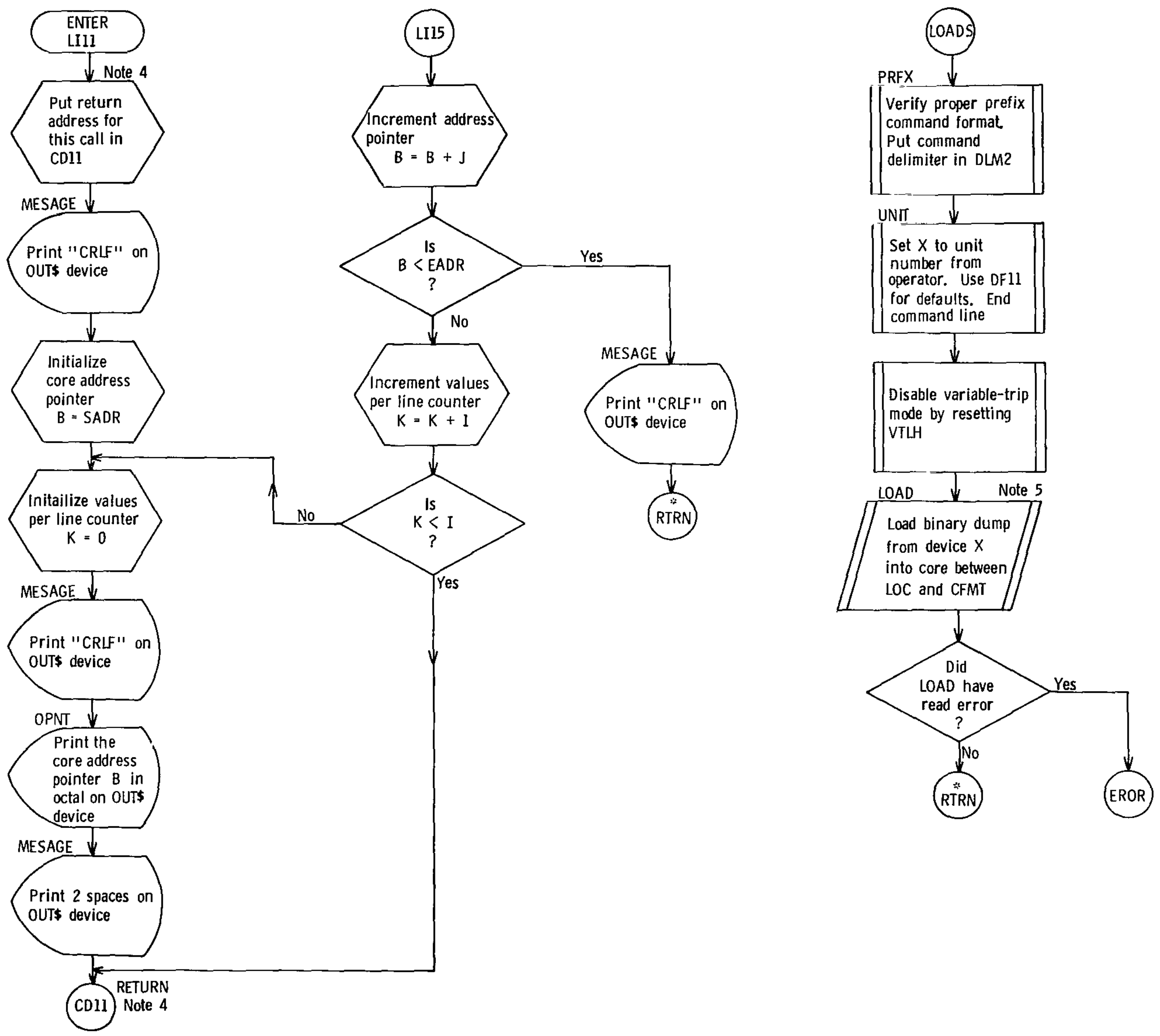
DU2
SVLH > 0
Is NAVG = 1 ?
Yes
No
Convert double-precision integer [TORG(I) + K] to integer B by dividing by NAVG
Load sampled value B = [TORG(I) + K]
Add increment to minor loop counter K = K + 1
Add increment to minor loop counter K = K + 1
Test A for dump type
1
2
Convert B to floating point EU BB = B x SVSF
FDD11
Dump B on OUT$ device
FDD22
Dump BB on OUT$ device
DU1
EROR
Set IN$ and OUT$ devices to operator's console
MESAGE
Type "Q", "CRLF" on OUT$ device
* RTRN
EQU
SUFX
Obtain indexing variable for name operand. Use OLDX for default. Save in INDX
Obtain current value from core A = [LOC(INDX)]
VAL
Print EU value of A using scaling of Jth name. Use OUT$ device
X2

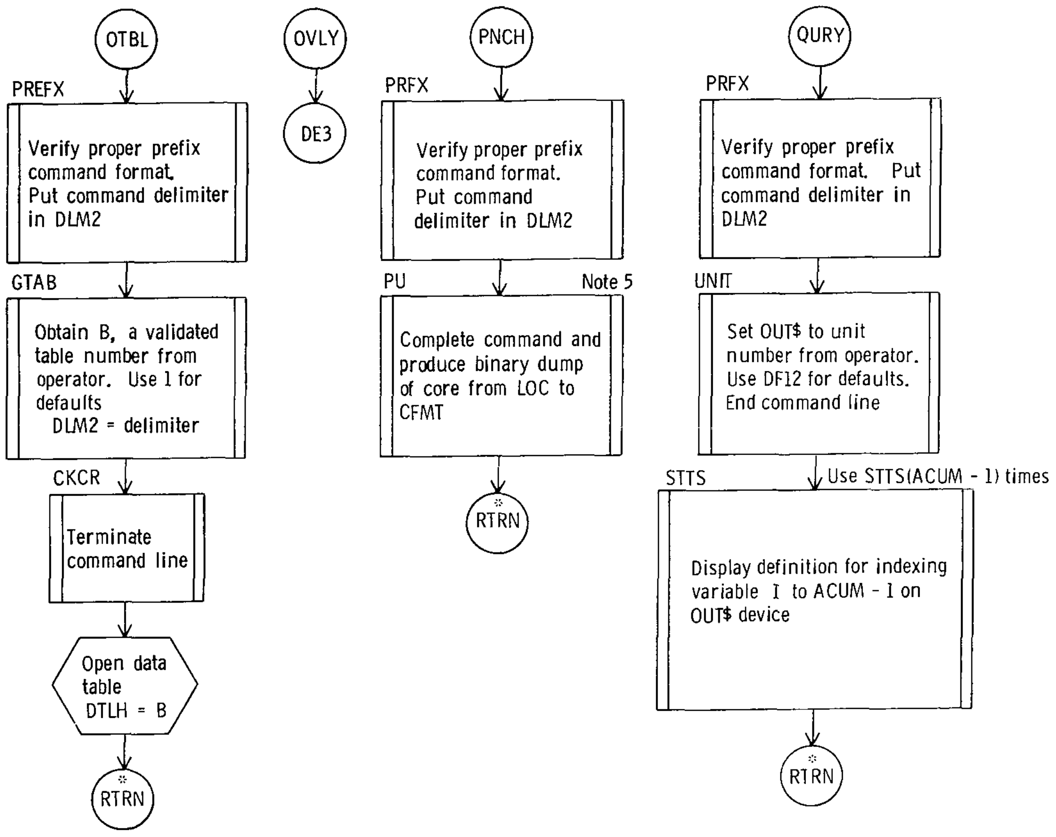

EVT
PFX2
Verify and complete command input format
Disable variable trip by resetting VTLH
*
RTRN


FORM
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
GTAB
Obtain TFMT, a validated table format, from operator. Use DF3 for defaults
GTAB
Obtain TABN, a validated table number, from operator USE DF4 for defaults
UNIT
Set OUT$ to unit number from operator. Use DF5 for defaults. End command line
PTBL

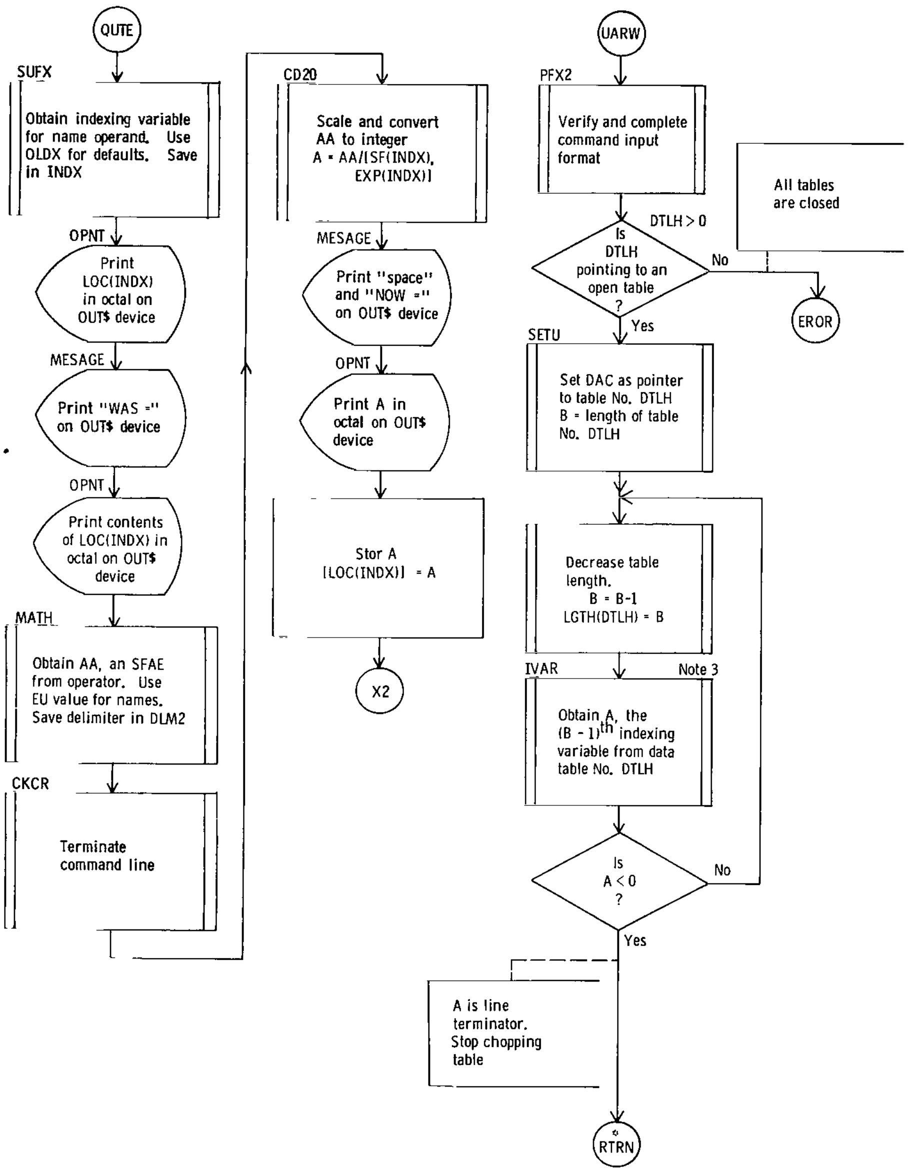

LIST
Set number of values/line I = 5
Set address incrementer J = 1
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
ADRS
Obtain address B from operator. Put delimiter in DLM2
Was address defaulted ?
Yes
No
Revise start and end addresses SADR = EADR = B
ADRS
Obtain address B from operator. Put delimiter in DLM2
Update end address EADR = B
No
Was address defaulted ?
Yes
PNUM
Obtain positive integer FMTY format type from operator. Use FMTY for defaults. Set DLM2
ISF
Obtain scale factor AA from operator. Put delimiter in DLM2
Was AA defaulted ?
No
Change scale factor of display OLSF = AA
UNIT
Set OUT$ to unit number from operator. Use DF11 for defaults. End command line
Test FMTY
0 → LI0
1 → LI1
2 → LI2
4 → LI4
6 → LI6
7 → LI7
8 → LI8
Other
EROR
LI0
LI11
Initialize, print, and get first address B
Note 4
Save program counter for this place in CD11
OPNT
Print contents of B in octal on OUT$ device
LI15
LI1
LI11
Initialize, print, and get first address B
Note 4
Save program counter for this place in CD11
C$12
Convert contents of B to floating point BB = .(B)
LI3
M$22
Scale BB BB = BB · OLSF
OUT22
Print BB in floating point on OUT$ device
MESAGE
Print 2 spaces on OUT$ device
LI15

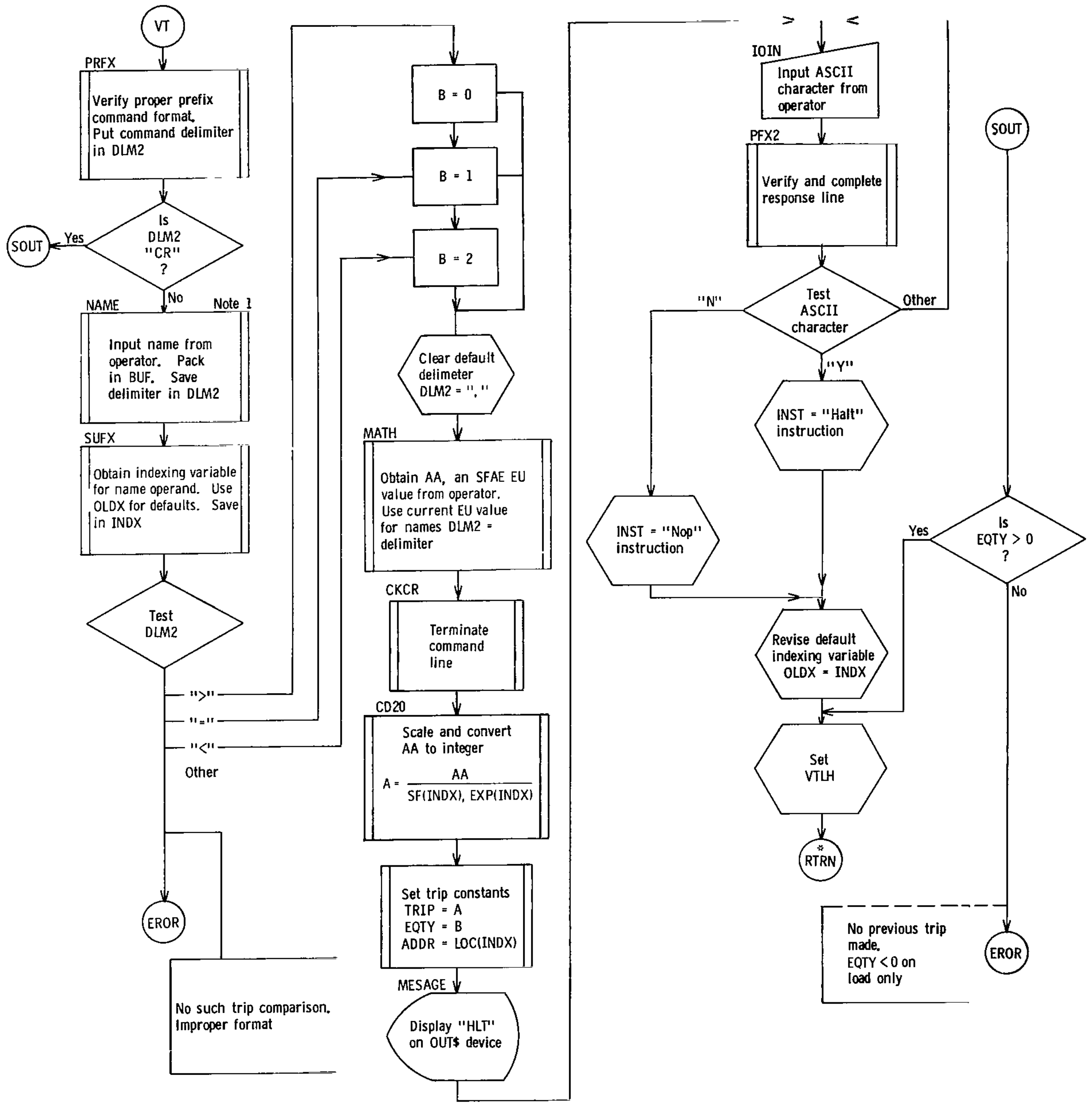

LI4
LI11
Initialize, print and get first address B
Note 4
Save program counter for this place in CD11
Convert contents of B to floating point and scale BB = [B]/32768.0
LI3
LI6
Increase address incrementer J = J + 1
LI11
Initialize, print, and get first address B
Note 4
Save program counter for this place in CD11
C$42
Convert double-precision integer contents at B to floating point BB = ([B], [B + 1])
LI3
LI7
Increase address incrementer J = J + 1
LI11
Initialize, print, and get first address B
Note 4
Save program counter for this place in CD11
OUT44
Print on OUT$ double-precision integer contents at B as double-precision integer
MESAGE
Print 2 spaces on OUT$ device
LI15
LI8
Set number of values/line I = 30
LI11
Initialize, print, and get first address B
Note 4
Save program counter for this place in CD11
AOP
Print contents of B as two ASCII characters on OUT$ device
LI15

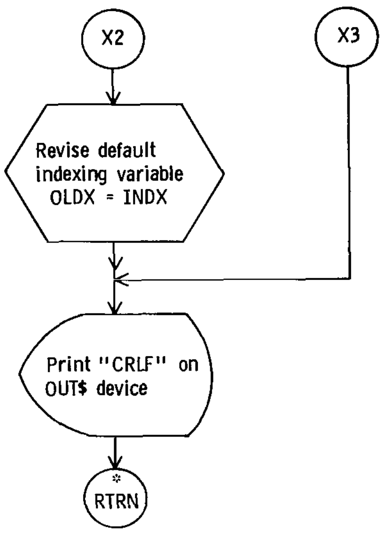

ENTER LI11
Note 4
Put return address for this call in CD11
MESAGE
Print "CRLF" on OUT$ device
Initialize core address pointer B = SADR
Initailize values per line counter K = 0
MESAGE
Print "CRLF" on OUT$ device
OPNT
Print the core address pointer B in octal on OUT$ device
MESAGE
Print 2 spaces on OUT$ device
CD11
RETURN Note 4
LI15
Increment address pointer B = B + J
Is B < EADR ?
Yes
No
MESAGE
Print "CRLF" on OUT$ device
RTRN
Increment values per line counter K = K + 1
Is K < I ?
No
Yes
LOADS
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
UNIT
Set X to unit number from operator. Use DF11 for defaults. End command line
Disable variable-trip mode by resetting VTLH
LOAD
Note 5
Load binary dump from device X into core between LOC and CFMT
Did LOAD have read error ?
Yes
No
RTRN
EROR



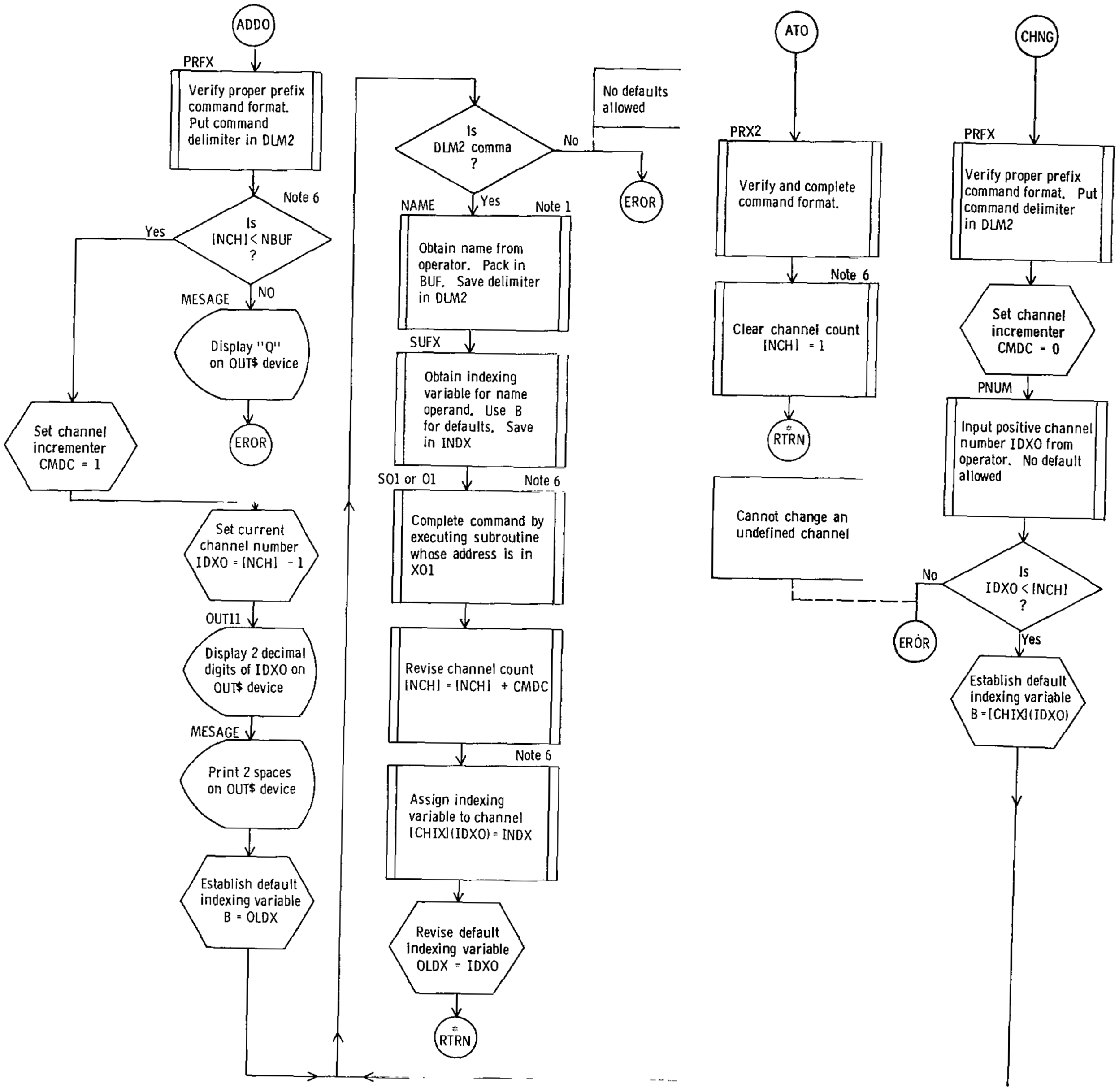
OTBL
PREFX
Verify proper prefix command format. Put command delimiter in DLM2
GTAB
Obtain B, a validated table number from operator. Use 1 for defaults DLM2 = delimiter
CKCR
Terminate command line
Open data table DTLH = B
RTRN
OVLY
DE3
PNCH
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
PU
Note 5
Complete command and produce binary dump of core from LOC to CFMT
RTRN
QURY
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
UNIT
Set OUT$ to unit number from operator. Use DF12 for defaults. End command line
STTS
Use STTS(ACUM - 1) times
Display definition for indexing variable I to ACUM - 1 on OUT$ device
RTRN

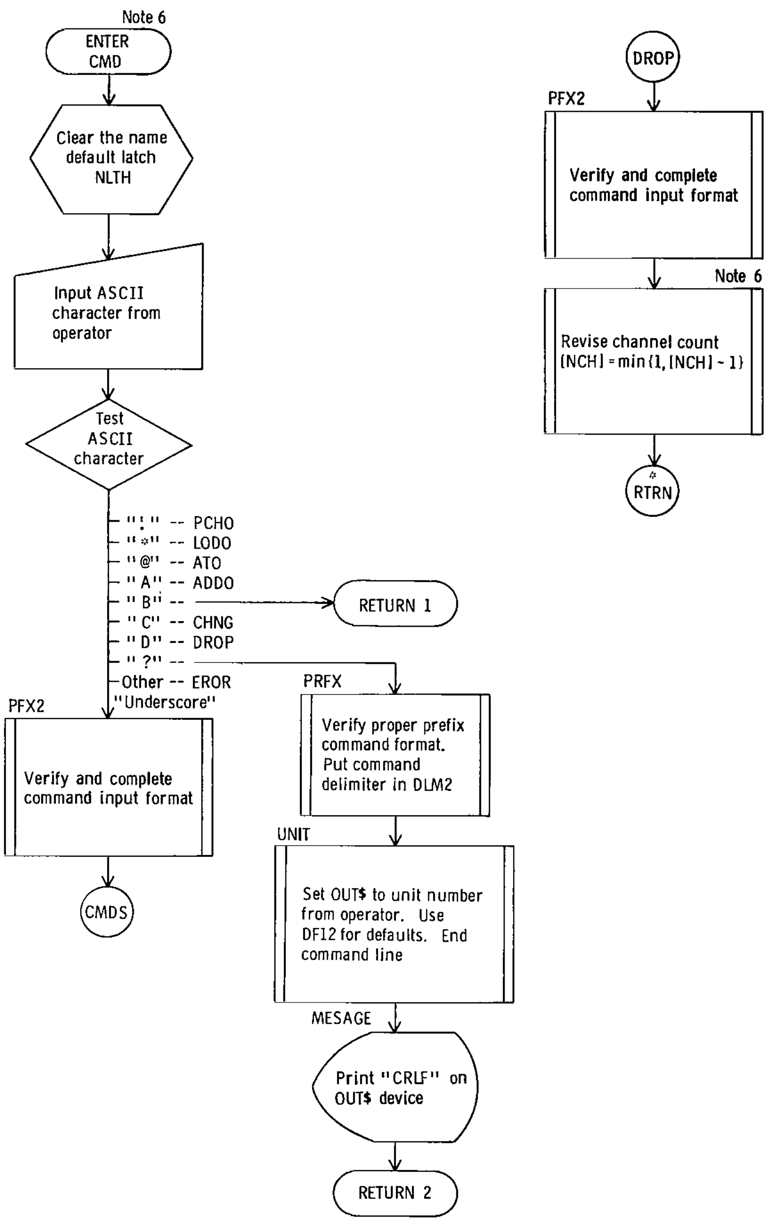

QUTE
SUFX
Obtain indexing variable for name operand. Use OLDX for defaults. Save in INDX
OPNT
Print LOC(INDX) in octal on OUT$ device
MESAGE
Print "WAS =" on OUT$ device
OPNT
Print contents of LOC(INDX) in octal on OUT$ device
MATH
Obtain AA, an SFAE from operator. Use EU value for names. Save delimiter in DLM2
CKCR
Terminate command line
CD20
Scale and convert AA to integer A = AA/[SF(INDX), EXP(INDX)]
MESAGE
Print "space" and "NOW =" on OUT$ device
OPNT
Print A in octal on OUT$ device
Stor A [LOC(INDX)] = A
X2
UARW
PFX2
Verify and complete command input format
DTLH > 0
Is DTLH pointing to an open table ?
No
All tables are closed
EROR
Yes
SETU
Set DAC as pointer to table No. DTLH B = length of table No. DTLH
Decrease table length. B = B-1 LGTH(DTLH) = B
IVAR
Note 3
Obtain A, the (B - 1)th indexing variable from data table No. DTLH
Is A < 0 ?
No
Yes
A is line terminator. Stop chopping table
RTRN

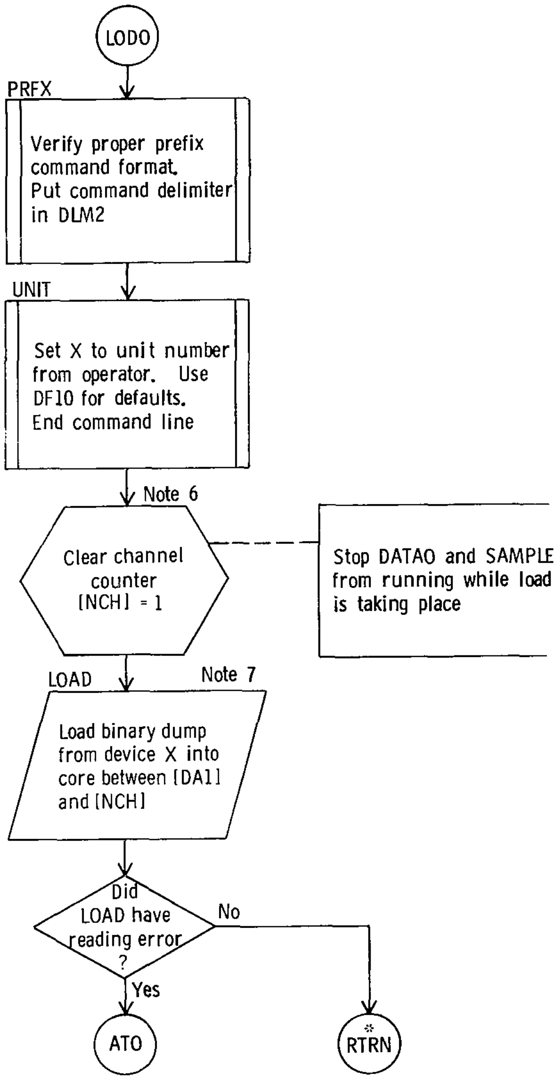

VT
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
Is DLM2 "CR" ?
Yes
SOUT
No
NAME
Note 1
Input name from operator. Pack in BUF. Save delimiter in DLM2
SUFX
Obtain indexing variable for name operand. Use OLDX for defaults. Save in INDX
Test DLM2
">"
"="
"<"
Other
EROR
No such trip comparison. Improper format
B = 0
B = 1
B = 2
Clear default delimeter DLM2 = ","
MATH
Obtain AA, an SFAE EU value from operator. Use current EU value for names DLM2 = delimiter
CKCR
Terminate command line
CD20
Scale and convert AA to integer
A = AA / SF(INDX), EXP(INDX)
Set trip constants TRIP = A EQTY = B ADDR = LOC(INDX)
MESAGE
Display "HLT" on OUT$ device
>
<
IOIN
Input ASCII character from operator
PFX2
Verify and complete response line
Test ASCII character
"N"
Other
"Y"
INST = "Halt" instruction
INST = "Nop" instruction
Revise default indexing variable OLDX = INDX
Set VTLH
* RTRN
SOUT
Is EQTY > 0 ?
Yes
No
No previous trip made. EQTY < 0 on load only
EROR


X2
X3
Revise default indexing variable OLDX = INDX
Print "CRLF" on OUT$ device
*RTRN

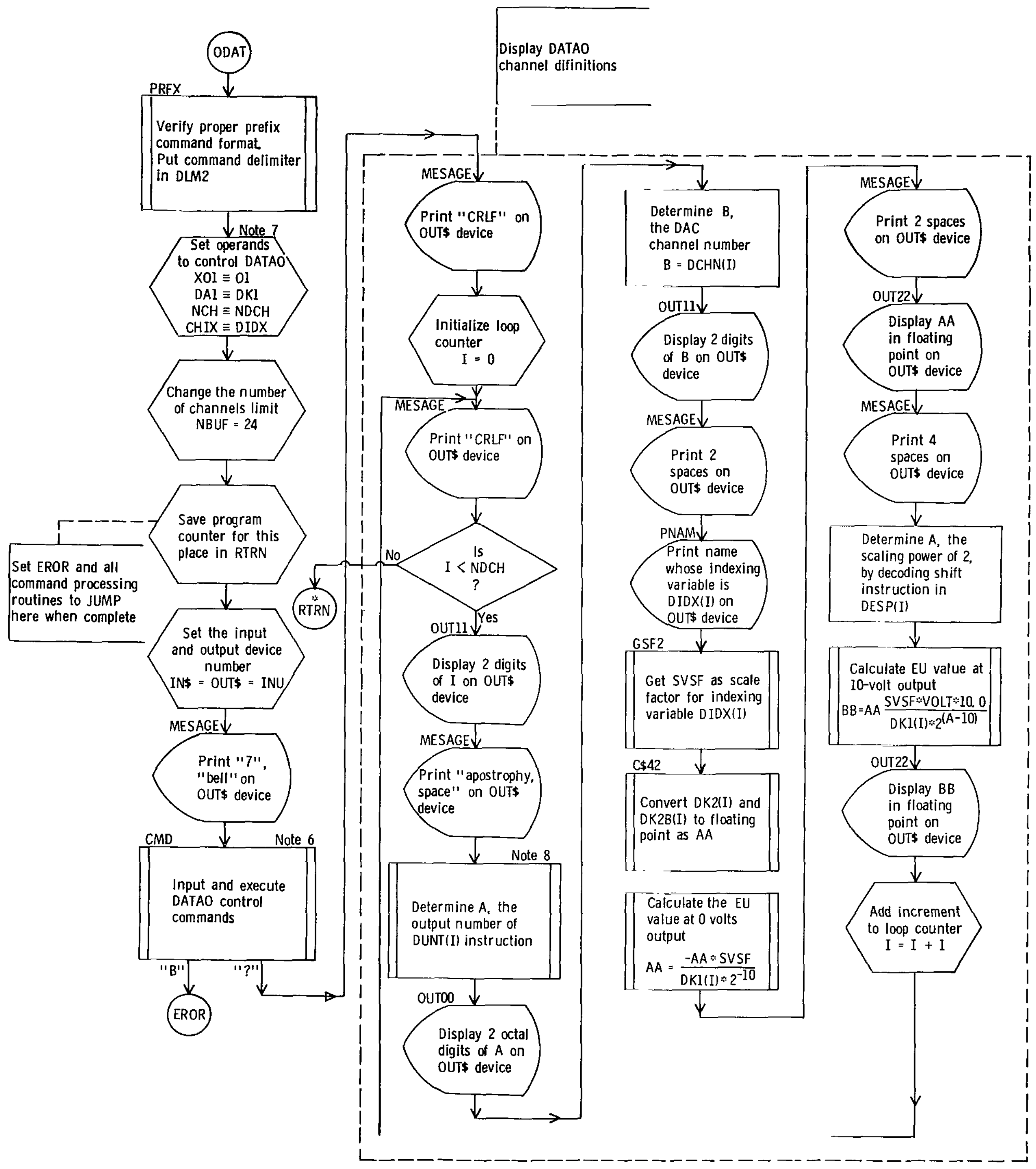
The ODAT and SPL command interpreters and their command processing routines follow in alphabetical order. See the DATAO and SAMPLE command processor section under PROGRAM STRUCTURE to understand the flow through the command processors.


ADDO
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
Note 6
Is [NCH] < NBUF ?
Yes
NO
MESAGE
Display "Q" on OUT$ device
EROR
Set channel incrementer CMDC = 1
Set current channel number IDXO = [NCH] - 1
OUT11
Display 2 decimal digits of IDXO on OUT$ device
MESAGE
Print 2 spaces on OUT$ device
Establish default indexing variable B = OLDX
Is DLM2 comma ?
No
No defaults allowed
EROR
Yes
NAME
Note 1
Obtain name from operator. Pack in BUF. Save delimiter in DLM2
SUFX
Obtain indexing variable for name operand. Use B for defaults. Save in INDX
S01 or 01
Note 6
Complete command by executing subroutine whose address is in X01
Revise channel count [NCH] = [NCH] + CMDC
Note 6
Assign indexing variable to channel [CHIX](IDXO) = INDX
Revise default indexing variable OLDX = IDXO
RTRN
ATO
PRX2
Verify and complete command format.
Note 6
Clear channel count [NCH] = 1
RTRN
Cannot change an undefined channel
CHNG
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
Set channel incrementer CMDC = 0
PNUM
Input positive channel number IDXO from operator. No default allowed
Is IDXO < [NCH] ?
No
EROR
Yes
Establish default indexing variable B = [CHIX](IDXO)

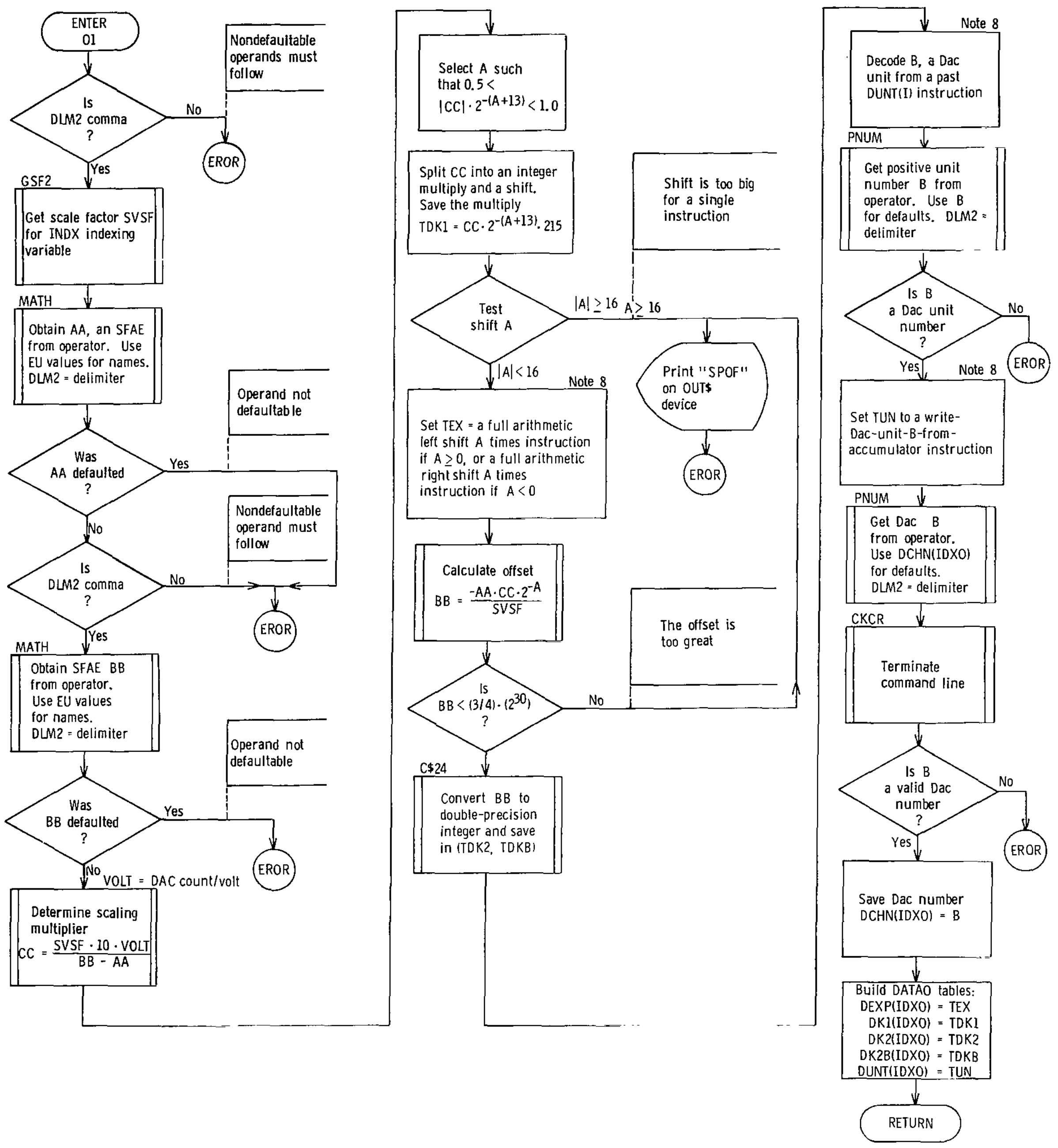

Note 6
ENTER CMD
Clear the name default latch NLTH
Input ASCII character from operator
Test ASCII character
"!" -- PCHO
"*" -- LODO
"@" -- ATO
"A" -- ADDO
"B" --
"C" -- CHNG
"D" -- DROP
"?" --
Other -- EROR
"Underscore"
RETURN 1
PFX2
Verify and complete command input format
CMDS
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
UNIT
Set OUT$ to unit number from operator. Use DF12 for defaults. End command line
MESAGE
Print "CRLF" on OUT$ device
RETURN 2
DROP
PFX2
Verify and complete command input format
Note 6
Revise channel count [NCH] = min{1, [NCH] - 1}
RTRN

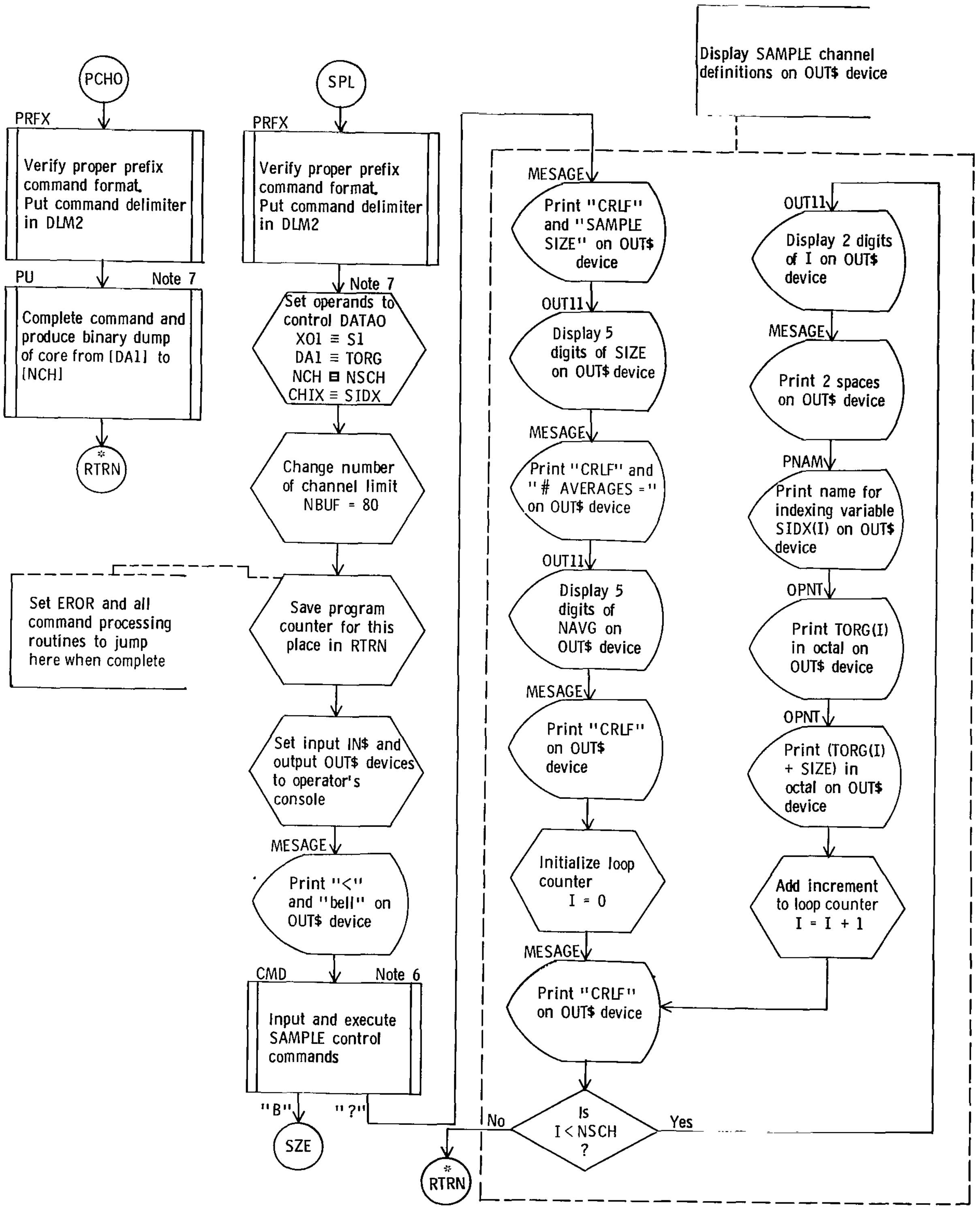


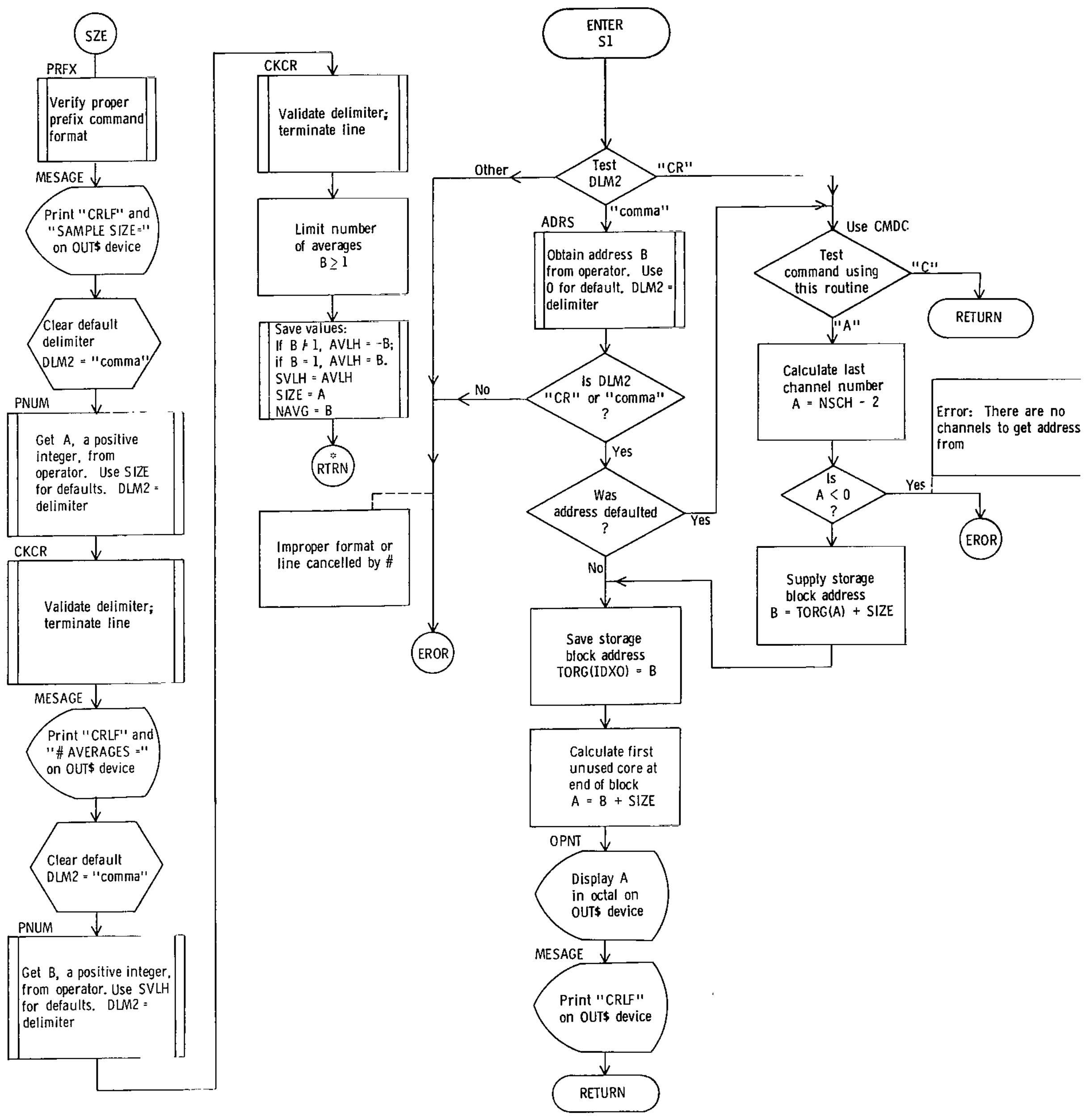
LODO
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
UNIT
Set X to unit number from operator. Use DF10 for defaults. End command line
Note 6
Clear channel counter [NCH] = 1
Stop DATAO and SAMPLE from running while load is taking place
LOAD
Note 7
Load binary dump from device X into core between [DA1] and [NCH]
Did LOAD have reading error ?
No
Yes
ATO
*
RTRN



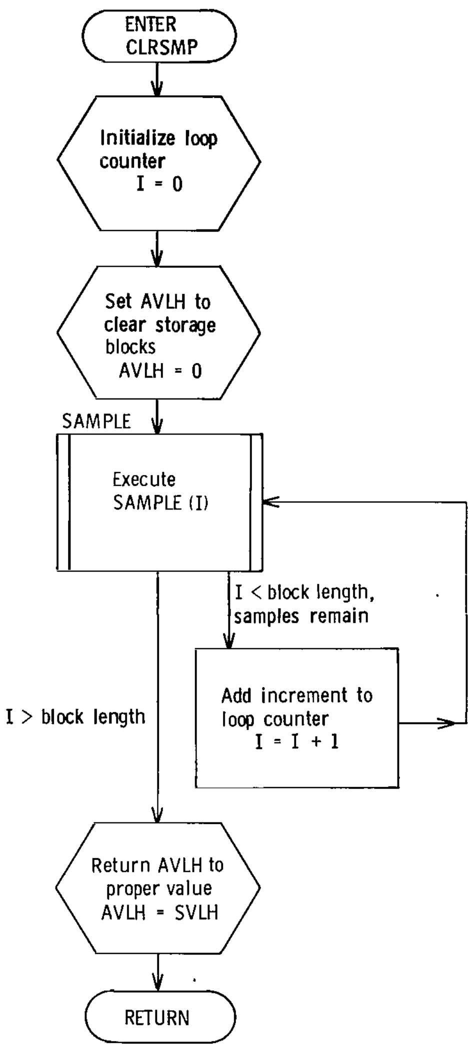
ODAT
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
Note 7
Set operands to control DATAO X01 ≡ 01 DA1 ≡ DK1 NCH ≡ NDCH CHIX ≡ DIDX
Change the number of channels limit NBUF = 24
Save program counter for this place in RTRN
Set EROR and all command processing routines to JUMP here when complete
Set the input and output device number IN$ = OUT$ = INU
MESAGE
Print "7", "bell" on OUT$ device
CMD
Note 6
Input and execute DATAO control commands
"B"
"?"
EROR
Display DATAO channel difinitions
MESAGE
Print "CRLF" on OUT$ device
Initialize loop counter I = 0
MESAGE
Print "CRLF" on OUT$ device
No
Is I < NDCH ?
Yes
*RTRN
OUT11
Display 2 digits of I on OUT$ device
MESAGE
Print "apostrophy, space" on OUT$ device
Note 8
Determine A, the output number of DUNT(I) instruction
OUT00
Display 2 octal digits of A on OUT$ device
Determine B, the DAC channel number B = DCHN(I)
OUT11
Display 2 digits of B on OUT$ device
MESAGE
Print 2 spaces on OUT$ device
PNAM
Print name whose indexing variable is DIDX(I) on OUT$ device
GSF2
Get SVSF as scale factor for indexing variable DIDX(I)
C$42
Convert DK2(I) and DK2B(I) to floating point as AA
Calculate the EU value at 0 volts output AA = -AA*SVSF / DK1(I)*2^-10
MESAGE
Print 2 spaces on OUT$ device
OUT22
Display AA in floating point on OUT$ device
MESAGE
Print 4 spaces on OUT$ device
Determine A, the scaling power of 2, by decoding shift instruction in DESP(I)
Calculate EU value at 10-volt output BB=AA SVSF*VOLT*10.0 / DK1(I)*2^(A-10)
OUT22
Display BB in floating point on OUT$ device
Add increment to loop counter I = I + 1

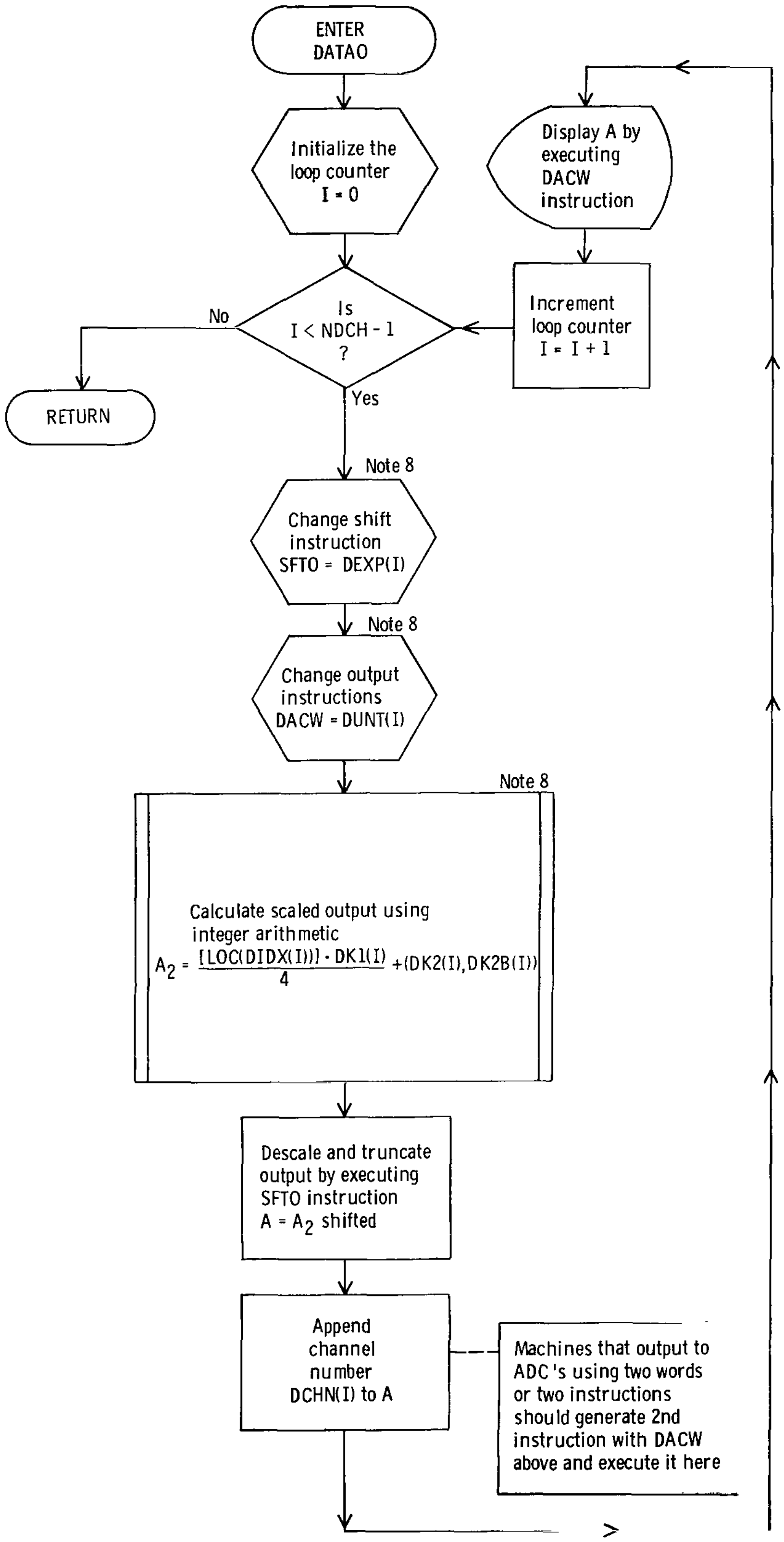

ENTER 01
Is DLM2 comma ?
No
Yes
Nondefaultable operands must follow
EROR
GSF2
Get scale factor SVSF for INDX indexing variable
MATH
Obtain AA, an SFAE from operator. Use EU values for names. DLM2 = delimiter
Operand not defaultable
Was AA defaulted ?
Yes
No
Nondefaultable operand must follow
Is DLM2 comma ?
No
EROR
Yes
MATH
Obtain SFAE BB from operator. Use EU values for names. DLM2 = delimiter
Operand not defaultable
Was BB defaulted ?
Yes
EROR
No
VOLT = DAC count/volt
Determine scaling multiplier
$CC = \frac{SVSF \cdot 10 \cdot VOLT}{BB - AA}$
Select A such that $0.5 < |CC| \cdot 2^{-(A+13)} < 1.0$
Split CC into an integer multiply and a shift. Save the multiply
$TDK1 = CC \cdot 2^{-(A+13)} \cdot 2^{15}$
Shift is too big for a single instruction
Test shift A
$|A| \geq 16$
$A \geq 16$
$|A| < 16$
Note 8
Print "SPOF" on OUT$ device
EROR
Set TEX = a full arithmetic left shift A times instruction if $A \geq 0$, or a full arithmetic right shift A times instruction if $A < 0$
Calculate offset
$BB = \frac{-AA \cdot CC \cdot 2^{-A}}{SVSF}$
The offset is too great
Is $BB < (3/4) \cdot (2^{30})$?
No
C$24
Convert BB to double-precision integer and save in (TDK2, TDKB)
Note 8
Decode B, a Dac unit from a past DUNT(I) instruction
PNUM
Get positive unit number B from operator. Use B for defaults. DLM2 = delimiter
Is B a Dac unit number ?
No
EROR
Yes
Note 8
Set TUN to a write-Dac-unit-B-from-accumulator instruction
PNUM
Get Dac B from operator. Use DCHN(IDXO) for defaults. DLM2 = delimiter
CKCR
Terminate command line
Is B a valid Dac number ?
No
EROR
Yes
Save Dac number DCHN(IDXO) = B
Build DATAO tables:
DEXP(IDXO) = TEX
DK1(IDXO) = TDK1
DK2(IDXO) = TDK2
DK2B(IDXO) = TDKB
DUNT(IDXO) = TUN
RETURN


PCHO
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
PU
Note 7
Complete command and produce binary dump of core from [DA1] to [NCH]
* RTRN
SPL
PRFX
Verify proper prefix command format. Put command delimiter in DLM2
Note 7
Set operands to control DATAO X01 ≡ S1 DA1 ≡ TORG NCH ≡ NSCH CHIX ≡ SIDX
Change number of channel limit NBUF = 80
Set EROR and all command processing routines to jump here when complete
Save program counter for this place in RTRN
Set input IN$ and output OUT$ devices to operator's console
MESAGE
Print "<" and "bell" on OUT$ device
CMD
Note 6
Input and execute SAMPLE control commands
"B"
"?"
SZE
Display SAMPLE channel definitions on OUT$ device
MESAGE
Print "CRLF" and "SAMPLE SIZE" on OUT$ device
OUT11
Display 5 digits of SIZE on OUT$ device
MESAGE
Print "CRLF" and "# AVERAGES =" on OUT$ device
OUT11
Display 5 digits of NAVG on OUT$ device
MESAGE
Print "CRLF" on OUT$ device
Initialize loop counter I = 0
MESAGE
Print "CRLF" on OUT$ device
Is I < NSCH ?
No
Yes
* RTRN
OUT11
Display 2 digits of I on OUT$ device
MESAGE
Print 2 spaces on OUT$ device
PNAM
Print name for indexing variable SIDX(I) on OUT$ device
OPNT
Print TORG(I) in octal on OUT$ device
OPNT
Print (TORG(I) + SIZE) in octal on OUT$ device
Add increment to loop counter I = I + 1

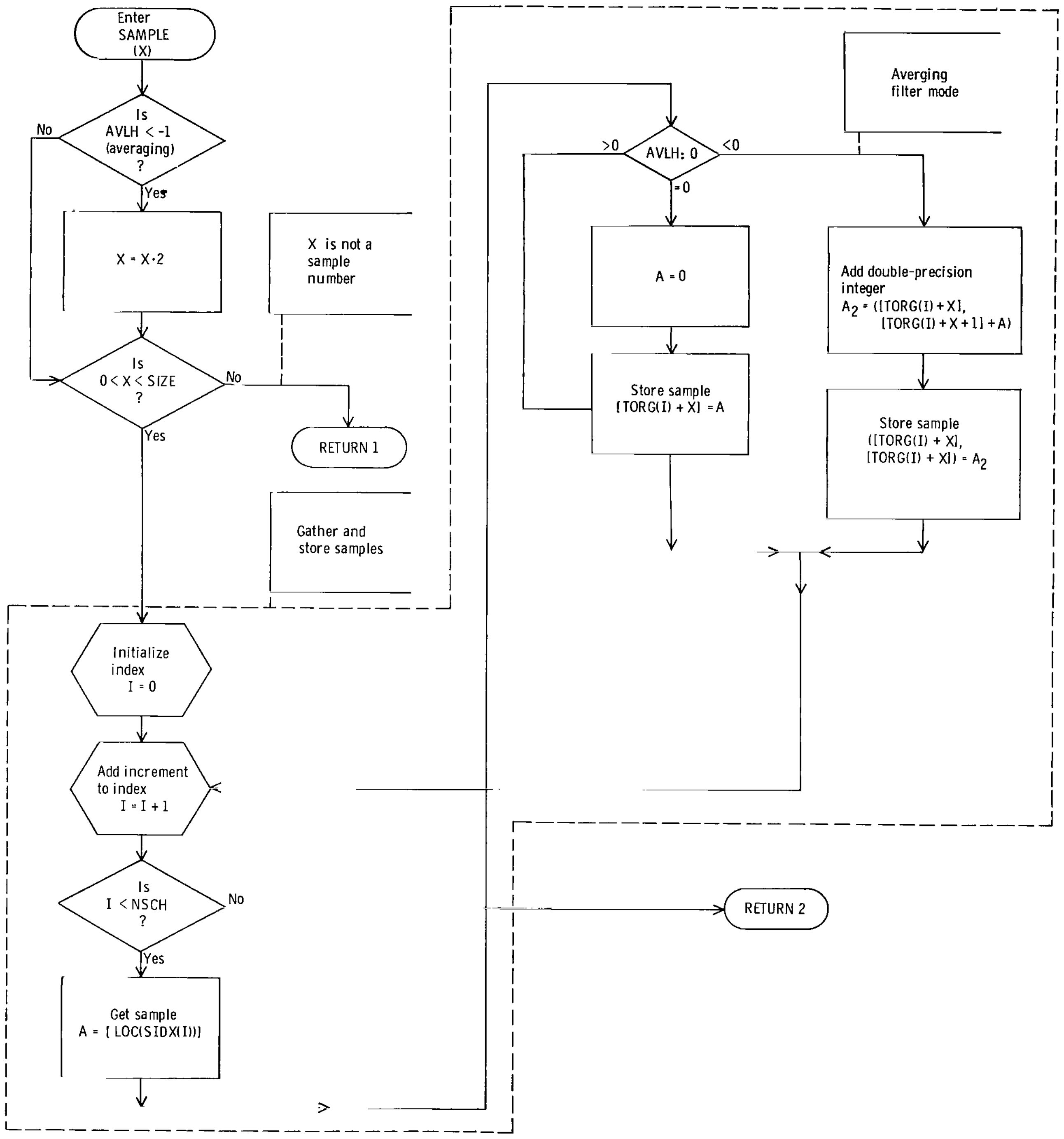

SZE
PRFX
Verify proper prefix command format
MESAGE
Print "CRLF" and "SAMPLE SIZE=" on OUT$ device
Clear default delimiter DLM2 = "comma"
PNUM
Get A, a positive integer, from operator. Use SIZE for defaults. DLM2 = delimiter
CKCR
Validate delimiter; terminate line
MESAGE
Print "CRLF" and "# AVERAGES =" on OUT$ device
Clear default DLM2 = "comma"
PNUM
Get B, a positive integer, from operator. Use SVLH for defaults. DLM2 = delimiter
CKCR
Validate delimiter; terminate line
Limit number of averages B ≥ 1
Save values: If B ≠ 1, AVLH = -B; if B = 1, AVLH = B. SVLH = AVLH SIZE = A NAVG = B
RTRN
Improper format or line cancelled by #
EROR
ENTER S1
Test DLM2
Other
"CR"
"comma"
ADRS
Obtain address B from operator. Use 0 for default. DLM2 = delimiter
Is DLM2 "CR" or "comma" ?
No
Yes
Was address defaulted ?
Yes
No
Save storage block address TORG(IDXO) = B
Calculate first unused core at end of block A = B + SIZE
OPNT
Display A in octal on OUT$ device
MESAGE
Print "CRLF" on OUT$ device
RETURN
Use CMDC
Test command using this routine
"C"
RETURN
"A"
Calculate last channel number A = NSCH - 2
Error: There are no channels to get address from
Is A < 0 ?
Yes
EROR
Supply storage block address B = TORG(A) + SIZE

Flow charts for CLRSMP, DATAO, and SAMPLE follow.


ENTER CLRSMP
Initialize loop counter I = 0
Set AVLH to clear storage blocks AVLH = 0
SAMPLE
Execute SAMPLE (I)
I < block length, samples remain
Add increment to loop counter I = I + 1
I > block length
Return AVLH to proper value AVLH = SVLH
RETURN

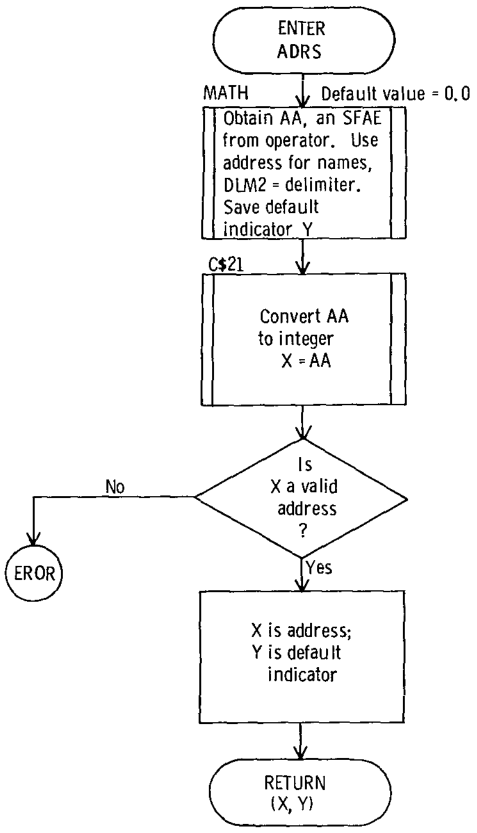


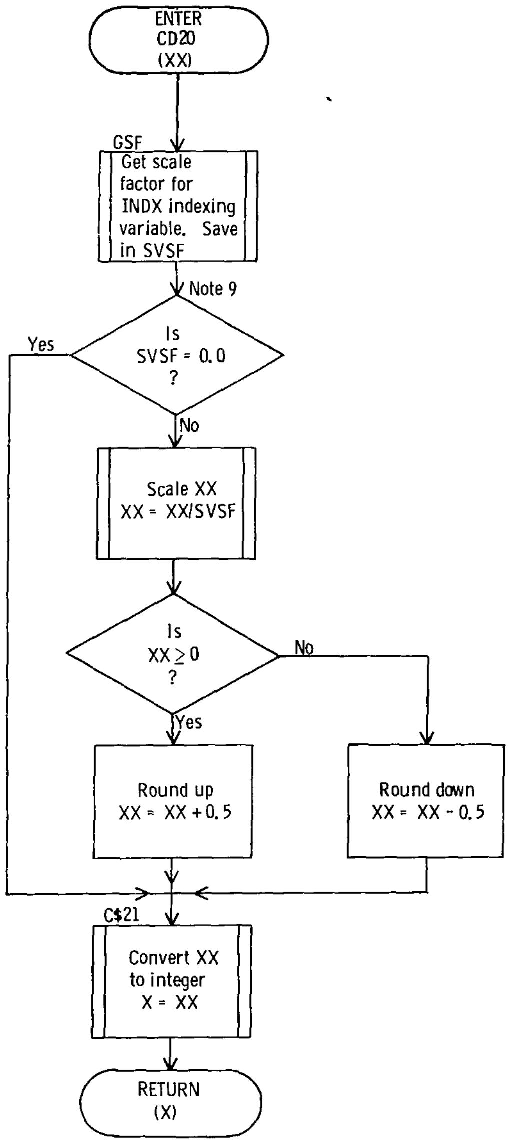
ENTER DATAO
Initialize the loop counter I = 0
Is I < NDCH - 1 ?
No
RETURN
Yes
Note 8
Change shift instruction SFTO = DEXP(I)
Note 8
Change output instructions DACW = DUNT(I)
Note 8
Calculate scaled output using integer arithmetic
A2 = [LOC(DIDX(I))] · DK1(I) / 4 + (DK2(I),DK2B(I))
Descale and truncate output by executing SFTO instruction A = A2 shifted
Append channel number DCHN(I) to A
Machines that output to ADC's using two words or two instructions should generate 2nd instruction with DACW above and execute it here
Display A by executing DACW instruction
Increment loop counter I = I + 1

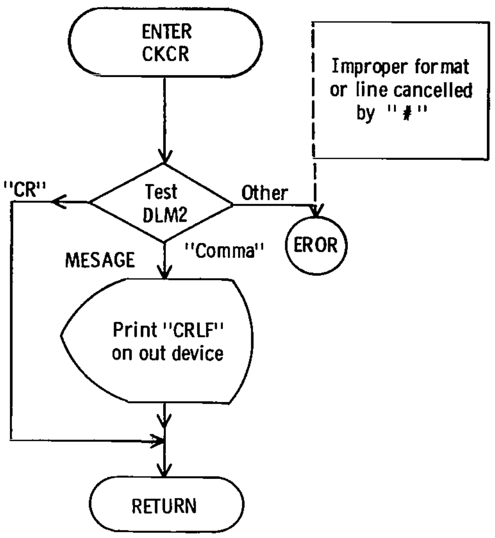


Enter SAMPLE (X)
Is AVLH < -1 (averaging) ?
No
Yes
X = X·2
X is not a sample number
Is 0 < X < SIZE ?
No
Yes
RETURN 1
Gather and store samples
Initialize index I = 0
Add increment to index I = I + 1
Is I < NSCH ?
No
Yes
Get sample A = [LOC(SIDX(I))]
RETURN 2
Averging filter mode
AVLH: 0
>0
<0
=0
A = 0
Store sample [TORG(I) + X] = A
Add double-precision integer A2 = ([TORG(I) + X], [TORG(I) + X + 1] + A)
Store sample ([TORG(I) + X], [TORG(I) + X]) = A2

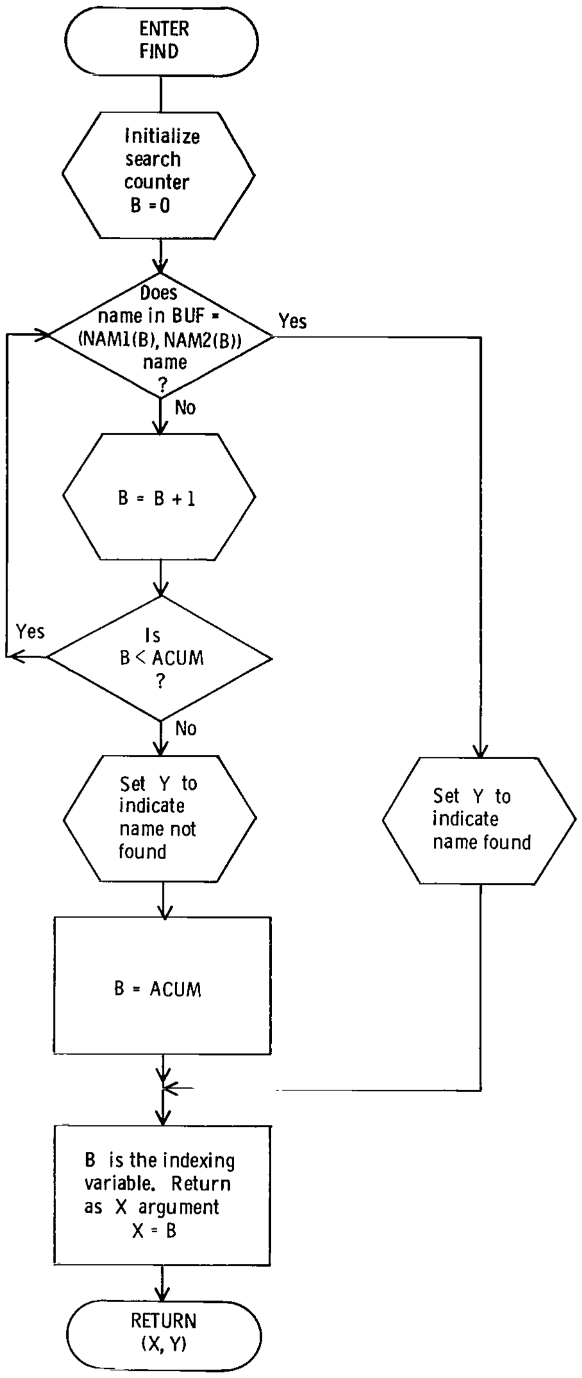
The subroutines used by the command processors follow in alphabetical order.


ENTER ADRS
MATH
Default value = 0.0
Obtain AA, an SFAE from operator. Use address for names, DLM2 = delimiter. Save default indicator Y
C$21
Convert AA to integer X = AA
Is X a valid address ?
No
EROR
Yes
X is address; Y is default indicator
RETURN (X, Y)

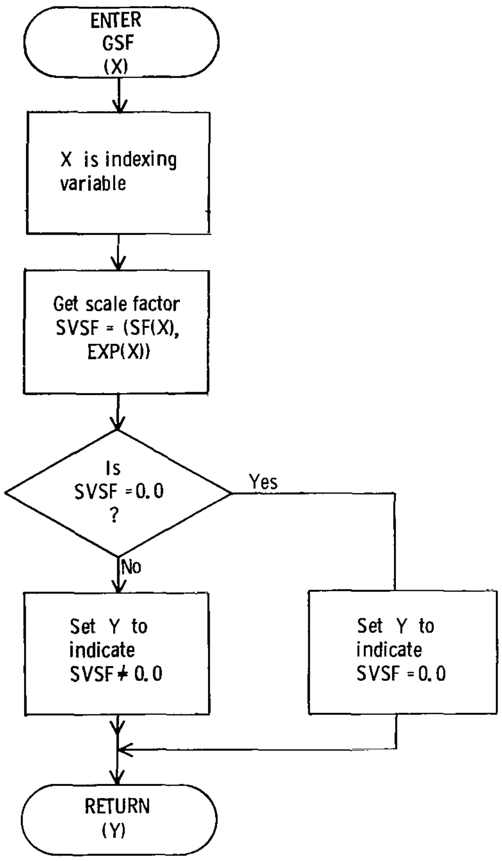


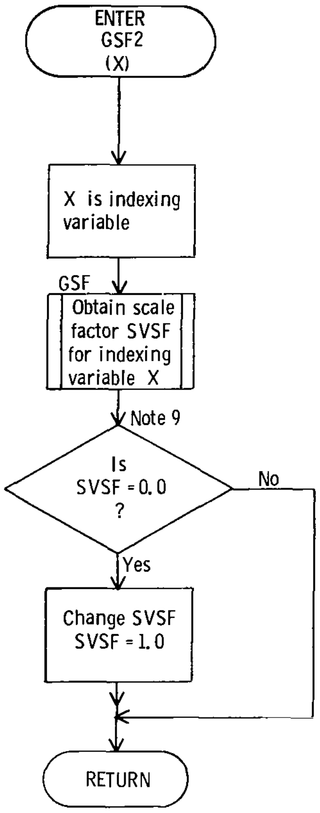


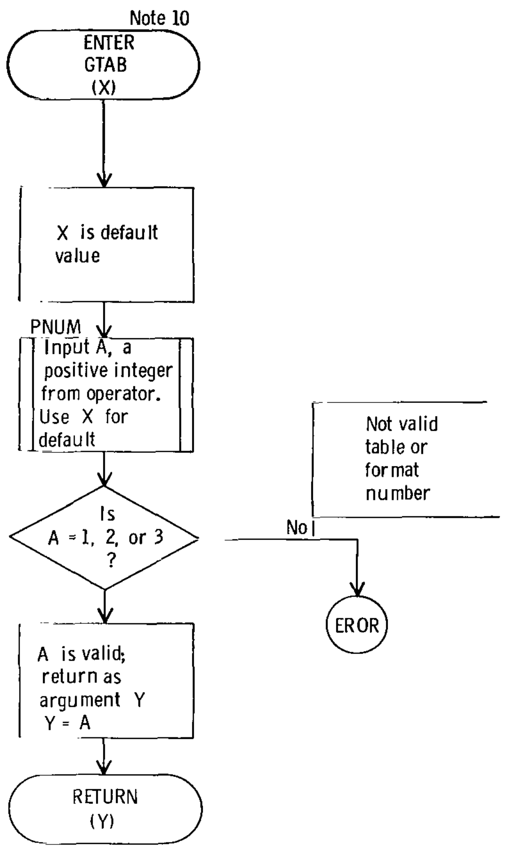
ENTER CKCR
Improper format or line cancelled by "#"
"CR"
Test DLM2
Other
EROR
MESAGE
"Comma"
Print "CRLF" on out device
RETURN

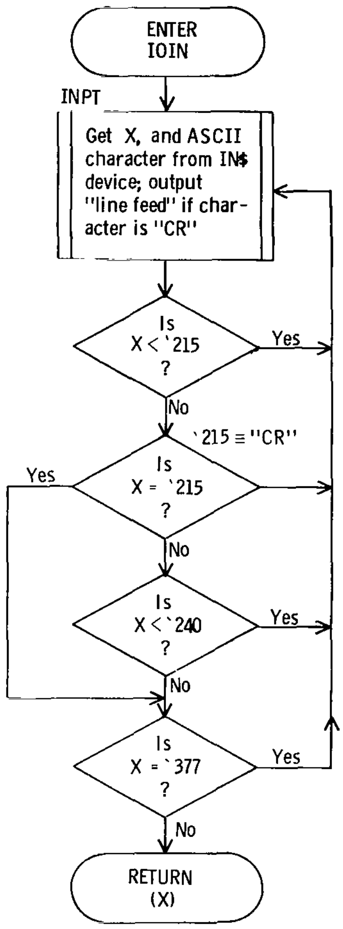


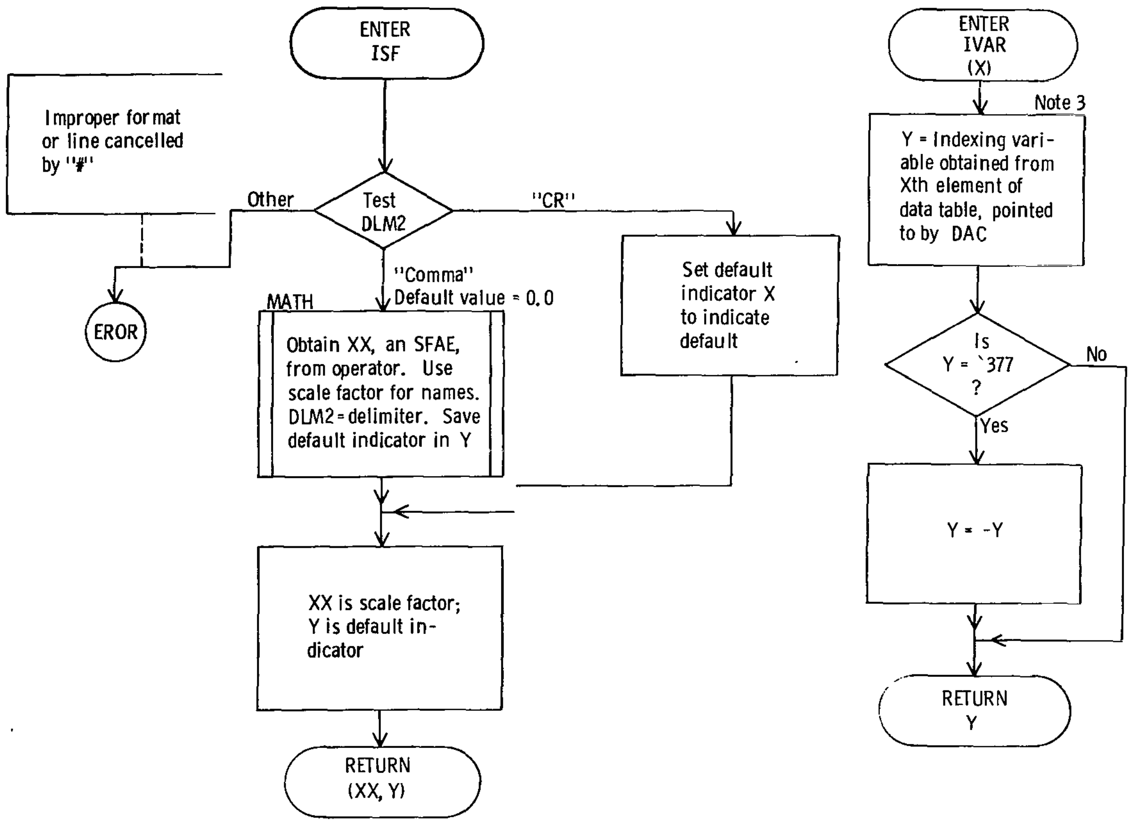
ENTER FIND
Initialize search counter B = 0
Does name in BUF = (NAM1(B), NAM2(B)) name ?
Yes
No
B = B + 1
Yes
Is B < ACUM ?
No
Set Y to indicate name not found
Set Y to indicate name found
B = ACUM
B is the indexing variable. Return as X argument X = B
RETURN (X, Y)


ENTER GSF (X)
X is indexing variable
Get scale factor SVSF = (SF(X), EXP(X))
Is SVSF = 0.0 ?
Yes
No
Set Y to indicate SVSF ≠ 0.0
Set Y to indicate SVSF = 0.0
RETURN (Y)

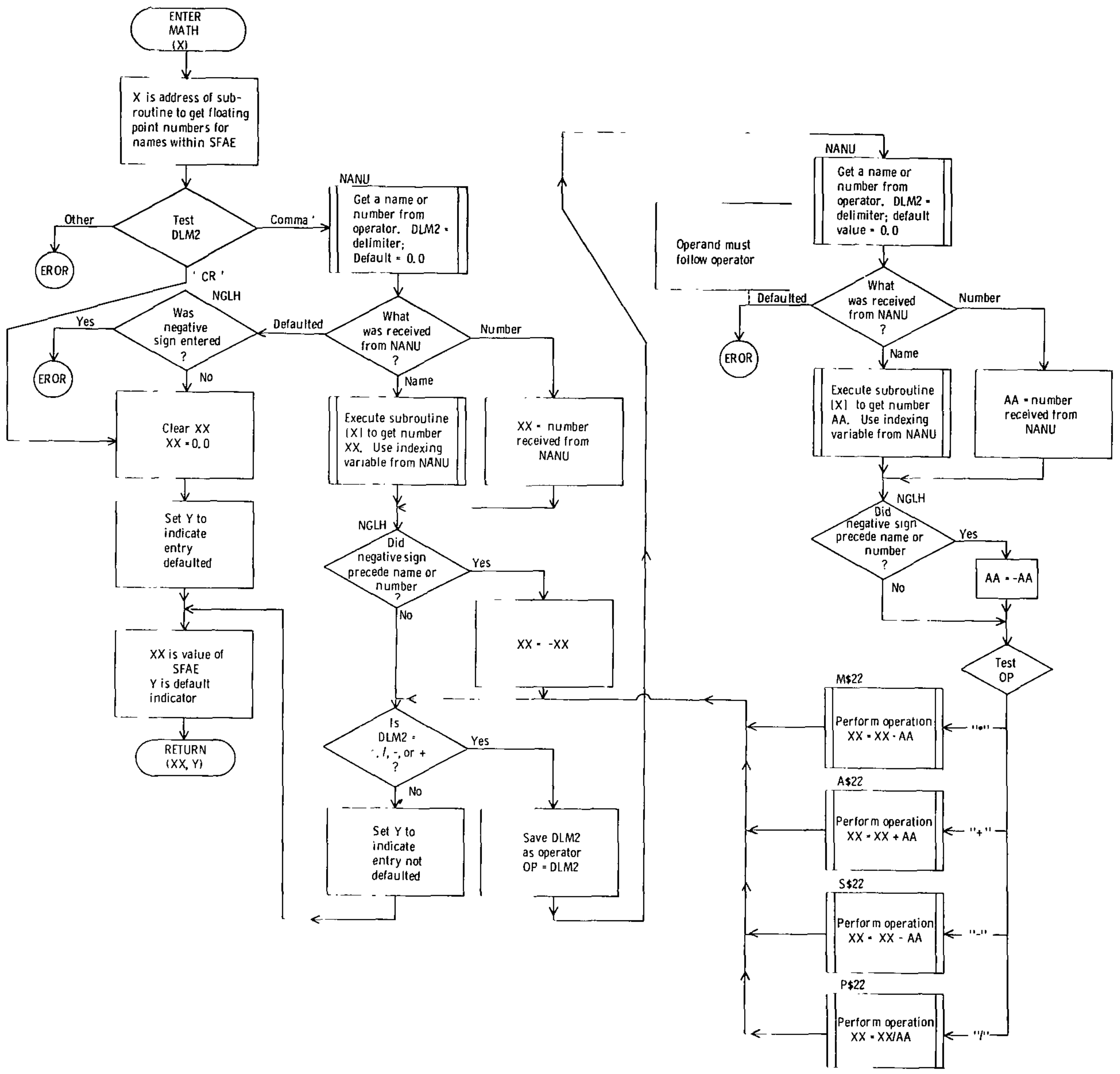

ENTER GSF2 (X)
X is indexing variable
GSF
Obtain scale factor SVSF for indexing variable X
Note 9
Is SVSF = 0.0 ?
No
Yes
Change SVSF SVSF = 1.0
RETURN


Note 10
ENTER GTAB (X)
X is default value
PNUM
Input A, a positive integer from operator. Use X for default
Is A = 1, 2, or 3 ?
No
Not valid table or format number
EROR
A is valid; return as argument Y
Y = A
RETURN (Y)

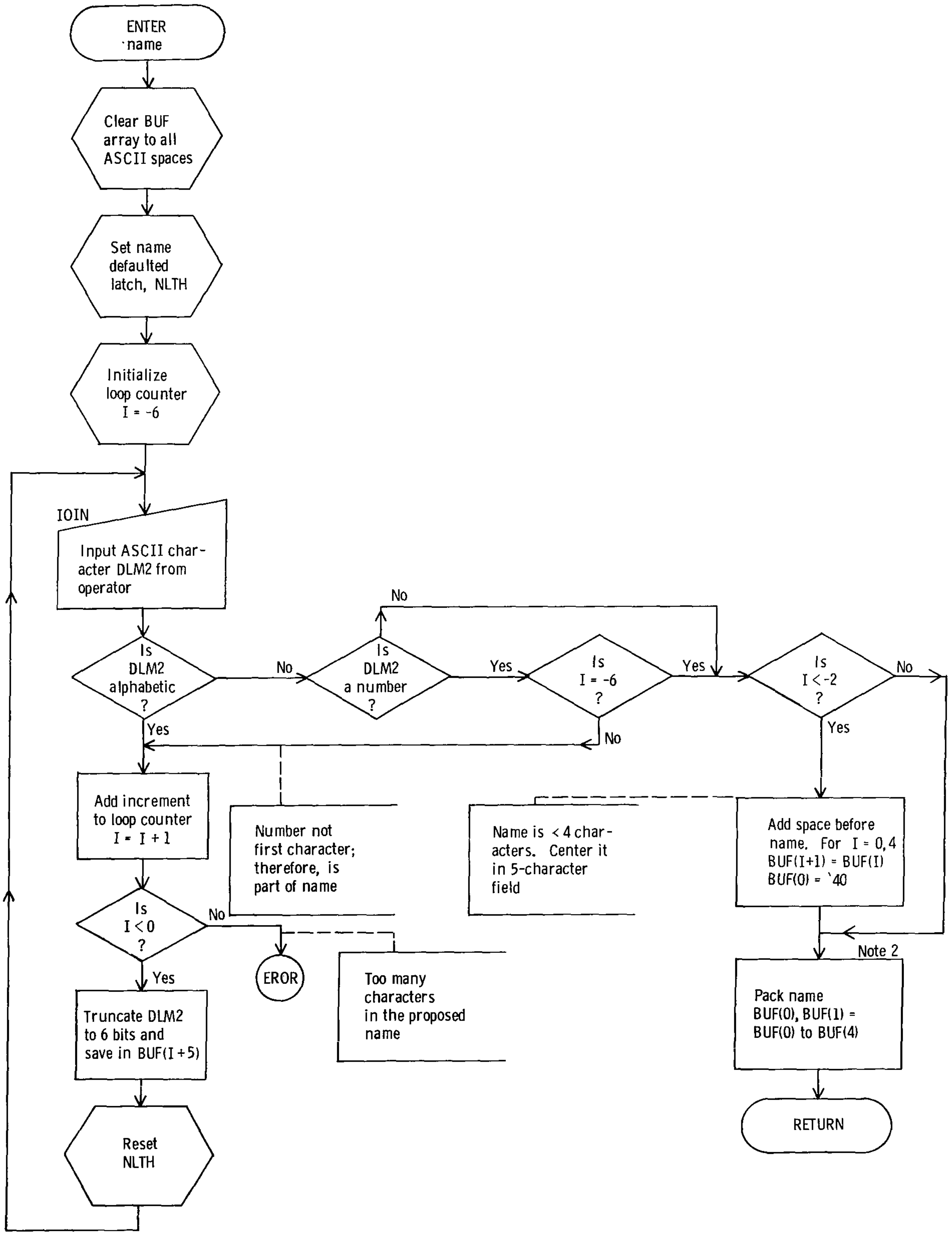

ENTER IOIN
INPT
Get X, and ASCII character from IN$ device; output "line feed" if character is "CR"
Is X < `215 ?
Yes
No
`215 ≡ "CR"
Is X = `215 ?
Yes
No
Is X < `240 ?
Yes
No
Is X = `377 ?
Yes
No
RETURN (X)

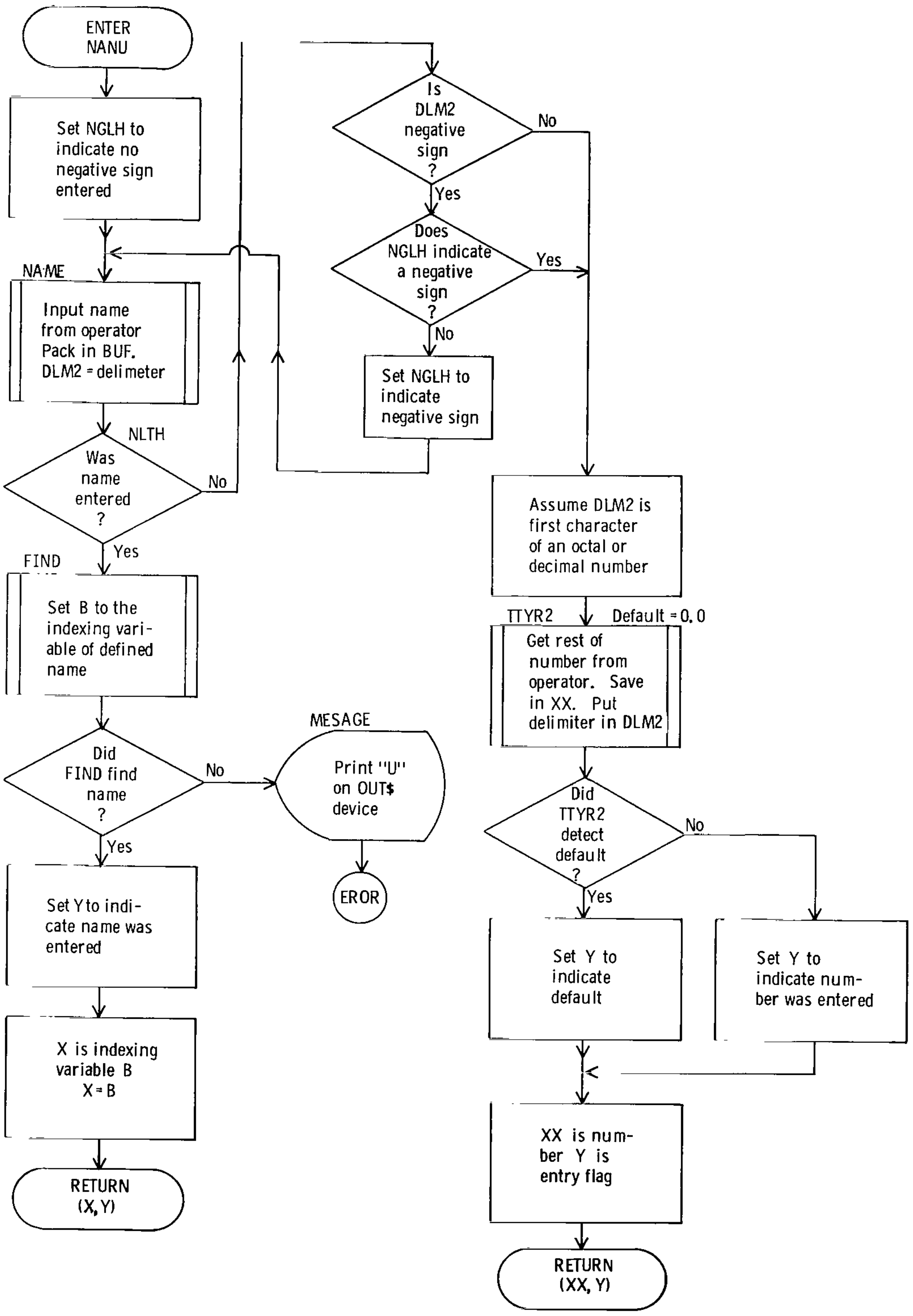


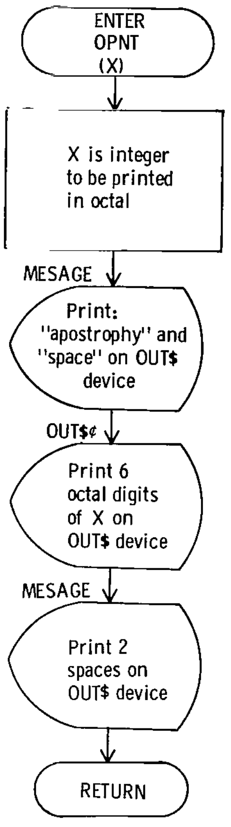
ENTER ISF
Improper format or line cancelled by "#"
Other
Test DLM2
"CR"
EROR
"Comma" Default value = 0.0
MATH
Obtain XX, an SFAE, from operator. Use scale factor for names. DLM2 = delimiter. Save default indicator in Y
Set default indicator X to indicate default
XX is scale factor; Y is default indicator
RETURN (XX, Y)
ENTER IVAR (X)
Note 3
Y = Indexing variable obtained from Xth element of data table, pointed to by DAC
Is Y = `377 ?
No
Yes
Y = -Y
RETURN Y

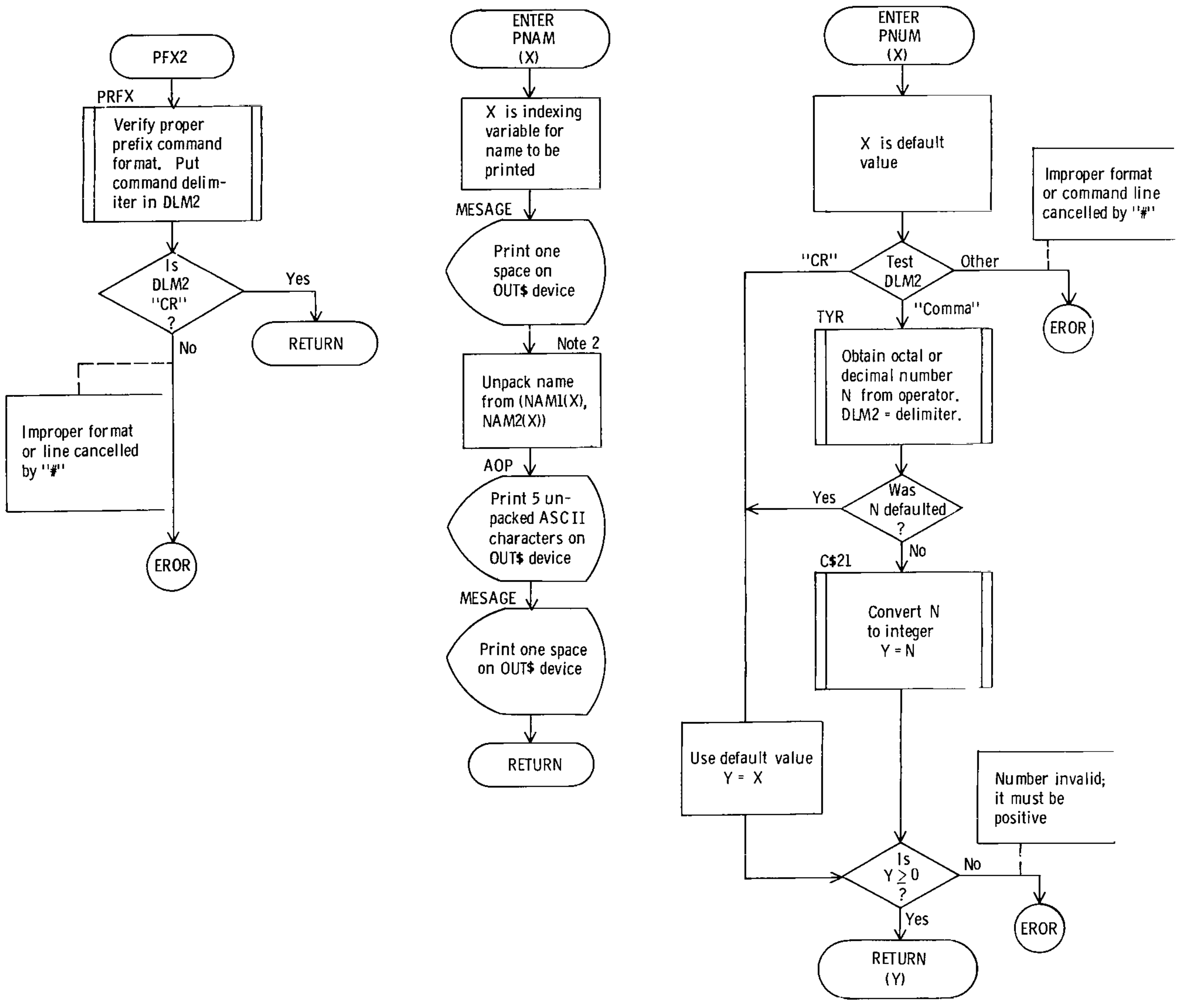


ENTER MATH (X)
X is address of sub-routine to get floating point numbers for names within SFAE
Test DLM2
Other
EROR
Comma ','
' CR '
NANU
Get a name or number from operator. DLM2 = delimiter; Default = 0.0
What was received from NANU ?
Defaulted
Number
Name
NGLH
Was negative sign entered ?
Yes
No
EROR
Clear XX XX = 0.0
Set Y to indicate entry defaulted
XX is value of SFAE Y is default indicator
RETURN (XX, Y)
Execute subroutine (X) to get number XX. Use indexing variable from NANU
XX = number received from NANU
NGLH
Did negative sign precede name or number ?
Yes
No
XX = -XX
Is DLM2 = *, /, -, or + ?
Yes
No
Set Y to indicate entry not defaulted
Save DLM2 as operator OP = DLM2
Operand must follow operator
NANU
Get a name or number from operator. DLM2 = delimiter; default value = 0.0
What was received from NANU ?
Defaulted
Number
Name
EROR
Execute subroutine (X) to get number AA. Use indexing variable from NANU
AA = number received from NANU
NGLH
Did negative sign precede name or number ?
Yes
No
AA = -AA
Test OP
M$22
Perform operation XX = XX * AA
"*"
A$22
Perform operation XX = XX + AA
"+"
S$22
Perform operation XX = XX - AA
"-"
P$22
Perform operation XX = XX/AA
"/"

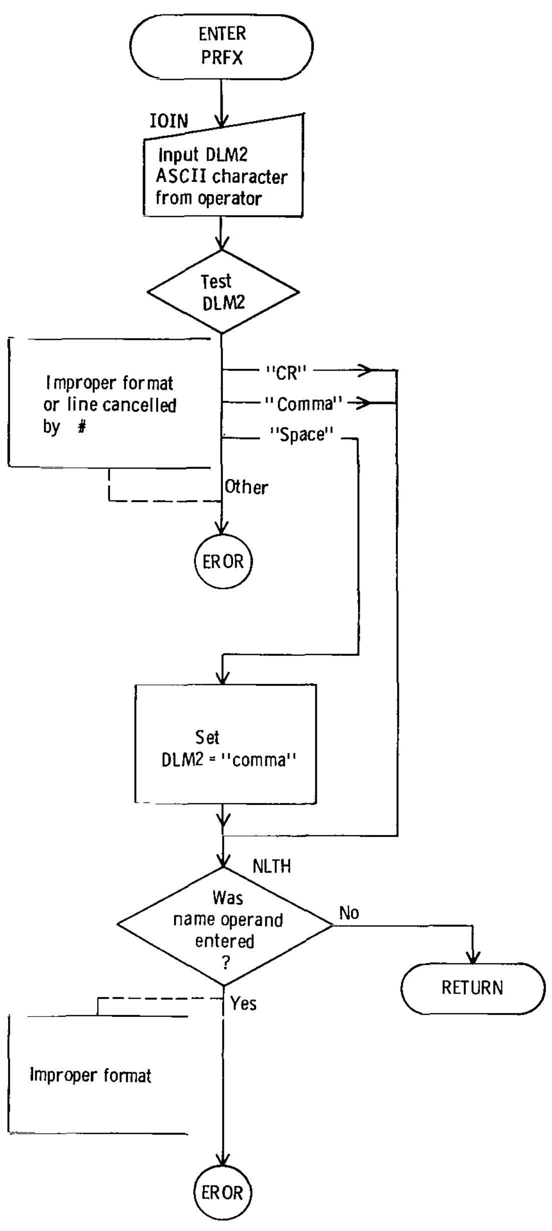

ENTER name
Clear BUF array to all ASCII spaces
Set name defaulted latch, NLTH
Initialize loop counter I = -6
IOIN
Input ASCII character DLM2 from operator
Is DLM2 alphabetic ?
No
Yes
Is DLM2 a number ?
No
Yes
Is I = -6 ?
Yes
No
Is I < -2 ?
No
Yes
Add increment to loop counter I = I + 1
Number not first character; therefore, is part of name
Name is < 4 characters. Center it in 5-character field
Add space before name. For I = 0,4 BUF(I+1) = BUF(I) BUF(0) = '40
Is I < 0 ?
No
Yes
EROR
Too many characters in the proposed name
Note 2
Pack name BUF(0), BUF(1) = BUF(0) to BUF(4)
Truncate DLM2 to 6 bits and save in BUF(I+5)
Reset NLTH
RETURN


ENTER NANU
Set NGLH to indicate no negative sign entered
NAME
Input name from operator Pack in BUF. DLM2 = delimeter
NLTH
Was name entered ?
No
Yes
FIND
Set B to the indexing variable of defined name
Did FIND find name ?
No
Yes
Set Y to indicate name was entered
X is indexing variable B X = B
RETURN (X, Y)
MESAGE
Print "U" on OUT$ device
EROR
Is DLM2 negative sign ?
No
Yes
Does NGLH indicate a negative sign ?
Yes
No
Set NGLH to indicate negative sign
Assume DLM2 is first character of an octal or decimal number
TTYR2
Default = 0, 0
Get rest of number from operator. Save in XX. Put delimiter in DLM2
Did TTYR2 detect default ?
No
Yes
Set Y to indicate default
Set Y to indicate number was entered
XX is number Y is entry flag
RETURN (XX, Y)

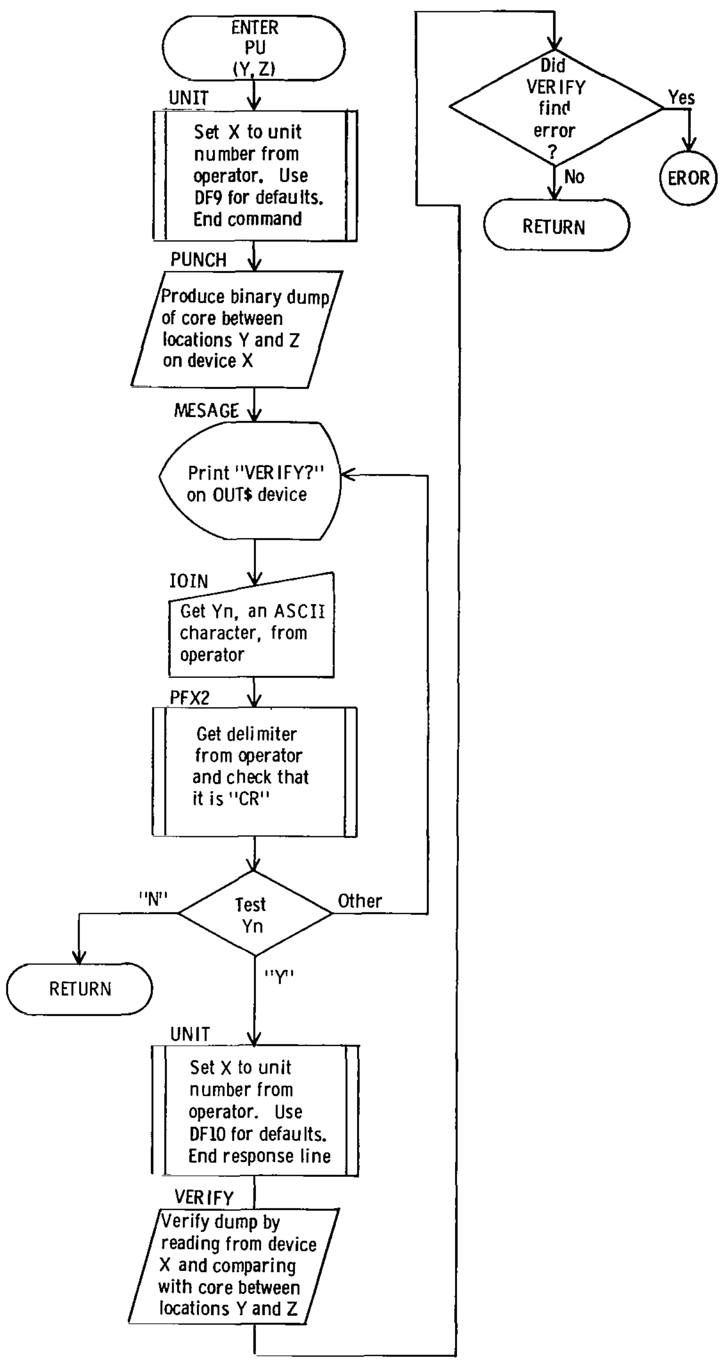

ENTER OPNT (X)
X is integer to be printed in octal
MESAGE
Print: "apostrophy" and "space" on OUT$ device
OUT$¢
Print 6 octal digits of X on OUT$ device
MESAGE
Print 2 spaces on OUT$ device
RETURN



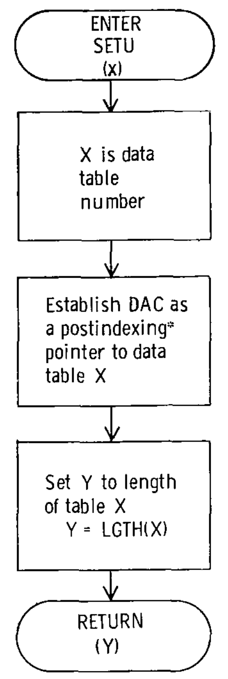
PFX2
PRFX
Verify proper prefix command format. Put command delim-iter in DLM2
Is DLM2 "CR" ?
Yes
RETURN
No
Improper format or line cancelled by "#"
EROR
ENTER PNAM (X)
X is indexing variable for name to be printed
MESAGE
Print one space on OUT$ device
Note 2
Unpack name from (NAM1(X), NAM2(X))
AOP
Print 5 un-packed ASCII characters on OUT$ device
MESAGE
Print one space on OUT$ device
RETURN
ENTER PNUM (X)
X is default value
Improper format or command line cancelled by "#"
"CR"
Test DLM2
Other
EROR
"Comma"
TYR
Obtain octal or decimal number N from operator. DLM2 = delimiter.
Yes
Was N defaulted ?
No
C$21
Convert N to integer Y = N
Use default value Y = X
Number invalid; it must be positive
Is Y ≥ 0 ?
No
EROR
Yes
RETURN (Y)



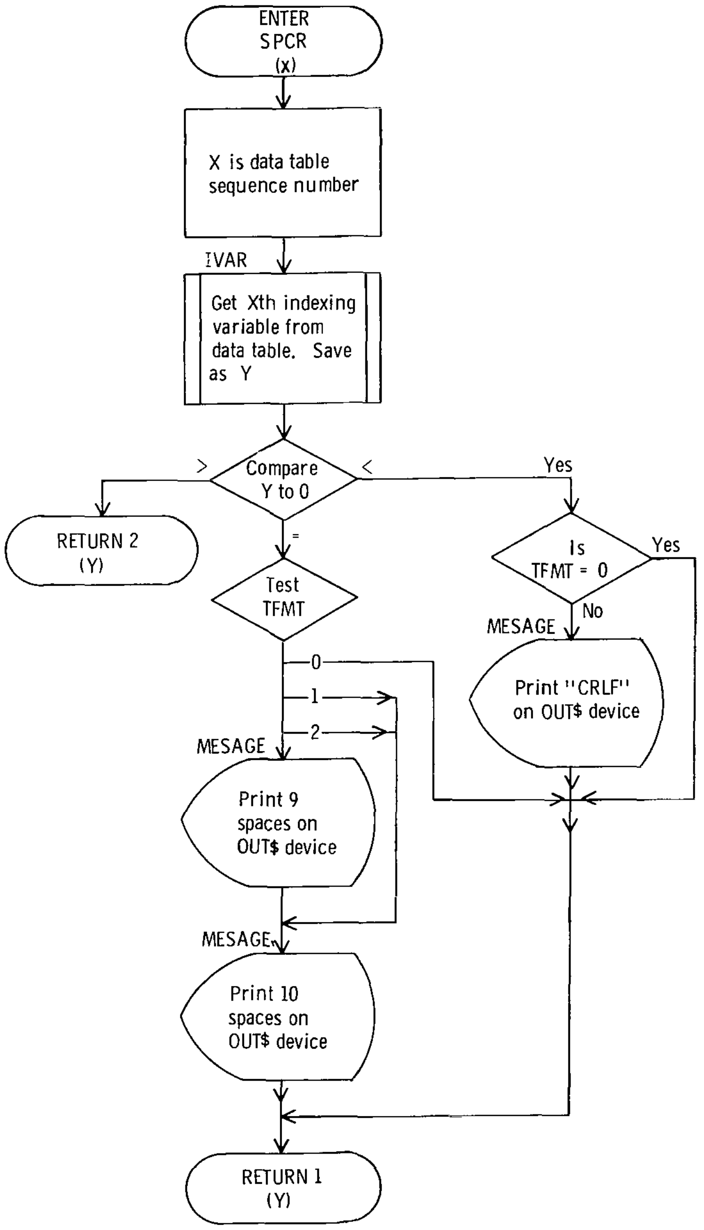
ENTER PRFX
IOIN
Input DLM2 ASCII character from operator
Test DLM2
"CR"
"Comma"
"Space"
Other
Improper format or line cancelled by #
EROR
Set DLM2 = "comma"
NLTH
Was name operand entered ?
No
Yes
RETURN
Improper format
EROR

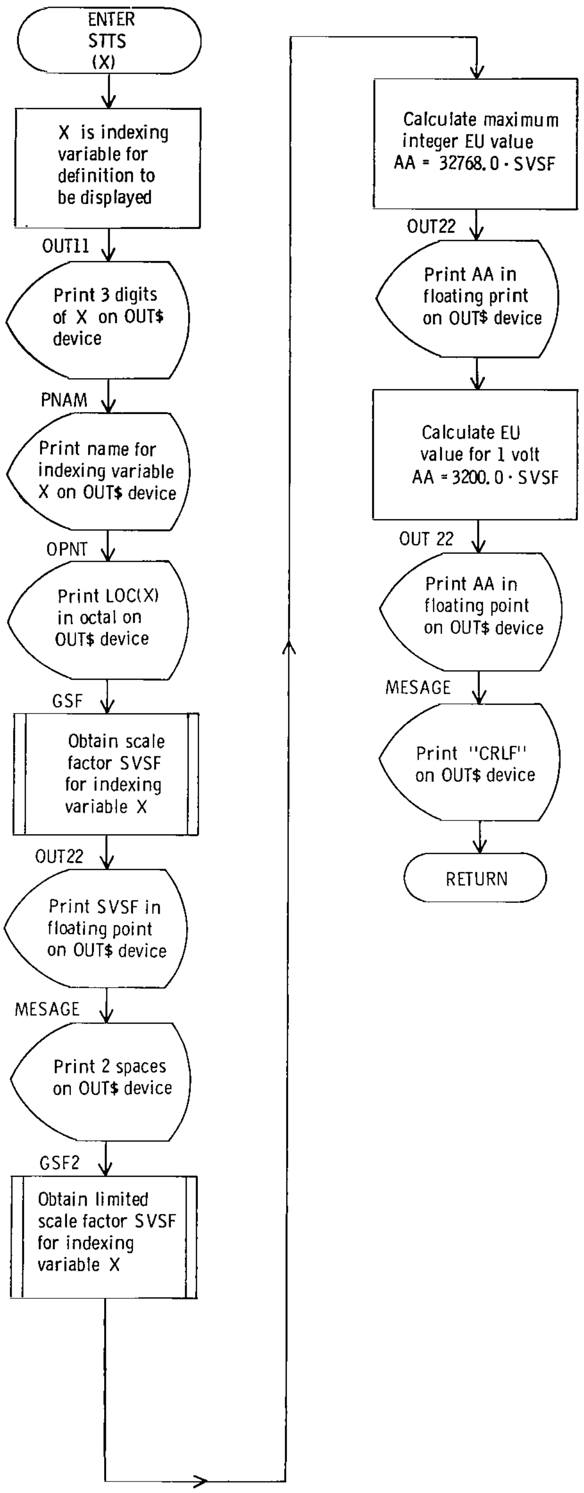
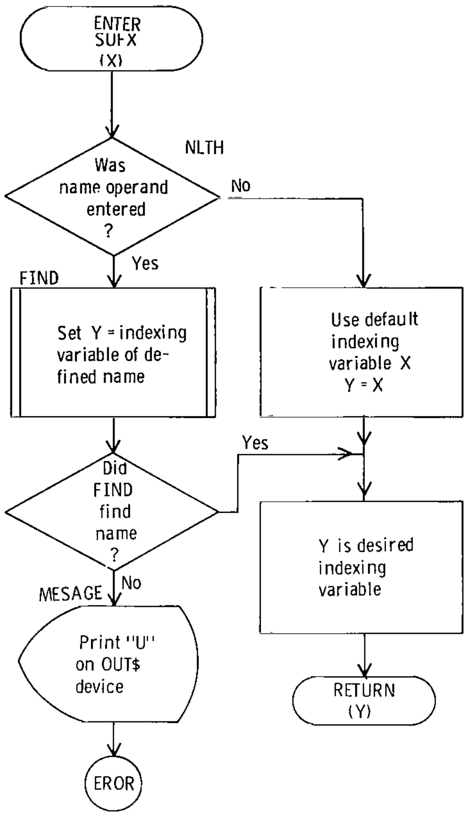


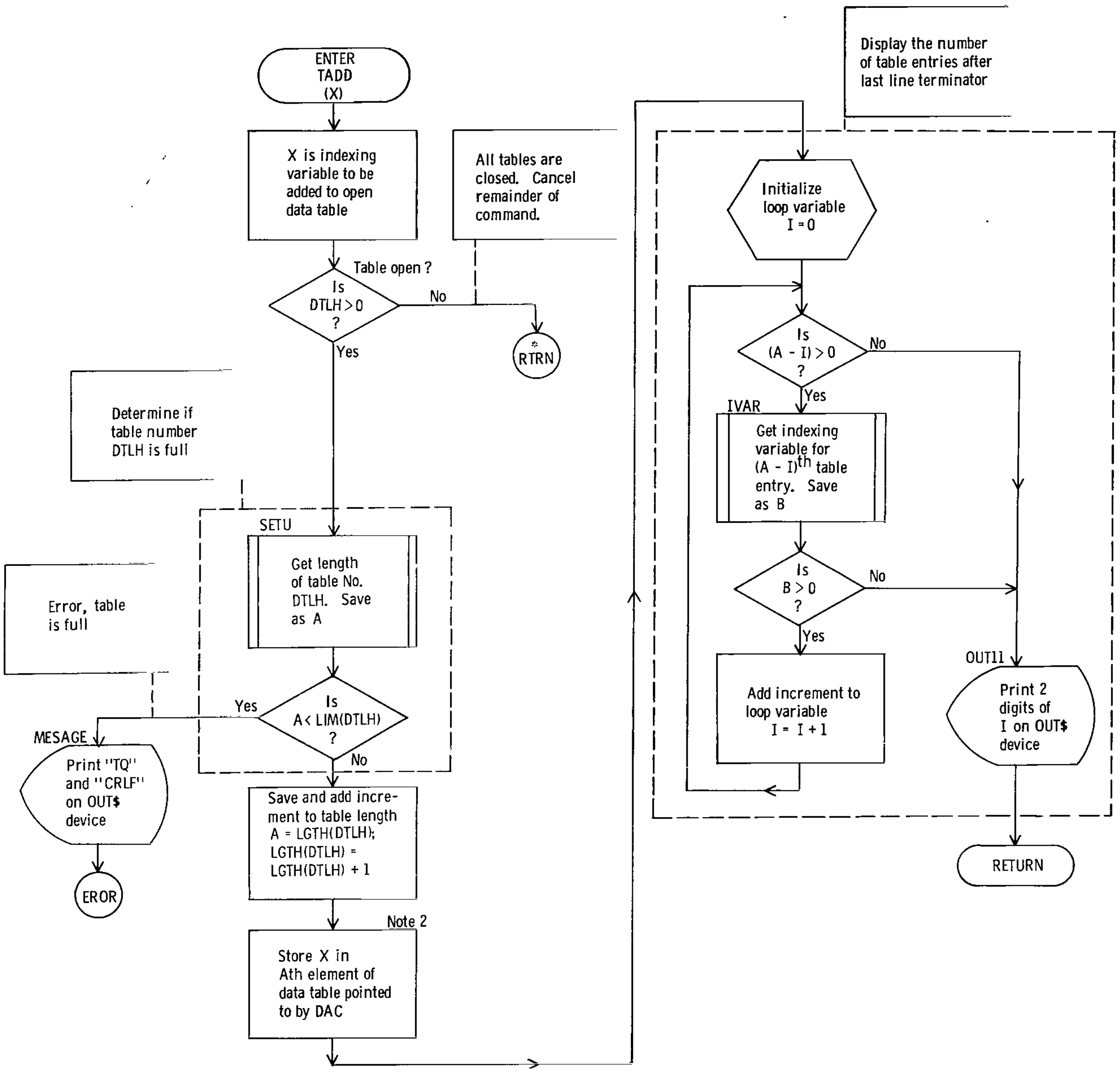
ENTER PU (Y, Z)
UNIT
Set X to unit number from operator. Use DF9 for defaults. End command
PUNCH
Produce binary dump of core between locations Y and Z on device X
MESAGE
Print "VERIFY?" on OUT$ device
IOIN
Get Yn, an ASCII character, from operator
PFX2
Get delimiter from operator and check that it is "CR"
Test Yn
"N"
Other
"Y"
RETURN
UNIT
Set X to unit number from operator. Use DF10 for defaults. End response line
VERIFY
Verify dump by reading from device X and comparing with core between locations Y and Z
Did VERIFY find error ?
Yes
No
EROR
RETURN

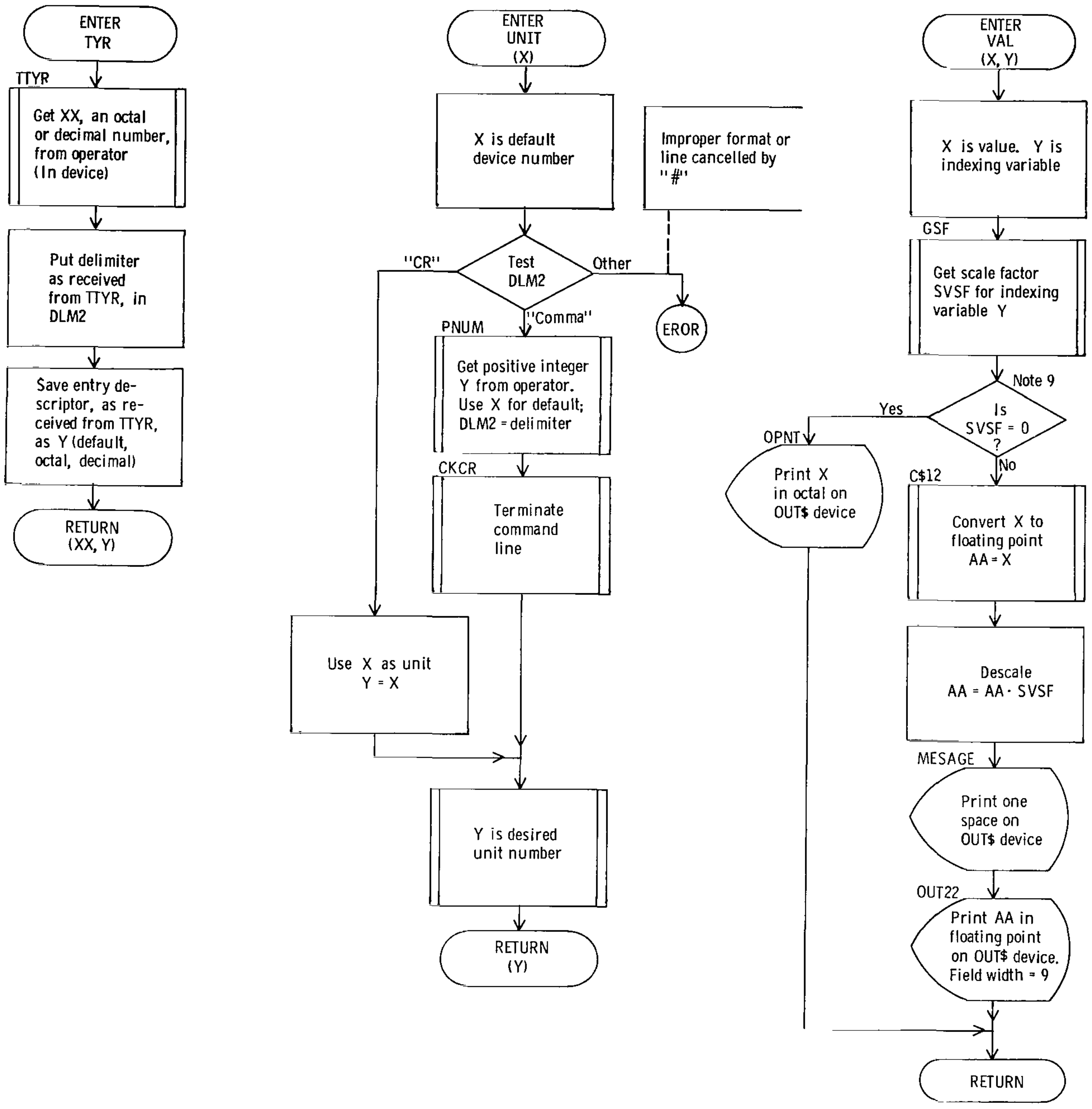

ENTER SETU (x)
X is data table number
Establish DAC as a postindexing* pointer to data table X
Set Y to length of table X Y = LGTH(X)
RETURN (Y)

*Depends upon machines addressing modes and methods


ENTER SPCR (x)
X is data table sequence number
IVAR
Get Xth indexing variable from data table. Save as Y
Compare Y to 0
>
<
=
RETURN 2 (Y)
Yes
Is TFMT = 0
Yes
No
Test TFMT
0
1
2
MESAGE
Print "CRLF" on OUT$ device
MESAGE
Print 9 spaces on OUT$ device
MESAGE
Print 10 spaces on OUT$ device
RETURN 1 (Y)


ENTER STTS (X)
X is indexing variable for definition to be displayed
OUT11
Print 3 digits of X on OUT$ device
PNAM
Print name for indexing variable X on OUT$ device
OPNT
Print LOC(X) in octal on OUT$ device
GSF
Obtain scale factor SVSF for indexing variable X
OUT22
Print SVSF in floating point on OUT$ device
MESAGE
Print 2 spaces on OUT$ device
GSF2
Obtain limited scale factor SVSF for indexing variable X
Calculate maximum integer EU value AA = 32768.0 · SVSF
OUT22
Print AA in floating print on OUT$ device
Calculate EU value for 1 volt AA = 3200.0 · SVSF
OUT 22
Print AA in floating point on OUT$ device
MESAGE
Print "CRLF" on OUT$ device
RETURN

ENTER SUFX (X)
NLTH
Was name operand entered ?
No
Yes
FIND
Set Y = indexing variable of defined name
Use default indexing variable X Y = X
Did FIND find name ?
Yes
No
MESAGE
Print "U" on OUT$ device
EROR
Y is desired indexing variable
RETURN (Y)


ENTER TADD (X)
X is indexing variable to be added to open data table
All tables are closed. Cancel remainder of command.
Table open ?
Is DTLH > 0 ?
No
Yes
RTRN
Determine if table number DTLH is full
SETU
Get length of table No. DTLH. Save as A
Error, table is full
Yes
Is A< LIM(DTLH) ?
No
MESAGE
Print "TQ" and "CRLF" on OUT$ device
EROR
Save and add increment to table length A = LGTH(DTLH); LGTH(DTLH) = LGTH(DTLH) + 1
Note 2
Store X in Ath element of data table pointed to by DAC
Display the number of table entries after last line terminator
Initialize loop variable I = 0
Is (A - I) > 0 ?
No
Yes
IVAR
Get indexing variable for (A - I)th table entry. Save as B
Is B > 0 ?
No
Yes
Add increment to loop variable I = I + 1
OUT11
Print 2 digits of I on OUT$ device
RETURN



ENTER TYR
TTYR
Get XX, an octal or decimal number, from operator (In device)
Put delimiter as received from TTYR, in DLM2
Save entry descriptor, as received from TTYR, as Y (default, octal, decimal)
RETURN (XX, Y)
ENTER UNIT (X)
X is default device number
Improper format or line cancelled by "#"
"CR"
Test DLM2
Other
EROR
"Comma"
PNUM
Get positive integer Y from operator. Use X for default; DLM2 = delimiter
CKCR
Terminate command line
Use X as unit Y = X
Y is desired unit number
RETURN (Y)
ENTER VAL (X, Y)
X is value. Y is indexing variable
GSF
Get scale factor SVSF for indexing variable Y
Note 9
Yes
Is SVSF = 0 ?
No
OPNT
Print X in octal on OUT$ device
C$12
Convert X to floating point AA = X
Descale AA = AA · SVSF
MESAGE
Print one space on OUT$ device
OUT22
Print AA in floating point on OUT$ device. Field width = 9
RETURN

APPENDIX H

SUBROUTINE FUNCTIONS

In the descriptions that follow "external subroutine" refers to subroutines that are not part of the INFORM program, "internal subroutine" refers to subprograms that are included in the assembly of INFORM.

ADRS: Internal routine to input operator-supplied octal or decimal addresses. The address supplied and information indicating default are passed back to the main program as the first and second arguments. The MATH subroutine is used for input; hence, the delimiter is contained in DLM2. ADRS branches to EROR is the address entered is invalid.

AIP: CIPHER I/O library subroutine that inputs a single word from the I/O device whose number is in the IN$ register. The word "input" is passed back in the hardware A accumulator. Any commands or software buffers required by the device to accomplish a single-byte input task are performed. Information is returned in the least significant bits of A. The hardware B and X registers are unaffected. Only input from devices 1 to 5 is accepted, the input being cancelled for other devices. No error messages are issued.

AIPX: External subroutine that inputs a single word from the I/O device whose number is in the hardware X register. The word input is passed back in the hardware A accumulator. Any commands or software buffers required by the device to accomplish a single-byte input task are performed. Information is returned starting with the least significant bits of A. The hardware B and X registers are unaffected. Only input from devices 1 to 5 is accepted, the input being cancelled for other devices. No error messages are issued.

AOP: External subroutine that outputs the hardware A accumulator to the I/O device whose number is in the OUT$ register. Any commands or software buffers required by the device to accomplish a single-byte output task are performed. Only input from devices 1 to 5 is accepted, the output being cancelled for other devices. No error messages are issued. Output is taken from the most significant bits of A. The hardware registers are unaffected.

AOPX: External subroutine that outputs the hardware A accumulator to the I/O device whose number is in the hardware X register. Any commands or software buffers required by the device to accomplish a single-byte output task are performed. Only output from devices 1 to 5 is accepted, the output being cancelled for other devices. No error messages are issued. Output is taken from the most significant bits of A. The hardware registers are unaffected.

A$22: External subroutine to add a floating-point real number in the floating-point accumulator to a real-valued argument from memory. The real-valued result is in the floating-point accumulator.

CD20: Internal subroutine that receives a floating-point number through the floating-point accumulators and scales this number to a machine integer. The result is passed back as a floating-point real number. The indexing variable in INDX is used to locate the scale factor.

CKCR: Internal subroutine used to supply carriage returns for command lines terminated by commas. This routine is used to supply the carriage-return line feeds necessary to format lines on the operator's input consol.

CLRSMP: Subroutine used to clear the storage blocks of the SAMPLE subroutine. Used for initialization when SAMPLE is filtering collected data by using the averaging feature. Its use is optional when SAMPLE is not averaging.

CMD: Internal subroutine that interprets DATAO and SAMPLE control command key characters and branches to the appropriate command processing routine. Two returns to the calling program are implemented: one for ? key characters, and one for B key characters.

C$12: External subroutine that converts the integer value in the hardware A accumulator to a floating-point real value in the floating-point accumulator.

C$21: External subroutine that converts the floating-point real value in the floating-point accumulator to an integer value in the hardware A accumulator.

C$24: External subroutine that converts a double-precision value in the hardware A and B accumulators to a floating-point real value in the floating-point accumulator.

C$42: External subroutine that converts a floating-point real value in the floating-point accumulator to a double-precision integer value in the hardware A and B accumulators.

DATAO: Dynamic data display program described in this report.

D$22: External subroutine to divide the floating-point real value in the floating-point accumulator by the real-valued argument from memory. The result returns in the floating-point accumulator.

FDDEND: CIPHER I/O library subroutine used to terminate a data dump initiated by the FDDH subroutine. It will terminate any incomplete line and produce an additional line representing a checksum for the incomplete line. If FDDEND is called more than once in succession, only the first call will produce output. If the last line of the data dump was complete, no output will ever occur. This subroutine will also re-initiate for subsequent data dumps by executing a portion of FDDH, but it will not print a header line nor change the output format as does the FDDH subroutine. See the "$1" and "$2" command descriptions for typical outputs using these routines. The I/O device whose number is in the OUT$ register is used.

FDDH: CIPHER I/O library subroutine to set the format for and to initialize data dumps. The first argument supplied through the hardware A accumulator is printed as an integer on the first line twice. The number of significant digits appearing is set by the integer value of the second argument. The third argument is an integer indicating the number of data points to be printed on each line preceding the checksum for that line. The fourth argument is a coded binary number setting the format of the data output routines FDD11, FDD22, etc. The least significant six bits indicate the number of digits to be output for integer values printed by the FDD11 subroutine or the number of digits of the mantissa for real values printed by the FDD22 subroutine. The next most significant three bits indicate the number of digits to be output for the exponent of real values. The remaining bits indicate the number of digits appearing in the integer checksum for each line. See the "$1" and "$2" command descriptions for typical outputs using these routines. The I/O device whose number is in the OUT$ register is used.

FDD11: CIPHER I/O library subroutine to output integer values from the hardware A accumulator in an integer format on the dump initiated by the FDDH subroutine. The checksums and line terminators are automatically printed when required as dictated by the last call to FDDH. The I/O device whose number is in the OUT$ register is used. See the $1 and $2 command descriptions for typical outputs using these routines.

FDD12: CIPHER I/O library subroutine with the input of FDD11 and the output of FDD22.

FDD21: CIPHER I/O library subroutine with the input of FDD22 and the output of FDD11.

FDD22: CIPHER I/O library subroutine to output floating-point real values from the floating-point accumulator in a floating-point format on the dump initiated by the FDDH subroutine. The checksums required are automatically printed when required as dictated by the last CALL to FDDH. The I/O device whose number is in the OUT$ register is used. See the $1 and $2 command descriptions for typical outputs using these routines.

FDD44: CIPHER I/O library subroutine similar to FDD11 except the arguments are double-precision integer.

FIND: Internal subroutine used to locate names in the NAM1 and NAM2 tables. The NAM1 and NAM2 tables are searched for the name residing in BUF. A method is also used to indicate to the calling program whether the name was located as a defined name. If found, a normal return occurs, with the indexing variable in the hardware B accumulator. If not found, a different return point is used with the value of ACUM in the hardware B accumulator.

GSF: Internal subroutine to obtain the floating-point real-valued scale factor from the SF table. The indexing variable supplied as the first argument in the hardware B accumulator is used. The scale factor is returned in SVSF. The second argument, returned through the A accumulator, indicates whether the scale factor was zero.

GSF2: Internal subroutine that used GSF to obtain, in SVSF, the floating-point, real-valued scale factor from the SF table. The indexing variable supplied as the argument is used. If the scale factor is zero, it is set to one.

H$22: External subroutine used to store the floating-point real value of the floating-point accumulator in the argument memory location.

GTAB: Internal subroutine used to obtain the data table number or format type numbers from the operator. The routine used PNUM to obtain the number and uses the value supplied as the first argument as a default value. The resulting numerical value is checked before it is passed to the calling program as the second argument. If the value is not 1, 2, or 3, the routine branches to EROR.

INFORM: The subject of this report.

INPT: CIPHER I/O library subroutine that inputs and outputs a single eight-bit character using the I/O devices whose numbers are in the IN$ and OUT$ registers. The character is returned in the least significant eight bits of the hardware A accumulator. If the character was a carriage return, a carriage-return and line feed is turned around to the OUT$ device. The AIP and AOP subroutines are used for I/O.

IVAR: Internal subroutine used to unpack indexing variables from data-table sequences contained in TAB1, TAB2, or TAB3. This subprogram requires that the DAC location be established as a postindexing pointer to either TAB1, TAB2, or TAB3. The sequence number of the data table is supplied through the hardware A accumulator.

LOAD: CIPHER I/O library subroutine used to load relocatable-binary-dump tapes produced by the PUNCH subroutine. The first argument is the I/O device to be used and is passed in the hardware X register. The values read are stored in memory starting at the memory address supplied as the second argument in the hardware A accumulator up to and including the memory address supplied as the third argument in the hardware B accumulator. If a checksum error is detected, a different return point is used, and no further operation takes place. Preceding this return, CK is printed on the device whose number is in the OUT$ register. The arguments supplied through the hardware registers remain intact on program exit. All device operations required for proper input and output are performed.

The AIPX and AOPX subroutines are used for I/O. Note: This subroutine does not open or close device files; thus multiple dumps may be read in succession.

L$22: External subroutine used to load the real-valued argument in memory into the floating-point accumulator.

MATH: Internal subroutine used to input and evaluate a sequentially formed arithmetic expression from the operator. The first argument, returned through the floating-point accumulator, is the value of the expression. The second argument, returned in the hardware A accumulator, is an indicator for default detection. If the SFAE is defaulted, the first argument is set to zero. Any character not part of the SFAE is used as a delimiter to cause return. Its value is held in DLM2. Improperly formatted SFAE's cause a branch to EROR.

MESAGE: External subroutine to print messages on the I/O device whose number is in the IN$ register. The message to be printed is supplied as an argument. The AOP subroutine is used for output. The hardware registers are unaffected.

M$22: External subroutine used to multiply the floating-point real value in the floating-point accumulator by the real-valued argument in memory. The result is in the floating-point accumulator.

NAME: Internal subroutine that gathers a name for a named location from the operator. The I/O device whose number is in the IN$ register is used. The name obtained is packed and stored in BUF. (See text for packing details.) A return occurs if the first character of the name is not alphabetic or when the first nonalphanumeric entry is received. A latch (NLTH) is set if no name character was entered before the branch back character. In this case the name stored in BUF consists of five spaces (the no-name name).

NANU: Internal subroutine used to gather either numerical or name entries from the operator. The routine distinguishes names from numbers and detects defaults. The first argument returned will be an indexing variable in the hardware B accumulator, if a name was entered, or a floating-point real value in the floating-point accumulator, if a number was entered. The second argument returned is an indicator of what type was detected. The routine uses the NAME and TTYR2 subroutines.

N$22: External subroutine used to negate the floating-point real value in the floating-point accumulator.

OPNT: Internal subroutine that prints six octal digits of the integer argument supplied in the hardware A accumulator. This value, which is printed using the OUT00 subroutine, is preceded by one space and one apostrophy. The value is followed by two spaces. The I/O device whose number is in the OUT$ register is used.

OUT00: CIPHER I/O library subroutine that prints octal digits of the integer argument supplied in the hardware A accumulator. The number of digits printed is supplied as an integer argument and must be a value between 1 and 6. If the number being printed has a significant digit beyond the number of digits requested, an asterisk is printed for all digits. The hardware A, B, and X registers are unaffected. All I/O is done using the AOP subroutine.

OUT11: CIPHER I/O library subroutine that prints decimal digits of the integer argument supplied in the hardware A accumulator on the I/O device whose number is in the OUT$ register. The number of digits printed is supplied as an integer argument. If the number being printed has a significant digit beyond the number of digits requested, and asterisk is printed for all digits. The hardware registers are unaffected. All I/O is done using the AOP subroutine.

OUT22, OUT12: CIPHER I/O library subroutine that prints the floating-point real value (integer value for OUT12) in the floating-point accumulator (hardware A accumulator for OUT12) on the I/O device whose number is in the IN$ register. The format is determined by a coded integer argument. If the number being printed has a significant digit beyond the number of digits requested, an asterisk is printed for all digits. The floating-point accumulator and hardware registers are unaffected. All I/O is done using the AOP subroutine. Double-precision integer values of the mantissa and exponent printed are returned as arguments in common core.

O1: Internal subroutine that represents the unique portion of the ADDO sequence which adds a channel definition to DATAO. All required operands except the name operand are input from the operator. Entires are made into the DK1, DK2, DK2B, DCHN, DUNT, and DEXP matrices. The position used is equal to the integer index of IDXO. Inputs are accomplished using various I/O subroutines and occur on the device whose number is in the IN$ register.

PFX2: Internal subroutine used for checking the operator's command line format when the command has no operands. If the operator enters an operand, the routine branches to EROR.

PNAM: Internal subroutine to print a name of a named location from the NAM1 and NAM2 tables. The indexing variable to be used is supplied through the hardware B accumulator.

PNUM: Internal subroutine to input octal or decimal numbers from the operator, convert them to integer values, and verify that they are positive. The number to be used if the operator defaults is supplied by the calling program as the first argument. The effective input is returned as the second argument. Before this argument is returned, it is tested to see if it is a positive integer. If it is not, the command in progress is cancelled by a branch to EROR. Hence, operator defaults are excluded if the default supplied as the first subroutine argument is negative. The TYR subroutine is used to input the numbers, and the delimiter that caused the return is saved in DLM2. PNUM will exit to EROR if this delimiter is neither a carriage return or comma.

PRFX: Internal subroutine to verify that no name was entered before the command keyword. It is used by commands not using a name operand. A branch to EROR occurs if a name was entered.

PU: Internal subroutine to produce relocatable binary-dump tapes of the core locations between and including the addresses of the arguments supplied. This routine completes the command line by receiving the I/O device operator from the operator. Provision is also made for the operator to verify the dump produced if desired. The PUNCH and VERIFY subroutines are used to produce and verify the dump.

PUNCH: CIPHER I/O library subroutine to produce relocatable binary-dump tapes of the core locations between and including the addresses of the first aid and second arguments supplied through the hardware A and B accumulators. The I/O device number supplied as the third argument in the hardware X register is used. The AOPX subroutine is used for output. Note: This subroutine does not open or close device files, so that multiple dumps may be produced in succession. The arguments supplied through hardware registers remain intact on program exit.

SAMPLE: Dynamic data collection program described in this report.

SETU: Internal subroutine that establishes DAC as a postindexing pointer to the data table number supplied in the hardware B accumulator. Also sets STOP equal to the length of the data table and returns this value in the hardware A accumulator.

SPCR: Internal subroutines used to obtain an indexing variable for a data table. The indexing variable is tested to see if it equals the blank space or line terminating codes. If it does. the proper number of spaces to produce a blank in the table or a carriage-return line feed is printed. Two return locations from the subroutines are used: one to indicate that a blank or "CRLF" was printed, and one to indicate nothing was printed.

STTS: Internal subroutine used to print the definition of a named location on the output device whose number is in OUT$.

SUFX: Internal subroutine used to obtain the indexing variable of named-location operands for commands that use and permit a default of the name operand. The indexing variable to be used for defaults must be supplied as the first argument. The effective indexing variable as entered or defaulted is returned as the second argument. If an undefined name is entered, the routine branches to EROR.

S1: Internal subroutine that represents the unique portion of the ADDO sequence that adds a channel definition to SAMPLE. All required operands except the name operand are input from the operator. Entries are made into the TORG matrix at the position indicated by the IDXO index.

S$22: External subroutine that subtracts the real-valued operand from the floating-point real value in the floating-point accumulator. The result resides in the floating-point accumulator.

TADD: Internal subroutine that adds the indexing variable supplied through the hardware B accumulator to the open data-table sequence array, TAB1, TAB2, or TAB3. The command in progress is cancelled by a branch to EROR if the length permitted for the open table is exceeded. If no data table is open, no action is taken; but the instruction in effect is cancelled by a jump to the address contained in RTRN.

TTYR: CIPHER I/O library subroutine used to input octal or decimal numbers from the input device whose number is in the IN$ register. A return from the subroutine occurs whenever a nonnumeric character, except +, -, E, ', S, B, or rubout, is entered. On return, the fourth argument contains the character that caused the return, and the third argument contains a value to describe which type of input format was used. A value greater than zero indicates a floating-point format, a value less than zero indicates an octal format, and a value of zero implies the number was defaulted. The numerical value or second argument is zero for defaulted numbers. The number entered is returned in floating-point real format in the floating-point accumulator as is also stored in a location supplied as the first argument. The AIP subroutine is used for all inputs.

TTYR2: Additional entry point to TTYR used when the first character to be supplied from the input device has already been input and is received as an argument instead.

TYR: Internal subroutine that calls the TTYR subroutine and puts the delimiter passed as the fourth argument from TTYR in DLM2. The format type supplied by TTYR as the third argument is retained and passed as the second argument. The number input by TTYR is passed as the first argument in the floating-point accumulator and locations N.

UNIT: Internal subroutine through which all unit number command operands are gathered. The first argument passed to the subroutine is the value to be used for defaults. The effective unit number determined by operator entry or default is passed back as the second argument. The delimiter is saved in DLM2. This routine also calls CKCR to terminate a command input line.

VAL: Internal subroutine used to descale and print the value of the integer operand supplied as the first argument. The indexing variable supplied as the second argument is used to locate the scale factor to be used. The MESAGE and OUT22 subroutines are used to print the calculated value in standard floating-point format preceded by one space.

VERIFY: CIPHER I/O library subroutine that reads a relocatable binary-dump tape produced by the PUNCH subroutine. The tape is read from the I/O device number supplied as the first argument in the hardware X register and compared with the values in memory. The starting and ending addresses are supplied as the second and third arguments. If no error is detected, return is to return point 2. If an error is detected in either the memory comparison or checksum validation, a return to point 1 occurs and no further verification takes place. Preceding this return, "CK" is printed on the I/O device whose number is in the OUT$ register. The hardware A, B, and X registers are unaffected. All input and output are done through the AOP and AIPX subroutines. Note: This subroutine does not open or close the device files so that multiple dumps may be read in succession.

XO1: Equivalent to S1 or O1. (See appendix G, flow chart note 6.)

APPENDIX I

SUMMARY OF COMMANDS

INFORM COMMANDS[a]

Page	Description	Prefix operand	Key character	Additional operands	Default values and notes
				Mode control	
69	Enable variable trip mode	None	[	{Na}x{SFAE} HLT? Yn	x is ">", "=", or "<" sign Na - Default = last used name operand SFAE - Default = 0.0, current EU value used for names Yn is supplied as response to HLT? query
70	Reenable previous trip mode	None	[	None	Reenables previous trip condition as established by last enable command
70	Disable variable trip mode	None	]	None	--
70	Accept DATAO control commands	None	>	None	--
71	Accept SAMPLE control commands	None	<	None	--
71	Exit subroutine INFORM	None	.	None	--
				Data table manipulation	
71	Clear and open data table	None	@	{Tn}	Tn - Default = table 1
71	Open data table	None	(	{Tn}	Tn - Default = table 1
72	Close an open data table	None	)	None	--
72	Add name to open data table	{Na}	,	None	Na - Default = blank space added to table
72	Add name and/or start new line on open data table	{Na}	-	None	Na - Default = no name added before new line
73	Delete last line from open data table	None	↑	None	--
73	Print data table	{Na}	\	{DF3}, {DF4}, {DF5}	DF3 - Default - format type 1 DF4 - Default - data table 1 DF5 - Default line printer (unit 5)
				Miscellaneous display	
74	Display current EU value	{Na}	=	None	Na - Default = last used name operand
74	Calculate and display address	None	:0	SFAE	Address used for names within SFAE
74	Calculate and display scale factor	None	:1	SFAE	Scale factor used for names within SFAE
75	Calculate and display EU	None	:2	SFAE	Current EU value used for names within SFAE
75	Calculate and display octal words of floating-point-real EU value	None	:3	SFAE	Current EU value used for names within SFAE
75	List core	None	/	{Ad1}, {Ad2}, {x}, {Sf}, {DF11}	Ad1 - Default = last Ad1 starting address Ad2 - Default = last Ad2 end address if and only if Ad1 defaulted = Ad1 if Ad1 not defaulted x - Default = last x data type Sf - Default = last Sf scale factor DF11 - Default = operator's consol (unit 1)

INFORM COMMANDS[a] (Continued.)

Page		Prefix operand	Key character	Additional operands	Default values and notes
			Data storage		
76	Scale and store integer	{Na}	"	{SFAE}	Na - Default = last used name operand SFAE - Default = 0.0. Current EU value used for names Note: Machine messages appear before and after the SFAE
			Definition control		
77	Define a named location	{Na}	"space"	{Ad}, {Sf}	Na - Default = no-name parameter Ad - Default = last name operand's address + 1 Sf - Default = last name operand's scale factor
78	Redefine a named location	Na	"space"	{xxxxx}, {Ad}, {Sf}	xxxxx = New name, Default = old name (Na) Ad - Default = currently defined address Sf - Default = currently defined scale factor Note: Na must be currently defined
79	Change a default value	None	%	DFn, {No}	Defaulting No → display current value
79	Display all INFORM definitions	None	?	{DF12}	DF12 - Default = line printer (unit 5)
80	Display a single INFORM definition	None	&	{Na}	Na - Default = no-name parameter

80	Save INFORM definitions	None	!	{DF9}
				VERIFY? Yn, {DF10}
					DF9 - Default = paper tape punch (unit 2)
					DF10 - Default = paper tape reader (unit 2)
					Note: Yn and DF10 supplied as response to VERIFY query

81	Restore INFORM definitions	None	*	{DF10}	DF10 - Default = paper tape reader (unit 2)
			Display or dump data collected by SAMPLE		
82	Output SAMPLE channels in format type 1	None	$1	{DF6}, {DF7}, {DF8}	DF6 = Start channel numbers, default = 0 DF7 = Stop channel number, default = last channel defined DF8 = Default = floppy disk (unit 3)
84	Output SAMPLE channels in format type 2	None	$2	{DF6}, {DF7}, {DF8}	DF6 = Start channel number, default = 0 DF7 = Stop channel number, default = last channel defined DF8 = Default = floppy disk (unit 3)

[a]Passive mode data table printout defaults:
DF0 - Format type - default = 1
DF1 - Data table number - Default = 1
DF2 - Output unit number default = line printer (unit 5)

DATAO CONTROL COMMANDS

Page	Description	Key character	Additional operands	Default values and notes
87	Add a new channel definition	A	{Na}, SFAE, SFAE, {Dn}, {Dnn}	Na - Default value = last used name operand Dn - Default = unit number from previous definition of this channel Dnn - Default = Dac number from previous definition of this channel
88	Change a channel definition	C	xx, {Na}, SFAE, SFAE, {Dn}, {Dnn}	xx = Channel number Na - Default = current name defined for this channel Dn - Default = current unit number defined for this channel Dnn - Default = current Dac number defined for this channel
89	Delete the highest numbered channel	D	None	Does not affect defaults for "A" and "C" commands
89	Delete all channel definitions	@	None	Does not affect defaults for "A" and "C" commands
89	Display all channel definitions	?	{DF12}	DF12 - Default line printer (unit 5)

89	Save channel definitions	!	{DF9}
			VERIFY? Yn, {DF10}
				DF9 - Default paper tape punch
				DF10 - Default paper tape reader
				Note: Yn and DF10 supplied in response to VERIFY? query

90	Restore channel definitions	*	{DF10}	DF10 - Default · paper tape reader (unit 2)
91	Accept INFORM commands	-	None	--

SAMPLE CONTROL COMMANDS

Page	Description	Key character	Additional operands	Default values and notes
92	Add a new channel definition	A	{Na}, {Ad}	Na - Default - last used name operand Ad - Default address of highest numbered channel defined plus current block length
93	Change a channel definition	C	xx, {Na}, {Ad}	xx = Channel number Na - Default - current name defined for this channel Ad - Default current address defined for this channel
93	Delete last channel definition	D	None	--
93	Delete all channel definitions	@	None	--
94	Display all channel definitions	?	{DF12}	DF12 - Default = line printer (unit 5)

94	Define block length	B	BLOCK SIZE = {No}
			#AVERAGES = {No}
				No - Default = current block size, initially = 0
				No - Default = current number of averages, initially = 0

94	Save channel definitions	!	{DF9}
			VERIFY? Yn, {DF10}
				DF9 - Default = paper tape punch (unit 2)
				DF10 - Default - paper tape reader (unit 2)
				Note: Yn and DF10 supplied in response to VERIFY? query

96	Restore channel definitions	*	{DF10}	DF10 - Default = paper tape reader (unit 2)
96	Accept INFORM commands	-	None	--

REFERENCES

1. Szuch, John R.; et al.: F100 Multivariable Control Synthesis Program: Evaluation of a Multivariable Control Using a Real-Time Engine Simulation. NASA TP-1056, 1977.

2. Flowchart Symbols and Their Usage in Information Processing. Am. Nat. Stand. Inst. Inc., Stand. no. X3.5-1970. Also FIPS PUB 24, June 30, 1973, U.S. Dept. of Commerce, National Bureau of Standards.

1. Report No. NASA TP-1424	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle INFORM - AN INTERACTIVE DATA COLLECTION AND DISPLAY PROGRAM WITH DEBUGGING CAPABILITY		5. Report Date Janaury 1980
		6. Performing Organization Code
7. Author(s) David S. Cwynar		8. Performing Organization Report No. E-9810
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135		10. Work Unit No. 505-05
		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Paper
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		14. Sponsoring Agency Code
15. Supplementary Notes.		

16. Abstract

INFORM was developed to aid assembly-language programmers of mini- and micro-computers in solving the man-machine communications problems that exist when scaled integers are involved. In addition to producing displays of quasi-steady-state values, INFORM provides an interactive mode for debugging programs, making program patches, and modifying the displays. Auxiliary routines SAMPLE and DATAO add dynamic data acquisition and high-speed dynamic display capability to the program. The report contains detailed programming information and flow charts to aid in implementing INFORM on various machines. Detailed descriptions of all supportive software are provided. Throughout the report, consideration is given to possible program modifications to satisfy the individual user's needs.

17. Key Words (Suggested by Author(s)) Engine controls; Digital computers; Software development; Operating system; Data acquisition; Data display; Diagnostics; Interactive; Communications; Computer program; System analysis		18. Distribution Statement Unclassified - unlimited STAR Category 60	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 168	22. Price* A08

* For sale by the National Technical Information Service, Springfield, Virginia 22161

National Aeronautics and
Space Administration

Washington, D.C.
20546

Official Business
Penalty for Private Use, $300

THIRD-CLASS BULK RATE

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

U.S.MAIL

```
 1   1 1U,G,       120379 S00903DS
DEPT OF THE AIR FORCE
AF WEAPONS LABORATORY
ATTN: TECHNICAL LIBRARY (SUL)
KIRTLAND AFB NM 87117
```

NASA

S

If Undeliverable (Section 158
Postal Manual) Do Not Return