DIGITAL 8-11-S

DĀTAK PROGRAMMING MANUAL

PDP-8

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

DIGITAL-8-11-S

PDP-8 DĀTAK PROGRAMMING MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

PREFACE

The programs discussed in this manual, though written on the Programmed Data Processor-8 computer, can also be used without change on Digital's Programmed Data Processor-5. This compatability between the libraries of the two computers results in four major advantages:

1. The PDP-8 comes to the user complete with an extensive selection of system programs and routines making the full data processing capability of the new computer immediately available to each user, eliminating many of the common initial programming delays.

2. The PDP-8 programming system takes advantage of the many man-years of field testing by PDP-5 users.

3. Each computer can take immediate advantage of the continuing program developments for the other.

4. Programs written by users of the PDP-5 and submitted to the users' library (DECUS Digital Equipment Corporation Users' Society) are immediately available to PDP-8 users.

CONTENTS

Chapter		Page
1	INTRODUCTION	1
2	SUMMARY OF THE DĀTAK SYSTEM	3
	Interfacing Transducers	3
	Interface Types	4
	Digital-Parallel Input Signal Buffer	4
	Serial-Parallel Input Signal Buffer	4
	Multiplexed Analog-to-Digital Conversion Input	5
	Programming	6
	System Inputs	6
	System Outputs	7
	Variables	7
	Time	8
	Arithmetic Operators	9
	Gray Binary Conversion	10
	Format Statements	10
	GOTO Statements	10
3	THE DĀTAK PROGRAMMING LANGUAGE	11
	DĀTAK Statements	11
	Clock	11
	Variable Names	12
	Format Statements	15
	Conditional Program Section	17
4	ERROR MESSAGES	21
	Compilation	21
	Run Time	21

CONTENTS (continued)

Appendix		Page
1	FLOW CHARTS	A1
2	PROGRAM EXAMPLES	A23

CHAPTER 1

INTRODUCTION

The speed of operation and powerful operation code structure of the Programmed Data Processor-8 make possible a unique programming system for use in data acquisition situations.

This system, called DĀTAK (for data acquisition), permits a complex, program-controlled, data acquisition system to be adapted to a particular experimental environment through the use of a highly sophisticated and precise pseudo code. The necessity of extensive machine programming to meet a particular data acquisition requirement is thereby eliminated.

In addition to its data acquisition applications, DĀTAK furnishes the experimenter with a means of calibrating transducers and is a powerful aid in troubleshooting a complex data-gathering system. Programs to accomplish these objectives may be written and readily revised in the DĀTAK language.

The DĀTAK system, when used to record experimental data on paper tape, produces a tape that may be used as input to a FORTRAN program. Thus the program for the processing of experimental data may be coded in a widely recognized compiler language if the user so desires.

A typical DĀTAK equipment configuration is illustrated in Figure 1.

DĀTAK does not, of course, require all of the equipment in Figure 1. If, for example, a given system did not include a plotter, the PLOT command would never be used.

Finally, the DĀTAK system may be readily adapted to special-purpose peripheral devices by specification of input/output transfer (IOT) commands appropriate to a given device. Thus a Type 350 Plotter might be replaced by a more elaborate plotting mechanism if desired.

The DĀTAK language presents for the first time a unified, systematic approach to the data acquisition, calibration, experiment modification, and troubleshooting requirements of the scientist engaged in experimental analysis. The recent use of digital computers in industrial

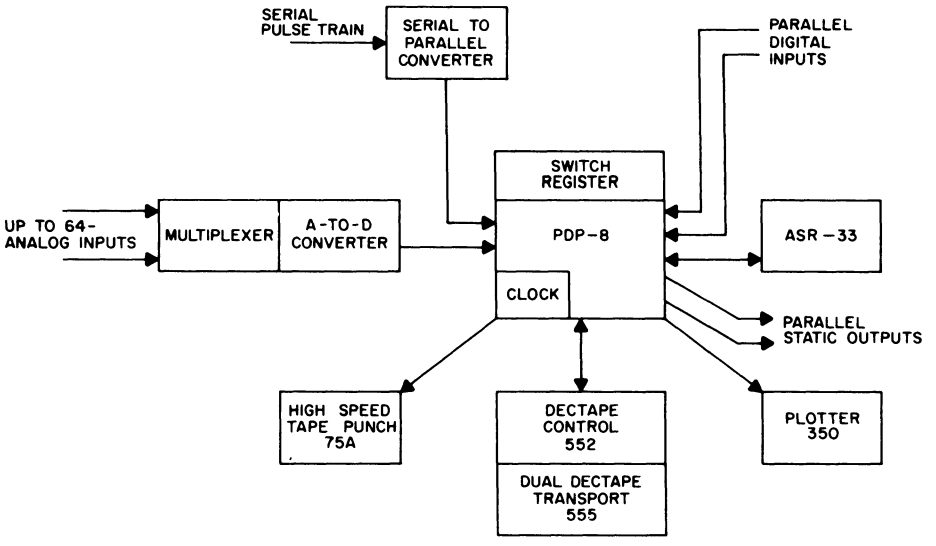


Figure 1 Typical Data Acquisition System

process control has shown that it is a valuable tool for acquiring information, both digital and analog, and making logical decisions concerning the acquisition of data while concurrently storing, processing, and displaying that information. Certainly a method of data manipulation similar to industrial control techniques would be advantageous to a scientist conducting an experiment. DĀTAK is a programming system designed precisely for these purposes; this manual explains how to use the DĀTAK system and language.

CHAPTER 2

SUMMARY OF THE DĀTAK SYSTEM

INTERFACING TRANSDUCERS

Three basic types of interfaces are used with the PDP-7 and PDP-8 computers in order to receive data from environment sensors. These interfaces allow data to enter the computer under control of a simple real-time symbolic compiler that gives the investigator the following flexibility when he samples the environment:

1. A variety of preprogrammed interface devices that easily connect the computer to instrumentation.

2. Simple symbolic assignment of identifying names to physical input variables such as pressure, temperature, etc.

3. Absolute control in sampling the experimental environment with respect to time.

4. Computer compatibility with respect to rapid response sensors using up to 176 separate sensing devices.

5. Capability to make logical decisions concerning the acquisition of data while sampling the environment.

6. Storage of data for future computations as well as immediate output for checking quality while obtaining data.

Transducers are sometimes considered unique devices that do not lend themselves to being connected directly to a computer; however, by the use of basic standard interface types, most instruments can be connected directly to a computer with little additional equipment. Let us examine the basic types of interfaces that would be necessary to interconnect a computer to various transducers.

INTERFACE TYPES

Digital-Parallel Input Signal Buffer

This buffer permits the direct parallel insertion of a digital number into the computer, using a method by which the number immediately becomes a computer word. Examples of devices feeding data in by this method are shaft-position encoders, switch registers, and other allied devices. Figure 2 shows the general method for digital-parallel input.

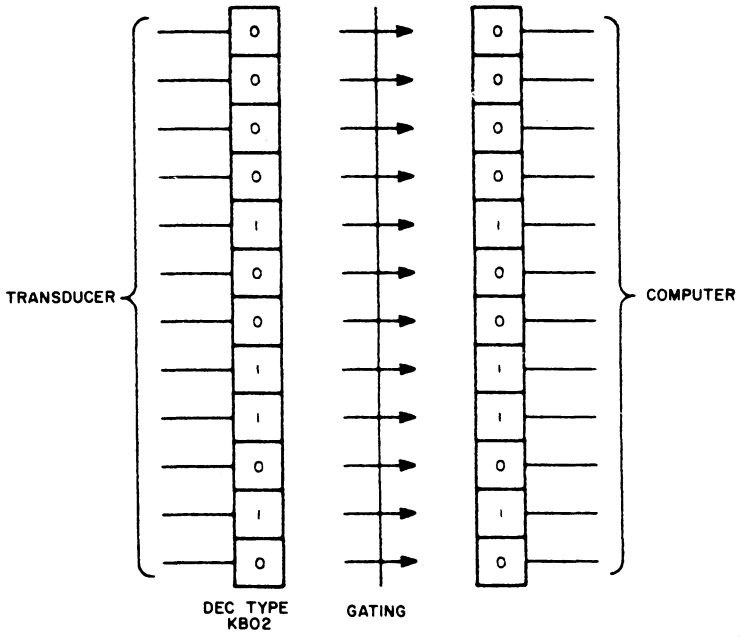


Figure 2 Digital-Parallel Signal Buffer

This method can accommodate signals or pulses from a minimum range of 0 to −10 mv up to a maximum of 20 to −15v. The system can include one buffer for each of several dozen variables.

Serial-Parallel Input Signal Buffer

This method would be used, for example, in telemetry applications where the transmitter is remotely located and able to transmit a number of input words one bit at a time along a single conductor cable or by radio (see Figure 3).

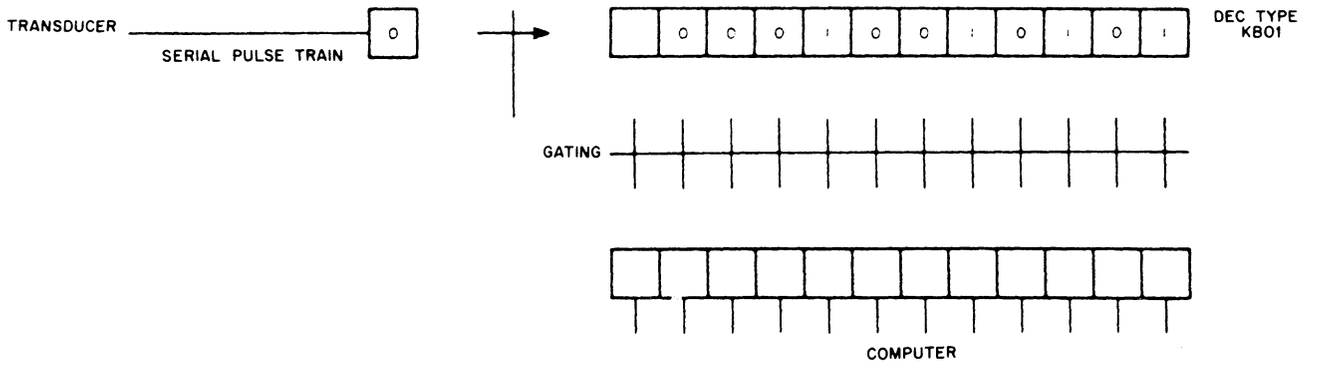


Figure 3 Serial-Parallel Signal Buffer

This buffer can convert a serial pulse train to a 12-bit word in 6 μsec, or one bit every 500 nsec. It accommodates ranges of input levels from a minimum of 0 to −10 mv, up to a maximum of 20 to −15v. The flexibility provided in this buffer allows the user to format the data before it is finally assembled as computer words; that is, he can insert octal constants in the specific serial word of his choice. It provides the programmer with the following additional instructions in the computer:

1. Skip if data flag = 0.
2. Skip if start flag = 0.
3. Clear data flag and start flag.
4. Read data into the accumulator.

Multiplexed Analog-to-Digital Conversion Input

This method is particularly advantageous in data acquisition when many devices such as thermistors, pressure sensors, and strain gages are working together. One example of where this method would prove advantageous is a thermistor chain in which each thermistor, pressure sensor, or conductivity sensor could be individually sampled by the computer. Figure 4 indicates this relationship.

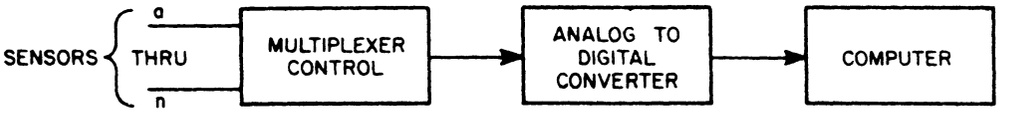


Figure 4 Multiplexed Analog-to-Digital Conversion

Switching from one input to the next in the multiplexer is accomplished in 2 μsec, and up to 64 separate inputs can be sampled directly by the computer. The analog-to-digital converter uses the range of 0 to −10v.

PROGRAMMING

The task of writing a special-purpose program for each installation could be a formidable problem involving a great deal of time and money. In order to alleviate this problem, Digital Equipment Corporation has written DĀTAK, an algebraic compiler that allows the experimenter to format his data acquisition problem in a simple language similar to algebra. By using the DĀTAK language, he is given a great deal of flexibility concerning the interface hardware that he has available. This program allows him flexibility in choosing the frequency and conditions under which he samples the experimental environment as well as making possible the adding or changing of sensors without the major task of reprogramming in machine coding. This program is available for the PDP-8, a compact 12-bit computer with a 1.5-μsec cycle time.

In summary, this program allows the investigator to analyze his sample using the following inputs.

Up to 96 independent data variables using digital-parallel input.
Up to 64 independent data variables using a multiplexed A-to-D converter.
Up to 25 independent data variables via serial buffer input.

Each independent variable can be sampled at a rate of up to 100 times per second.

System Inputs

Data can come to the computer from the digital-parallel input buffer, the serial input buffer, and the multiplexed analog-digital converter. Each of these devices is assigned a symbolic name that tells the computer which device is transmitting information.

The symbols are:

DGIN: Digital-parallel signal buffer; input can be converted from Gray code to binary.

BUFR: Serial buffer input.

ADCV: Multiplexed analog-digital converter.

System Outputs

Data output can be distributed to a number of specifically named devices to allow immediate presentation as well as permanent storage. The following output symbols and their associated devices are available:

TYPE: On-line teleprinter. Variables can be typed in decimal or octal. Decimal is specified by † immediately following the variable name.

PNCH: High-speed paper tape punch. Variables can be punched in decimal or octal. Decimal is specified by † immediately following the variable name.

DCTP: Digital's compact DECtape. Variables are recorded magnetically on DECtape in binary with identifying words.

PLOT: X-Y plotter. The plotter pen is moved to a new position each time an output is specified.

DIG 1 : DIG 4: Up to four digital outputs are available through parallel buffers to other devices such as relays, buffers, sense lines, and range switching devices.

Variables

Input variables to the computer are assigned alphanumeric symbols by the investigator. They can be one to four characters long, and must begin with a letter. Some examples of these would be:

T 123, SURF, DEEP, AIR, X, Y, TEMP, H20, H202

In addition, variables that are inputted through the multiplexer have specified channels; that is, T 123 (1) would be input through channel 1 of the multiplexer.

Time

Four types of time can be used by the computer: basic, program, variable, and reference.

Basic Time

Basic time represents the basic interval of .01 sec in which a clock interrupts the program.

Program Time

This time represents the basic rate at which the investigator desires to interrogate the sensors. It is some multiple of the basic time and is under program control. Its symbology is simply expressed as follows:

QUNT: 50 The investigator has specified that the program time will be (50) x (.01) = 0.5 sec; that is, each sensor will be sampled every 0.5 sec. If he desires the fastest rate possible, he may express the following:

QUNT: 1 In which case each variable will be sampled every 0.01 sec.

Variable Time

In order to allow more flexibility in timing, digital inputs can be sampled at a slower rate than the program time specifies. For example, the expression:

DGIN:L1 (4, L2 (4

specifies that the digital input variables L1 and L2 should be sampled once in four cycles of the program time or every (4) x (0.5) = 2 sec.

Reference Time

It is often desirable to know the reference time in order to associate data with time. Within the program is a 3-word variable, CLOK, which counts the number of seconds, minutes, and hours that have elapsed since start-up time; it can be used as an output variable to reference data with time.

Arithmetic Operations

Addition and Subtraction

Variables can have constants added to or subtracted from them as they are sampled, or the variables can be added to or subtracted from each other.

All arithmetic operations are done in 2's complement arithmetic, with the operands being considered signed, fixed-point numbers. The following examples mean that the variable T2 will have constants or variables added or subtracted before output:

T2 + 137	Add a constant to the variable.
T2 - 3	Subtract a constant.
T2 + T3	Add a second variable.

Arithmetic Comparison

Variables can be compared against constants, compared against other variables, or compared against themselves with respect to sample time. The basic comparison instructions are:

IFEQ X, Y	if X is equal to Y
IFLS X, Y	if X is less than Y
IFGR X, Y	if X is greater than Y

An example of the comparison of a variable against a constant would be:

```
IFEQ X, 1000;
```

meaning that is X is equal to 1000_8, execute the operation following the semicolon; otherwise, go to the next line.

Every time a variable is recorded and outputted, its value is preserved and is given the name of the original variable. Thus, X and @ X represent the current value of the variable X, and its value when last used as output, respectively.

The following example of the comparison of a variable and its predecessor:

```
IFLS X, @ X;
```

means that if X is less than it was when last recorded and output, execute the operation following the semicolon; otherwise, go to the next line.

Gray Binary Conversion

Gray binary code can be converted to simple binary under program control if the input method is digital (DGIN). This provides the investigator with a rapid means of conversion in order to intercompare the usual shaft-encoded Gray binary numbers, if the shaft encoder does not convert from Gray Code to simple binary prior to buffering.

The conversion is accomplished by inserting ↑ immediately before the time multiple.

```
DGIN L1↑4
```

This means that the Gray binary variable L1 is sampled every fourth time through the program and is converted to a simple binary number before comparison or storing.

Format Statements

Format statements are numbered from 1-15 (decimal). They contain the names of variables and their output forms (octal or decimal for the Teletype or punch). Format numbers appear along with a device name in every output statement. Thus, the statement:

```
FORM: 1,X,Y↑,Z
```

indicates that the variables X, Y, and Z are to be outputted to the Teletype, with X typed in octal, Y typed in decimal, and Z typed in decimal.

GOTO Statement

Program control can be unconditionally transferred through the use of the GOTO statement.

CHAPTER 3

THE DĀTAK PROGRAMMING LANGUAGE

Every DĀTAK program consists of four parts:

1. Assignment of input devices.
2. Definition of output formats.
3. Unconditional outputs, i.e., outputs controlled by the clock.
4. The conditional program section. This is illustrated in the short DĀTAK program below:

```
QUNT:  62
DGIN:  SWIT (2                                          INPUT ASSIGNMENT

FORM:1,CLOK,SWIT,X,Y,Z                                  OUTPUT DEFINITIONS
FORM:7,CLOK,SWIT

<7,PNCH,2>                                              UNCONDITIONAL OUTPUTS

[ 5:   Y=1+(X=SWIT-10)!77                               CONDITIONAL
   :   Z=Y←(X+17);OUTP(1,TYPE);OUTP(1,DCTP)             PROGRAM
  6:   IFEQ@SWIT,SWIT;GOTO 6                            SECTION
   :   GOTO 5
]
  END                                                   END STATEMENT
```

The maximum hardware configuration for which DĀTAK is designed is illustrated in Figure 5.

DĀTAK STATEMENTS

Clock

All sampling of the input is controlled by the clock, except for the serial-parallel converter which sends a signal to the PDP-8 when it has twelve bits in its buffer. Output to the devices may be unconditionally controlled by the clock or may be initiated when certain prescribed conditions are met. When a variable or quantity is output, its value at output time is stored and hence may be compared with incoming values to detect changes. The basic clock cycle is .01 sec.

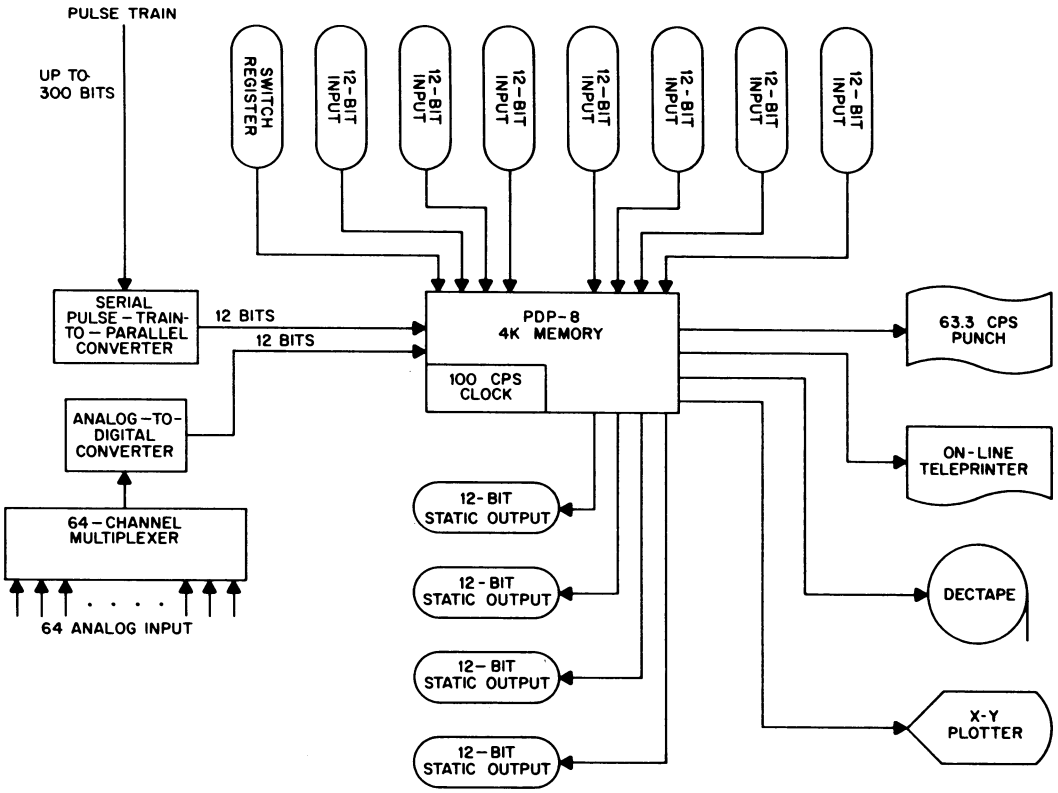


Figure 5 Maximum Hardware Configuration

The sampling period for the analog-digital converter is some multiple of this as specified by the statement:

QUNT: XXXX

where XXXX is a given decimal integer. Leading zeros need not be specified. For example:

QUNT: 5

means that all specified channels of the analog-digital converter are to be sampled every 0.05 sec and the values stored.

Variable Names

Variable names may be up to four characters long, must start with a letter, and may contain only letters and numbers:

T723 SYMB SWIT HEAD

Each variable is assigned two unique addresses. One is used by the input routines and contains the most recent input value. The second address contains the value of the variable when it was last output. The output variable may be addressed in the conditional program section by preceding the variable name with "@." This may be used for comparison purposes, etc.

Variable names are assigned to input devices in the following manner:

1. Analog-to-Digital Converter

   ADCV:

   followed by variable names and channel numbers; thus:

   ADCV: T723(6), HEAD(4)

   This means that channels 6 and 4 of the analog converter-multiplexer will be sampled and the values stored in registers T723 and HEAD, respectively. These will be sampled at some multiple of .01 sec, where the multiple is specified by a QUNT: statement.

   QUNT: 5
   ADCV: T723(6), HEAD(4)

   Channels 6 and 4 of the converter-multiplexer will be sampled every .05 sec.

2. Serial-Parallel Converter

   BUFR:

   The serial-parallel converter sends a start pulse to the computer, followed by data pulses every time twelve bits have been assembled. BUFR: is followed by variable names in the order in which the 12-bit words are received. If the computer received. If the computer receives more data pulses than there are variables, additional data from the converter is ignored. If more variables are specified than there are data pulses, the remaining variables are not used. Each start pulse causes the list to be reset.

   BUFR: SYMB, THIS

After the start pulse from the buffer, succeeding data pulses cause the 12-bit words to go into registers named SYMB, THIS. If a third data pulse is received before the start pulse, it is ignored.

3. Digital Inputs

The program is capable of handling up to eight 12-bit digital inputs. These are sampled at periods that are multiples of the period specified by the QUNT: statement. These inputs may be converted from 12-bit Gray code to 12-bit binary if specified.

```
QUNT:   2
DGIN:   SWIT(6, CODE↑2
```

This means that the first digital input (usually the switch register) is to be sampled and stored at a period that is six times the period specified by QUNT: 2 or every .12 sec. The second digital input is sampled every (2) (.02) sec or .04 sec and is to be converted from Gray code to binary before it is stored. This is useful when sampling shaft encoders, etc.

4. Digital Outputs

Output is specified with a format number and a device name. Format statements contain the format number and a list of variables. Output form depends on the output device.

DIG1, DIG2, DIG3, DIG4 — These are static output devices; i.e., relay buffers, etc. The first variable in the specified format statement is output to the appropriate device as a 12-bit binary number.

Teleprinter

TYPE — The entire list of variables in the specified format statement is typed in octal (or decimal) on the 33 ASR. Each output is preceded by two decimal digits which are the format number, followed by the values of the variables separated by spaces, and finally a carriage return-line feed. Decimal output is specified in the format statement.

High-Speed Paper Tape Punch

PNCH — Output format is identical to the teleprinter format. Since the format number starts every line, this tape may be processed with the PDP-8 FORTRAN System.

DECtape

DCTP — The datum is recorded on DECtape using block lengths of 201_8. The first word of each block indicates whether or not it is the last block. The second word contains the number of 12-bit words of data on the block. Each command to output to DECtape causes a 12-bit identifier word to be placed on the tape followed by 12-bit binary words. Decimal specifications in format statements are overridden. When the DECtape nears the end of the tape, a message is typed and a fresh tape may be mounted on the second transport.

X-Y Plotter

PLOT — The first two variables in the specified format statements represent, respectively, the X and Y coordinates. The plotter pen is moved from its previous position to this specified position. It is initially set to location (0.0)--the lower left-hand corner of the plotting area.

Format Statements

Format statements are of the following form:

```
FORM:   6,SWIT†,HEAD
```

This is format statement number 6 which consists of the variables SWIT and HEAD. If output is to the teleprinter or punch, SWIT is converted to decimal when it is outputted.

There is a fixed system variable called CLOK. This is a 3-register variable containing, respectively, the hours, minutes, and seconds. The seconds are initially set to 0, and the hours and minutes are set from the switch register when the execution of the program is initiated.

When output is to the teleprinter or to the punch and CLOK is in the specified format statement, it is outputted as three 2-digit decimal numbers. When it is output to the DECtape, it is written as three 12-bit words. CLOK may not be output to the plotter or to any of the four digital outputs. However, the following is permissible:

```
FORM:    12, X,Y, CLOK
```

with format statement 12 being specified in output statements to the punch and to the plotter.

Output may be initiated unconditionally (i.e., by the clock) or conditionally. Unconditional output statements contain the format statement number, the device name, and the output period (a multiple of the clock period established by the QUNT: statement).

The unconditional output statements are delimited by angle brackets < >.

Example: A program to:

1. Sample the contents of the switch register every .5 sec;
2. Punch this value in octal every .5 sec; and
3. Type the value in decimal, along with the time, every 2 sec;

could be programmed as follows:

```
QUNT:    50
DGIN:    SWIT(1
FORM:    7,CLOK,SWIT↑
FORM:    6,SWIT
<7, TYPE, 4
 6, PNCH,1>
END
```

The first statement, QUNT: 50, established the time interval as (50) · (.01 sec) = · 5 sec.

The second statement, DGIN: SWIT(1, identifies the first digital input as the variable SWIT which is to be sampled every (1) (.5 sec) = .5 sec.

The next two statements identify the output variables. Decimal output is specified for SWIT in format statement number 7.

The unconditional output statements are interpreted as follows:

7,TYPE,4 output to the teleprinter under control of format statement number 7 every (4) (.5 sec) = 2 sec.

6,PNCH,1 output to the punch under control of format statement number 6 every (1) (.5 sec) = .5 sec.

Format numbers are decimal and may be from 1-15.

Conditional Program Section

The conditional program section consists of algebraic and control statements that may be used to initiate output, test for tolerance levels, or alter variables.

There are four types of statements allowable in this section of the program:

1. Algebraic Expressions

: Variable = Variable Operator Variable

For example:

: Y=A+B.

These statements may be nested to any reasonable degree:

: Y=A+(B=C+D)

There are four basic operators:

A+B	2's Complement Addition	Arithmetic Operators
A+B	2's Complement Subtraction	Arithmetic Operators
A!B	Inclusive OR	Boolean Operators
A←B	Clear bit. For every bit in B that is a 1, clear the corresponding bit in A. A∧B	Boolean Operators

Truth tables for the logical operations are as follows:

A	B	A.!B	A←B
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	0

Examples:

A	B	A+B	A-B	A.!B	A←B
6000	7000	5000	7000	7000	0000
0077	0017	0116	0060	0077	0060

Expressions within parentheses are evaluated first. The Boolean operators have a higher priority than the arithmetic operators:

```
: Y=A+B←C
```

is interpreted as if it had been written:

```
: Y=A+(B←C)
```

The equal sign (=) means replace the single variable on the left with the value of the expression on the right. Thus:

```
: X=B+(Y=C-D)
```

is perfectly valid. Two equal signs may not be used in one expression at the same level. Thus:

```
: A=B=0
```

is not valid, although:

```
: A=(B=0)
```

is valid.

An arithmetic statement is terminated by either a carriage-return or a semi-colon (;). The single quote (') acts as a line continuer as it does in all program sections.

```
: Y=A+B; Z=Y+D←A
```

is a valid line and is evaluated from left to right.

2. Unconditional Transfer

Any statement may begin with a statement number delimited by a colon (:). There is one transfer instruction:

```
: GOTO XX
```

which says: transfer control to statement number XX.

Example:

```
6: Y=A+(B=C-D)
 : GOTO 6
```

The GOTO statement must be the last statement on a line, although it need not be the first.

```
  :     :
  :     :
6: Y=A+(B=C-D) ; GOTO 7
  :     :
  :     :
```

3. Conditional Expressions

In addition to the unconditional transfer instruction, there must be conditional expressions. Conditional expressions are of the following general form:

```
FUNCTION EXPRESSION, EXPRESSION;
```

If the conditions are met, the instruction following the semicolon is executed; if they are not met, execute on the next line. The expressions may be a single variable or a statement within parentheses. The conditional functions are:

IFEQ X,Y; If equal. If X=Y, the statement is true. The arguments are considered unsigned.

IFLS X,Y; If less than. If X is less than Y in absolute value, the statement is true. The arguments are considered to be signed integers. Thus: 7770 <211

IFGR X,Y; If greater than. If X is greater than Y in absolute value, the statement is true. The arguments are considered to be signed integers.

IFBE X,Y; If bit equal. If there are any corresponding 1's in X or Y, the statement is true. IFBE 177,1; is true.

For example, if:

```
6: IFEQ A,B; GOTO 6
 : GOTO 5
```

This waits in a loop until A and B are unequal. To reverse it:

```
6: IFEQ A,B; GOTO 5
 : GOTO 6
```

This waits in a loop until A and B are equal.

4. Output Statements

Output statements are expressed in the conditional section as follows:

OUTP (format number, device name)

Whenever a variable is output, its value at output time is recorded in a location that has, as a name, the original variable name preceded by "@."

The conditional program section is delimited by square brackets [ ].

The last statement in a program is END.

CHAPTER 4

ERROR MESSAGES

COMPILATION

During compilation, there may be two diagnostic messages:

LANGUAGE ERROR	There was some error in the source language. Since the source language is typed out as it is processed, the programmer can see where the error occurred.
S # NOT FOUND	A statement number was referenced by a GOTO statement in the conditional program section, but has not been defined anywhere in the program.

Both of these errors cause the compiler to halt. It must be restarted at 200_8.

RUN TIME

At run time, there may be several diagnostic messages:

EX INTR	Extraneous interrupt. Some device that is not used by DĀTAK has caused an interrupt.
INTR OVR	Interrupt overflow. DĀTAK was interrupted at a rate higher than its processing rate.
FORM OVR	Format overflow. The output devices are incapable of handling the specified output rates. The format list is about 30 positions long.
STACK ERROR	Indicates an error in the arithmetic statements that was not detected by the compiler.

The above errors are catastrophic; that is, no recovery from these errors is possible, and the object system halts.

DT ERROR	The DECtape error flag was raised for something other than an end-zone condition. DĀTAK will clear the flag and attempt to proceed.
TIMING ERROR	The DECtape buffer was filled before the second buffer could be written. DĀTAK will continue although some information will be lost.
READY TO SWITCH	DECtape block number 2400 is being written. The write routines will switch from DECtape unit 1 to DECtape unit 2 at block number 2700. The next time the switch will be from DECtape unit 2 to DECtape unit 1, etc.

APPENDIX 1

FLOW CHARTS

EXPLANATION OF FLOW CHARTS

DĀTAK consists of two basic elements: the compiler and the operating system. The function of the compiler is to read the source language from the ASR-33 Paper Tape Reader or Keyboard. The compiler produces the structured lists and other elements that control the operating system, and compiles the logical and arithmetic statements into an interpretive code that is executed at run time. When the compiler reads the word END, it initializes the operating system and halts. The switch register is set to the current time of day and when the CONTINUE Key is depressed, the CLOK registers are set to this value and the operating system is started.

The operating system consists of three major sections:

1. The arithmetic interpreter, which executes the interpretive code generated by the compiler from the arithmetic and logical statements.

2. The output controlling routines, which call routines for specific devices and initiate output through use of the interrupt system.

3. The interrupt system, which handles all of the actual output operations, counts elapsed time, and initiates sampling and unconditional output.

The arithmetic interpreter may call the output controlling routines (conditional outputs), as may the clock service routines (unconditional outputs). Either type of routine may be interrupted except for the DECtape flag servicing routines, which are accorded the highest priority.

Page A3 of the flow charts shows the overall flow of events at run time. After each interpretive instruction is executed, the status of the output waiting list is tested. If there is an output word waiting to be processed, it is transmitted to the appropriate device handling routine. Page A4 shows the overall flow of the compiler.

Pages A5 and A6 show the list structures used by the input/output routines.

Page A7 shows the interrupt servicing technique. If the interrupt was not caused by the DECtape; the C(AC), C(L) and the return address are pushed down onto a list. When the flag causing the interrupt has been determined, it is cleared and the interrupt is reenabled.

Page A8 shows the clock service routine. It counts to 1 sec before incrementing the CLOK registers. It also counts to 1 time quantum (specified by QUNT:) before testing to see if any input or output is to be initiated. It may call the routines ADCSER (analog-digital converter service routine), DISERV (digital input service routine) or CGO1 (unconditional output controller).

Page A9 illustrates the serial-parallel buffer service routine.

Page A9 shows the teleprinter and punch service routines. Both of these routines have buffers which are cleared before new data is read in.

Pages A10 and A11 show the routines called by the clock service routine to initiate sampling or output. The digital input routines contain a list of eight IOT's that the user may define to correspond to his particular equipment configuration.

Page A11 shows the output-control routines. FORMAT is used to place an output word on the waiting list and to test for overflow. TFORM tests to see whether or not the waiting list is empty. If it is not empty, the top word on the list is saved and all others are moved up one position on the list. FOP1 is then called to operate on this format-output word. If the list is empty, TFORM exits. FOP1 calls the appropriate device routine and then returns to TFORM to test the waiting list status.

Page A12 contains the digital-output routines. The user may define up to four IOT instructions, one each for DIG1, DIG2, DIG3, and DIG4, to correspond to his equipment configuration.

Pages A12 through A15 contain routines used for teleprinter and punch output.

Pages A15 through A20 contain the DECtape output routines. DECtape uses two buffers in core memory and one may be filled while the second is being written. The write routine DWRT determines whether or not the tape is nearing the end; if it is, the message "READY TO SWITCH" is typed, and 300_8 blocks later write-out is continued on another DECtape unit. The DECtape search routine is informed of the setting of the DT flag by the interrupt service routine IOSERV.

Page A21 contains the plotter service routines. The plotter attempts to draw a diagonal line from its previous position to its new position.

Page A22 contains the arithmetic interpreter. It calls routines to carry out indicated operations. This is the main "background" program for DĀTAK. If there are no arithmetic statements in the source program, DĀTAK executes a loop in the interpreter.

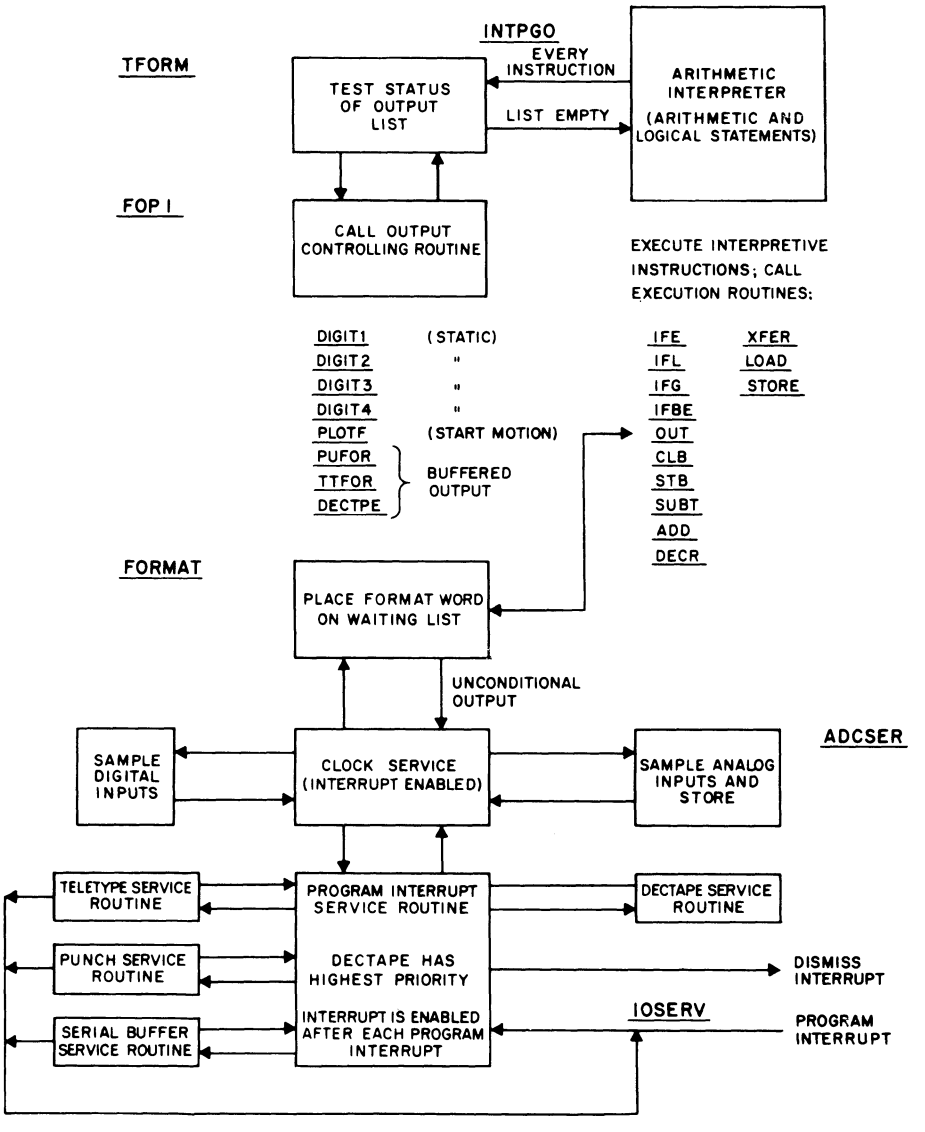


Figure A1 Overall Flow (Run Time)

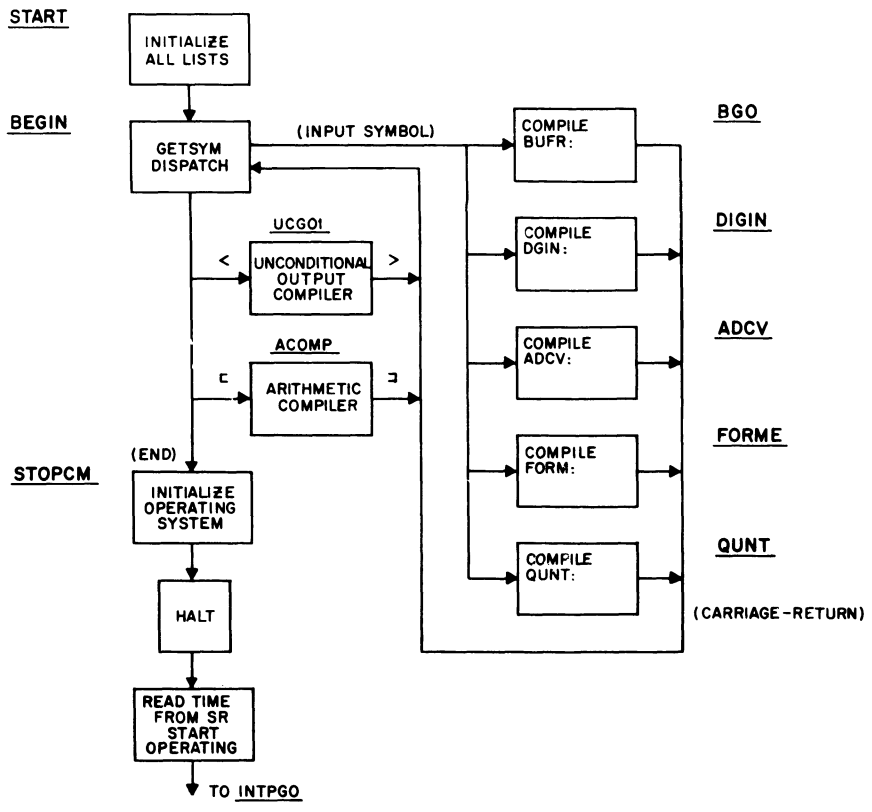


Figure A2 Overall Flow (Compile Time)

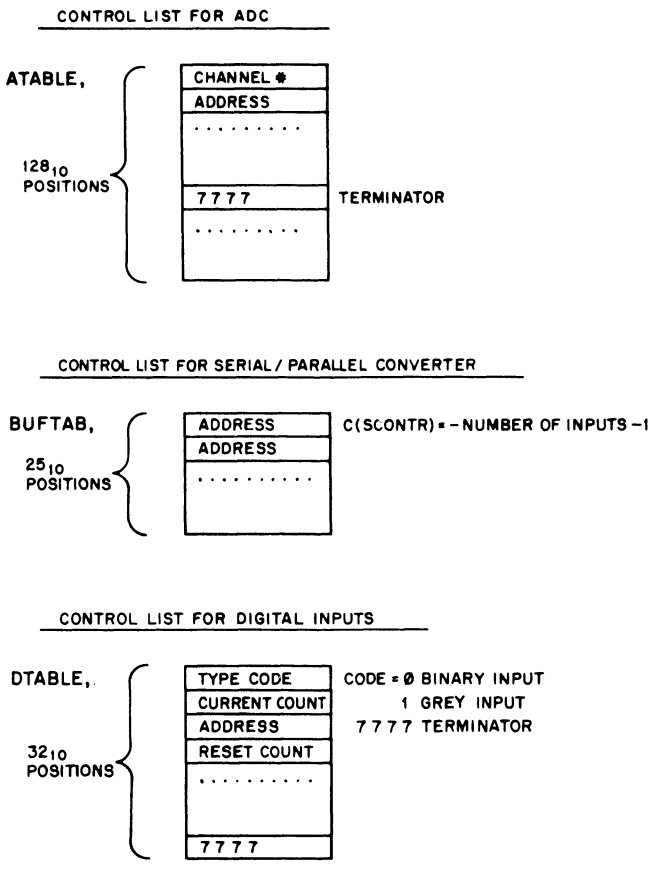


Figure A3 Input Device Control Lists

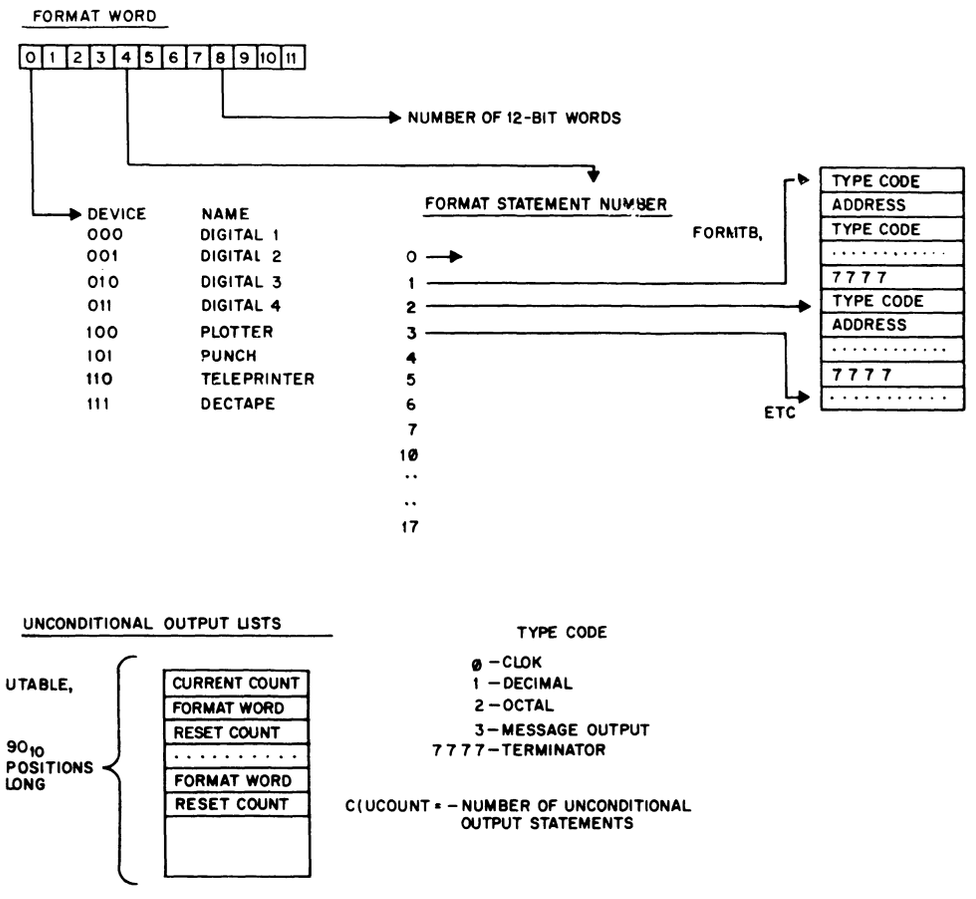


Figure A4 Output Control Lists

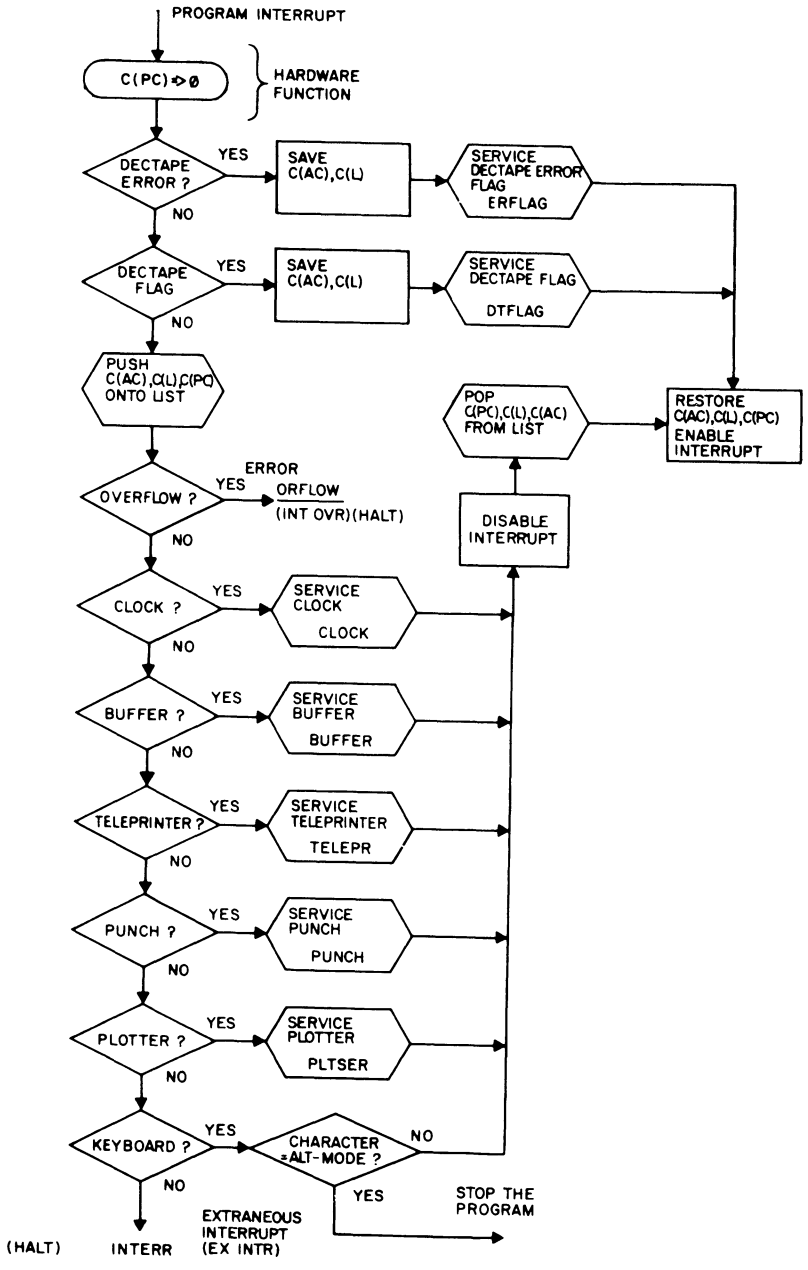


Figure A5 Interrupt Service Routines

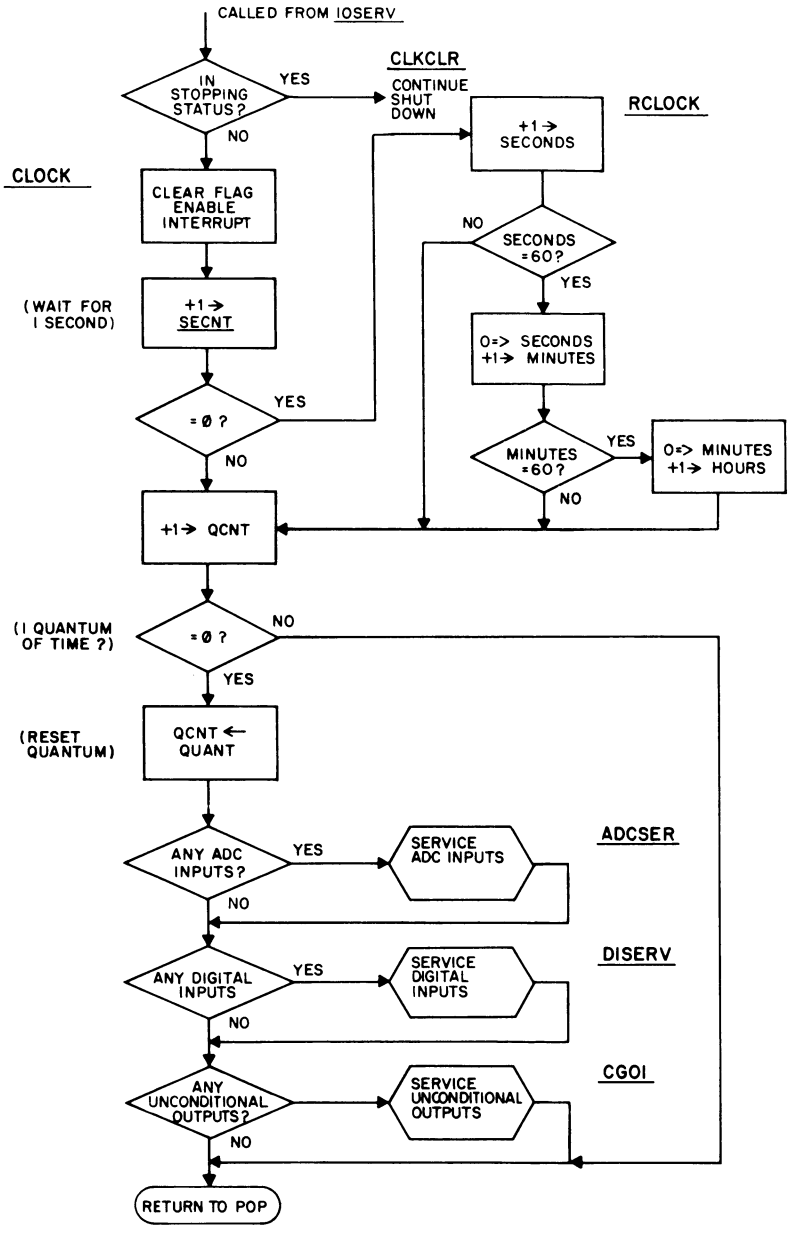


Figure A6 Clock Service Routine

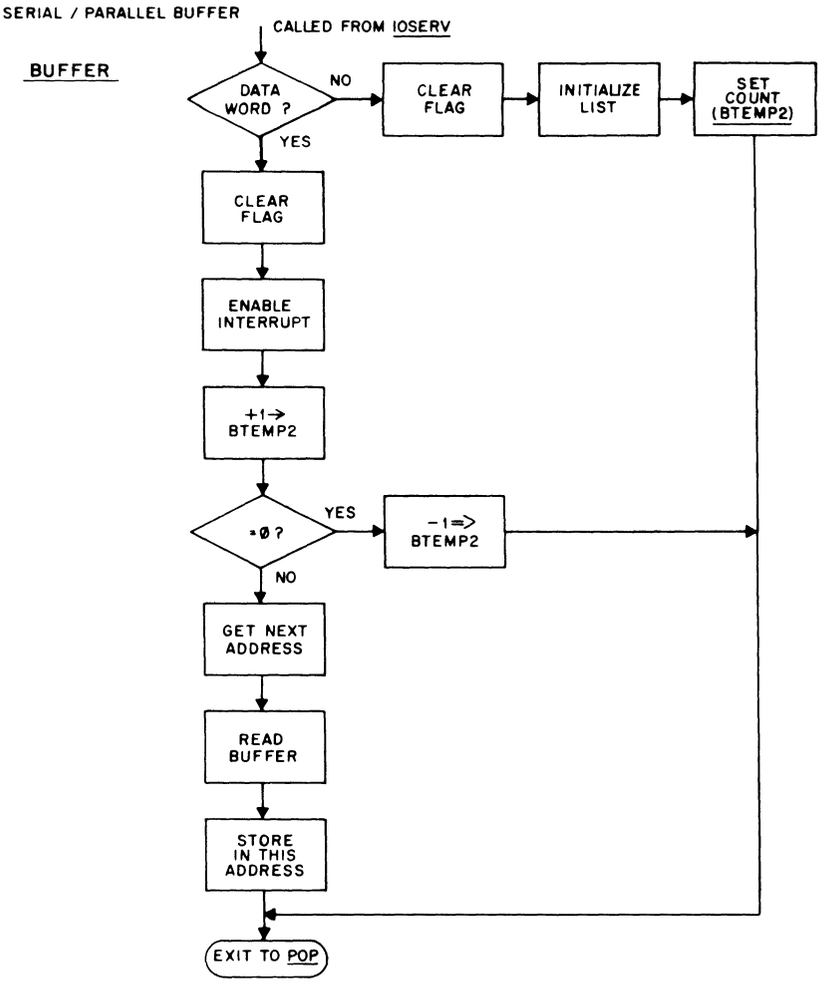


Figure A7 Buffer Service Routine

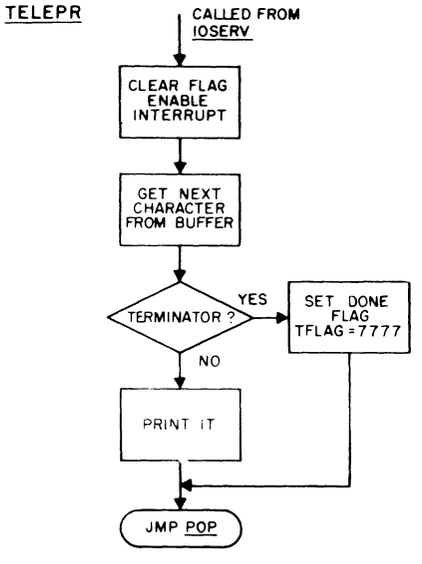


Figure A8 Teletype Service Routine

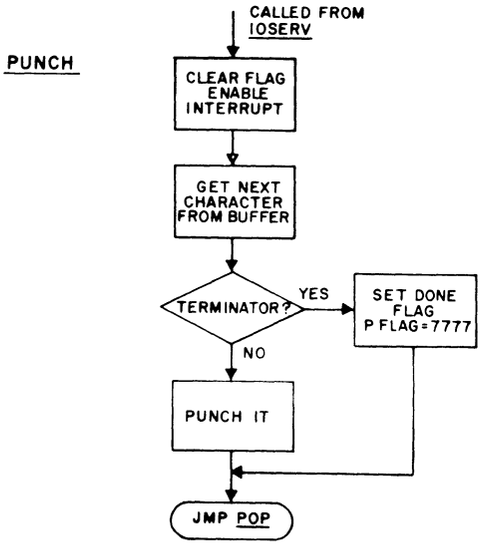


Figure A9 Punch Service Routine

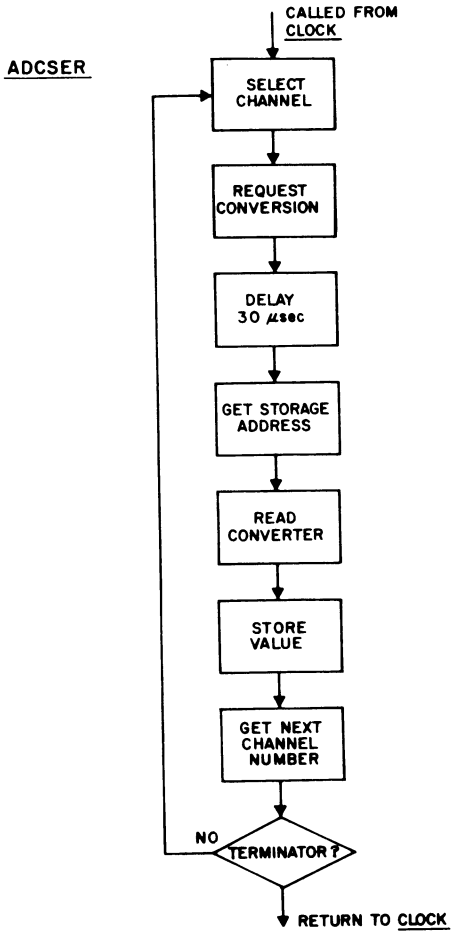


Figure A10 Analog-Digital Service Routine

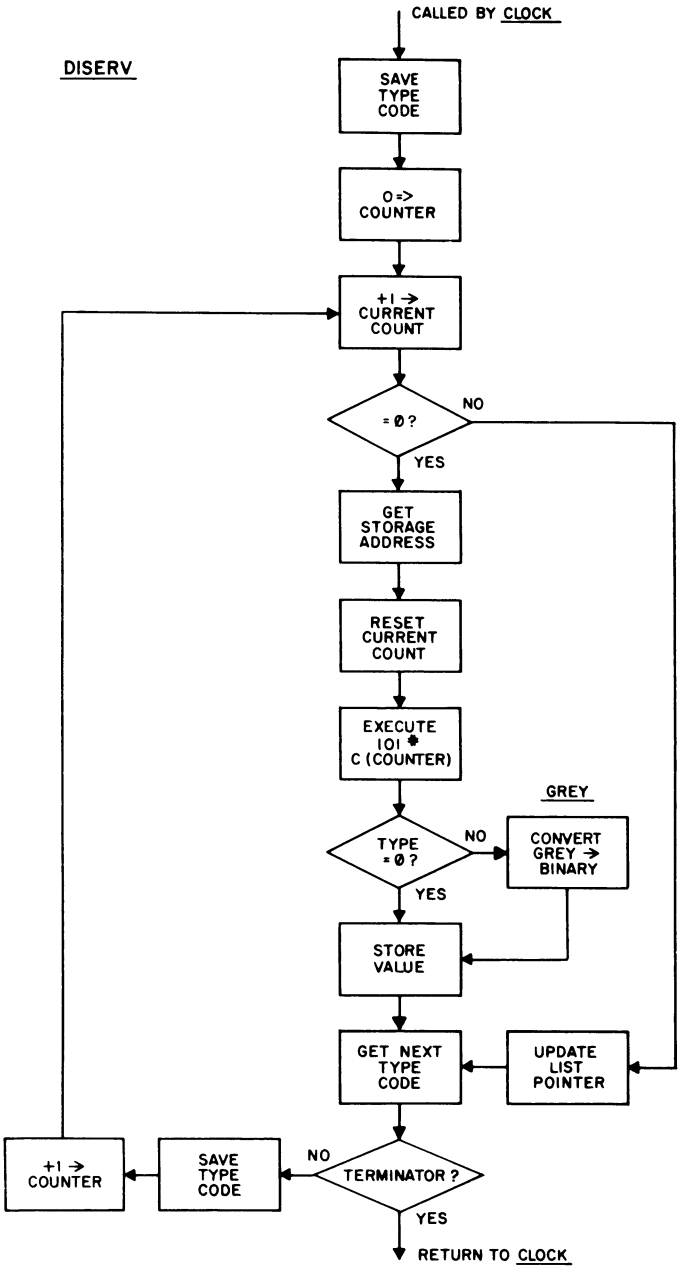


Figure A11 Digital Input Service Routine

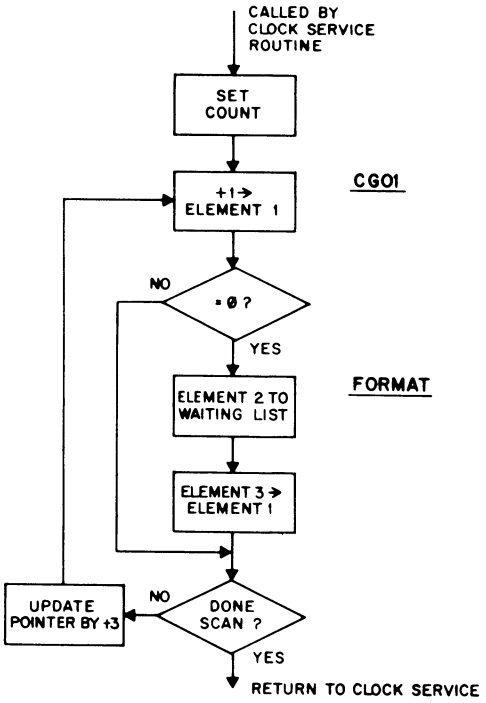
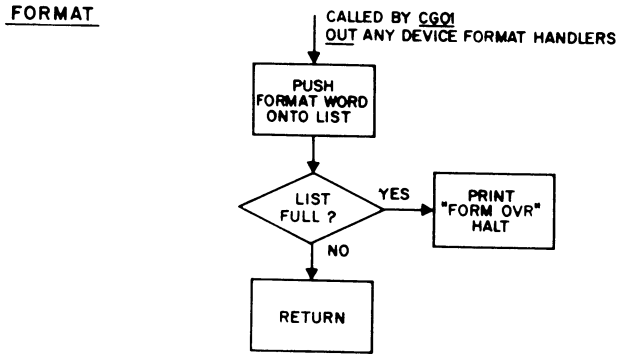


Figure A12 Unconditional Outputs



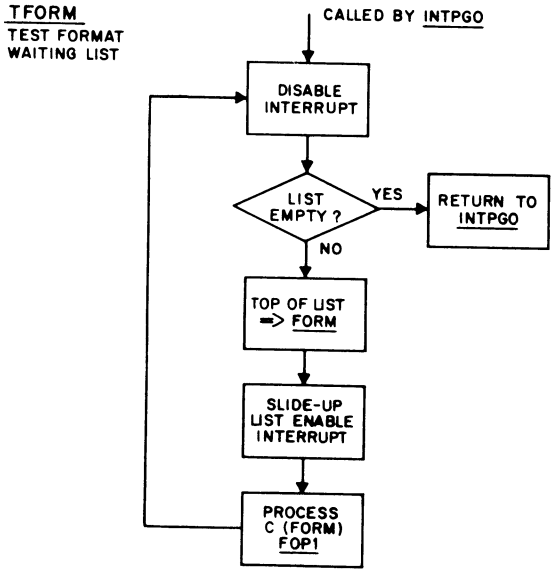


Figure A13 Format Output Routines

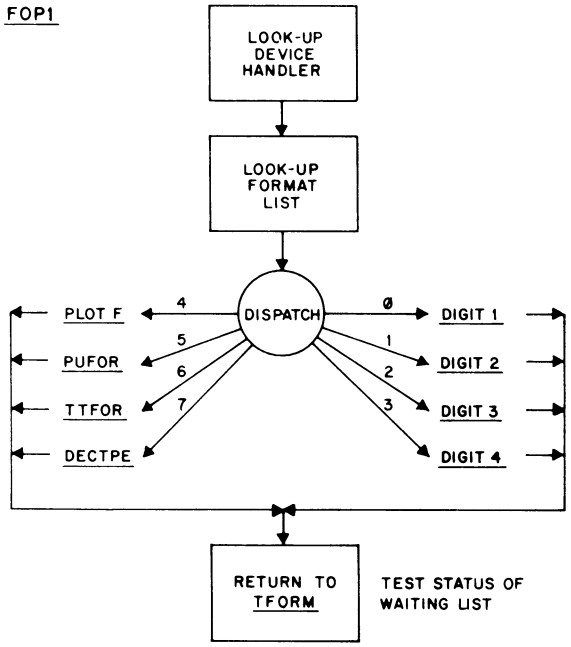


Figure A14 Call Output Device Format Routines

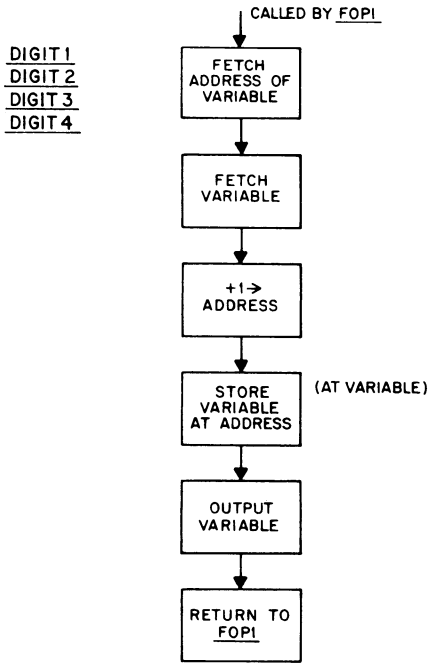


Figure A15 Digital Output Format Routine

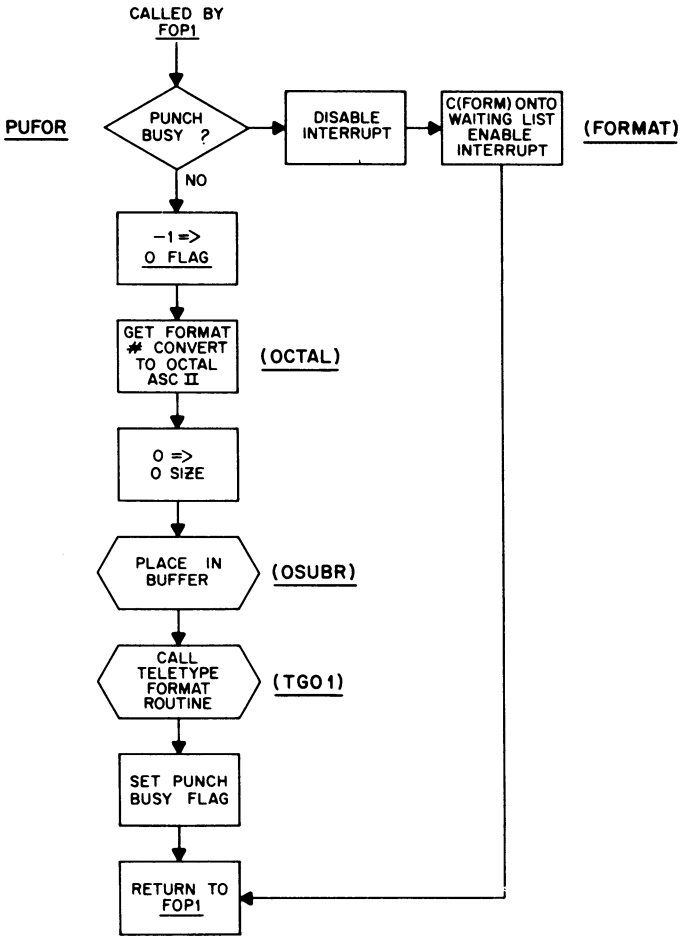


Figure A16 Punch Format Routine

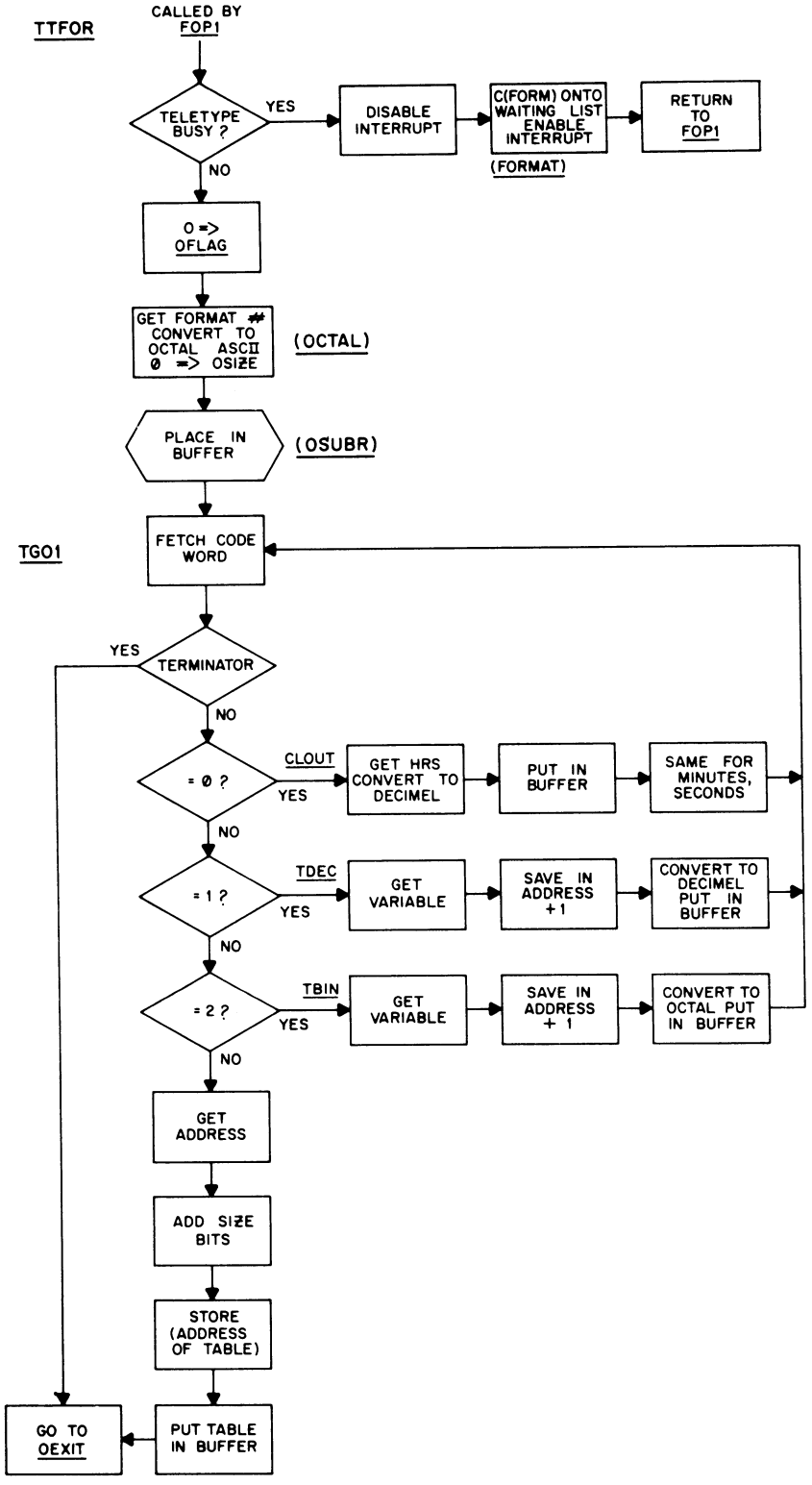


Figure A17 Teleprinter Format Routine

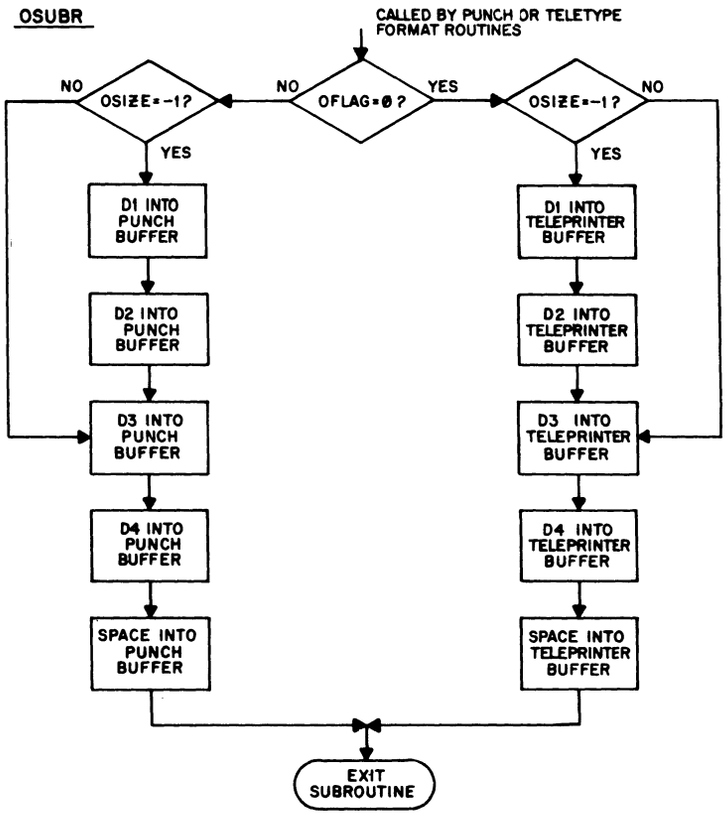


Figure A18 OSUBR

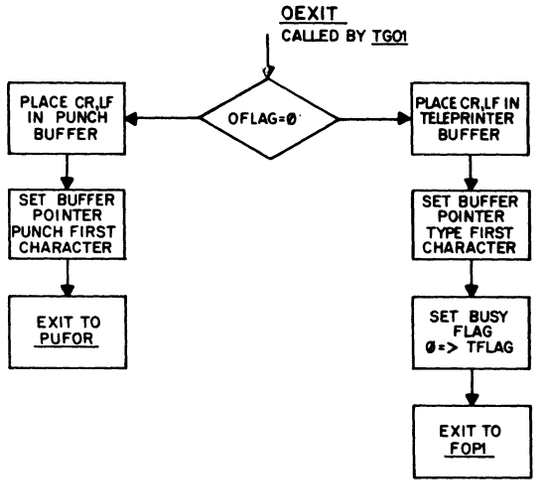


Figure A19 OEXIT

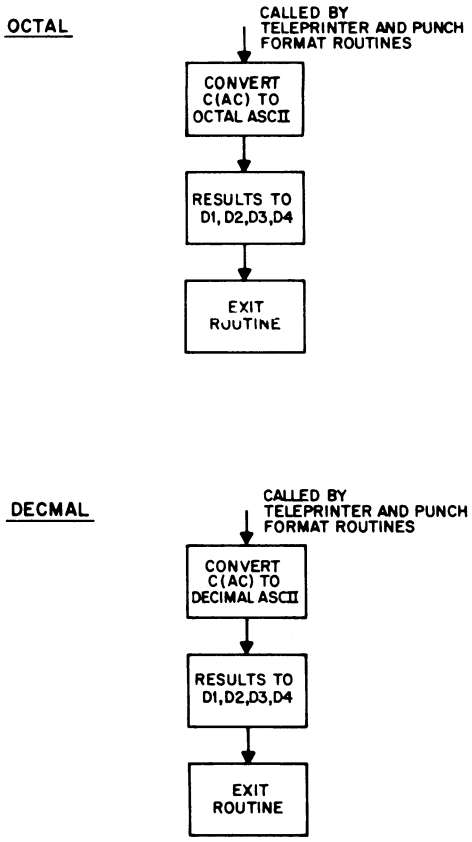


Figure A20 Conversion Routines

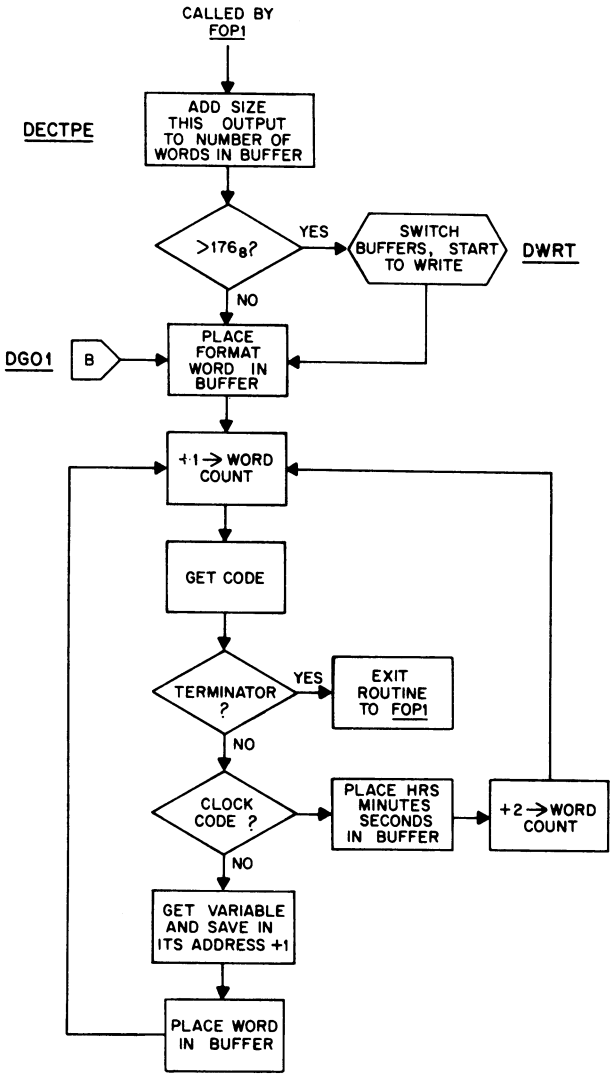


Figure A21 DECtape Format Routines

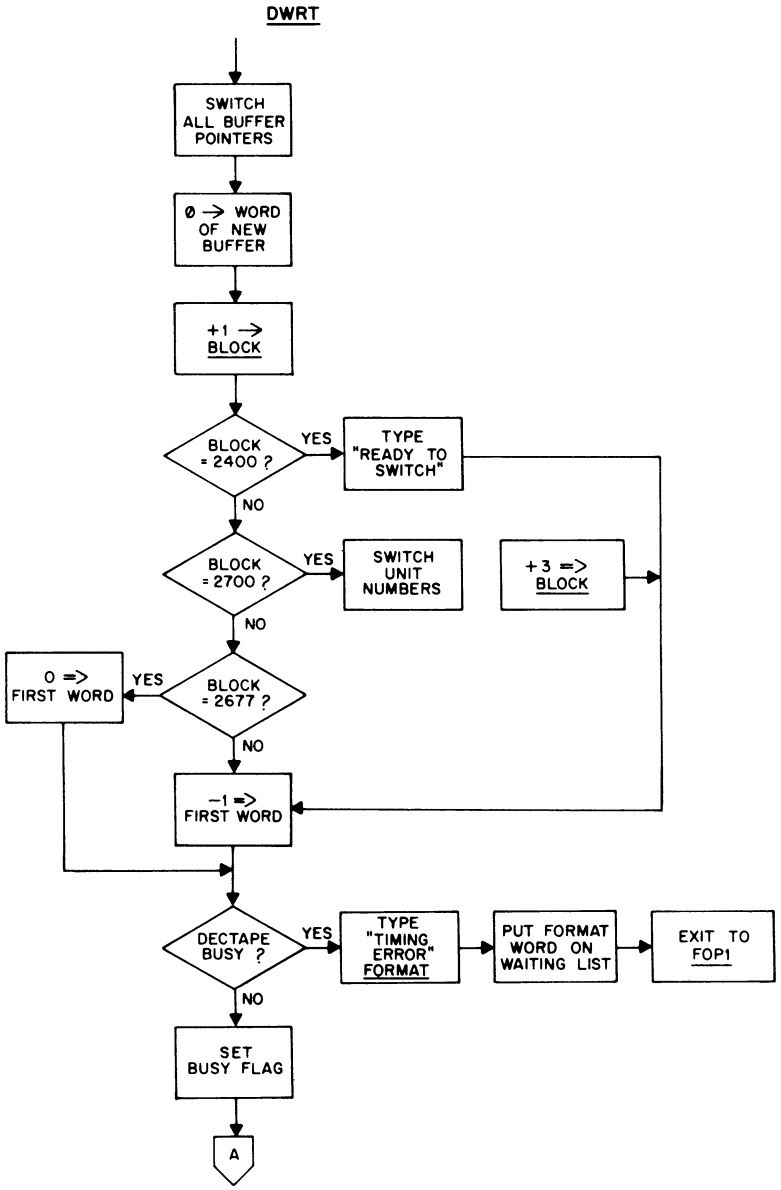


Figure A22 DECtape Write

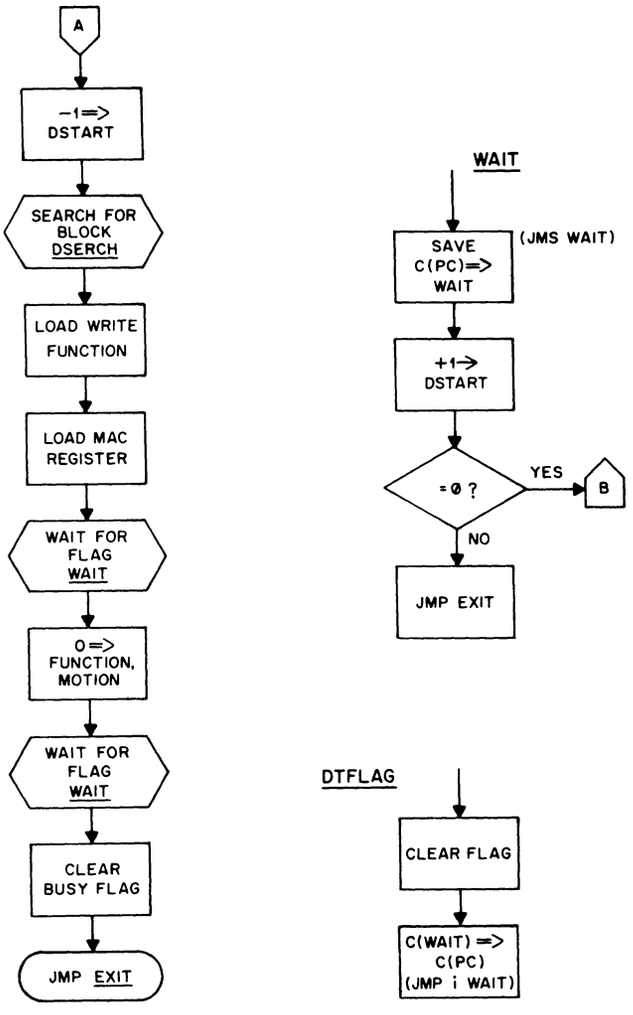


Figure A22 DECtape Write (continued)

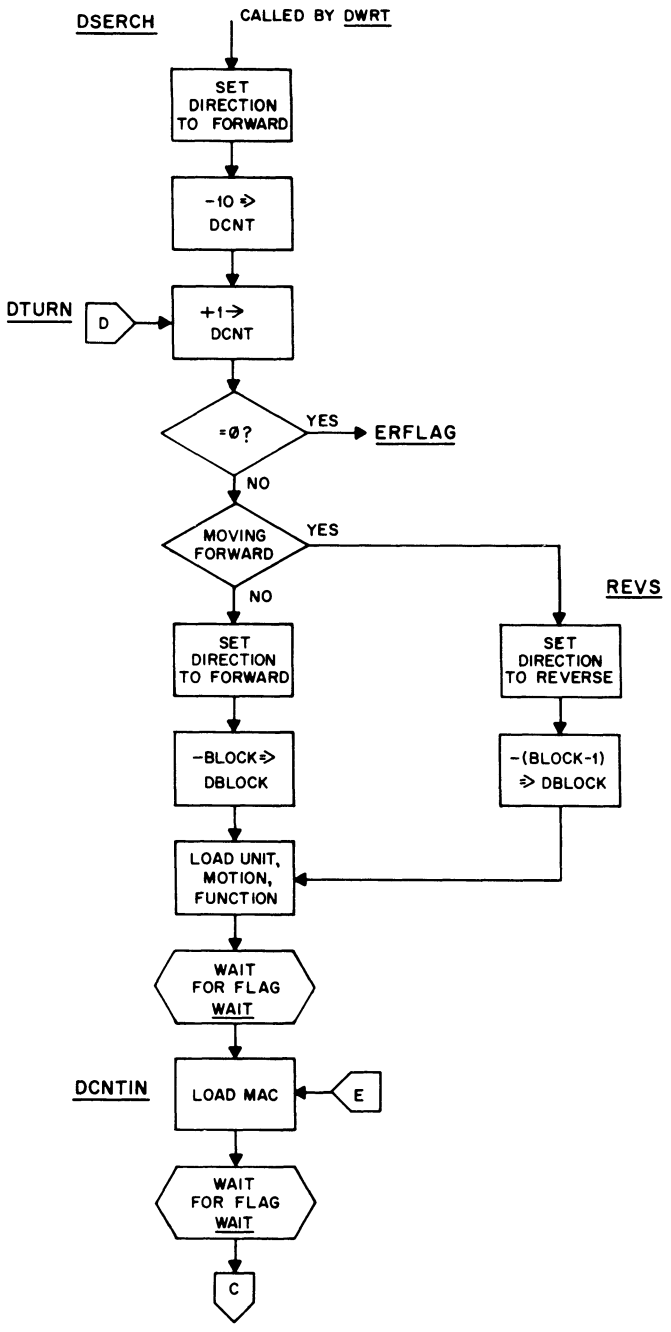


Figure A23 DECtape Search

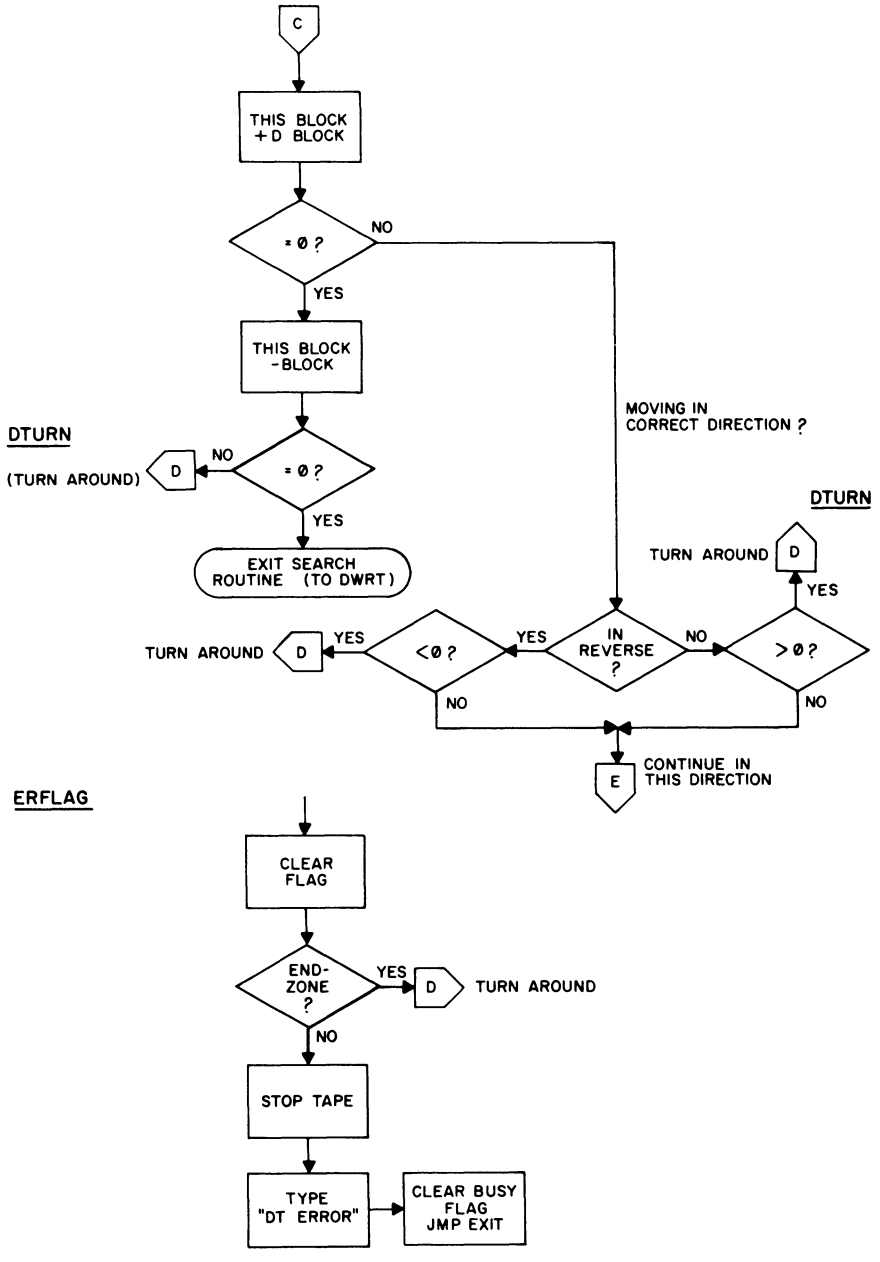


Figure A23 DECtape Search (continued)

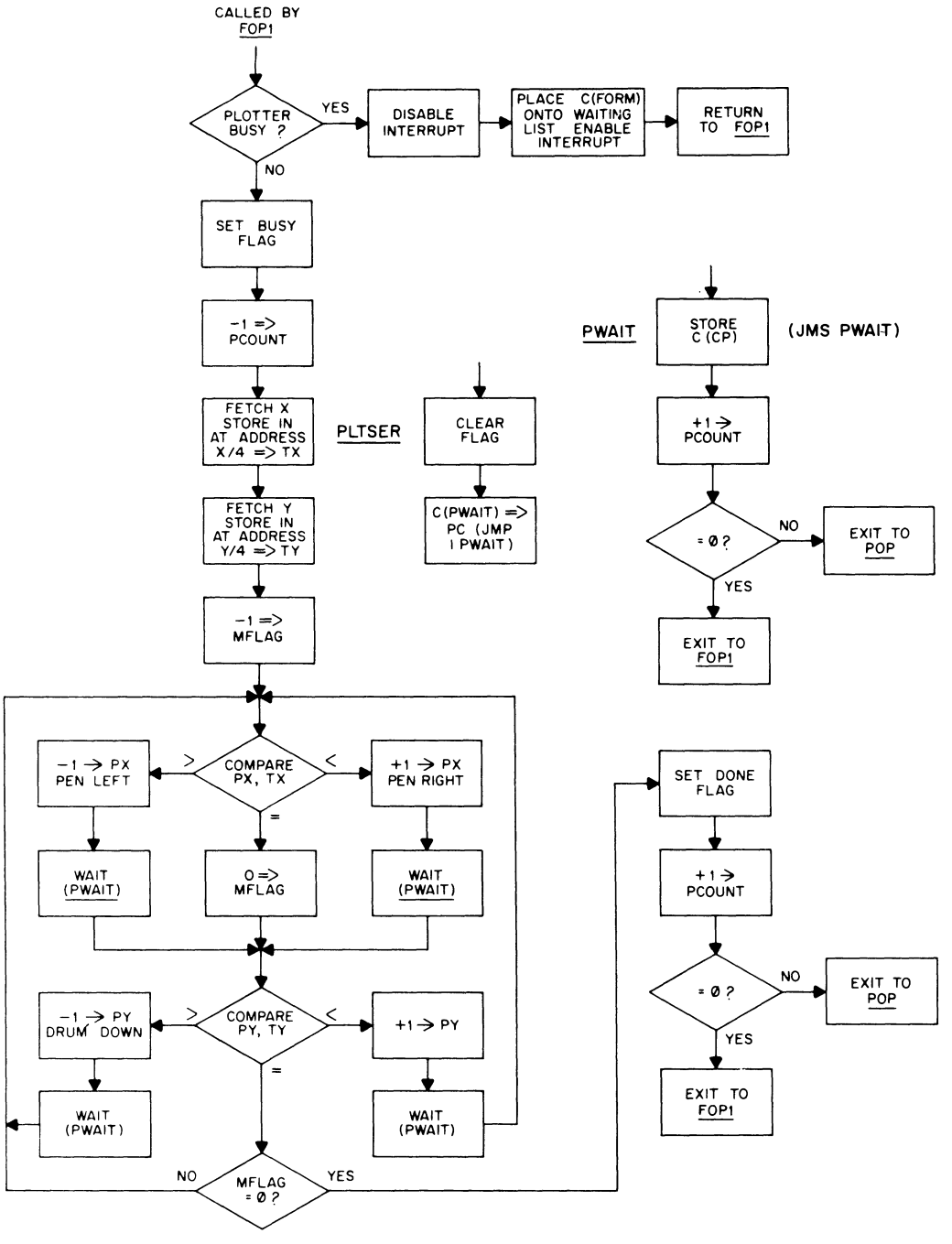


Figure A24 Plotter Format Routines

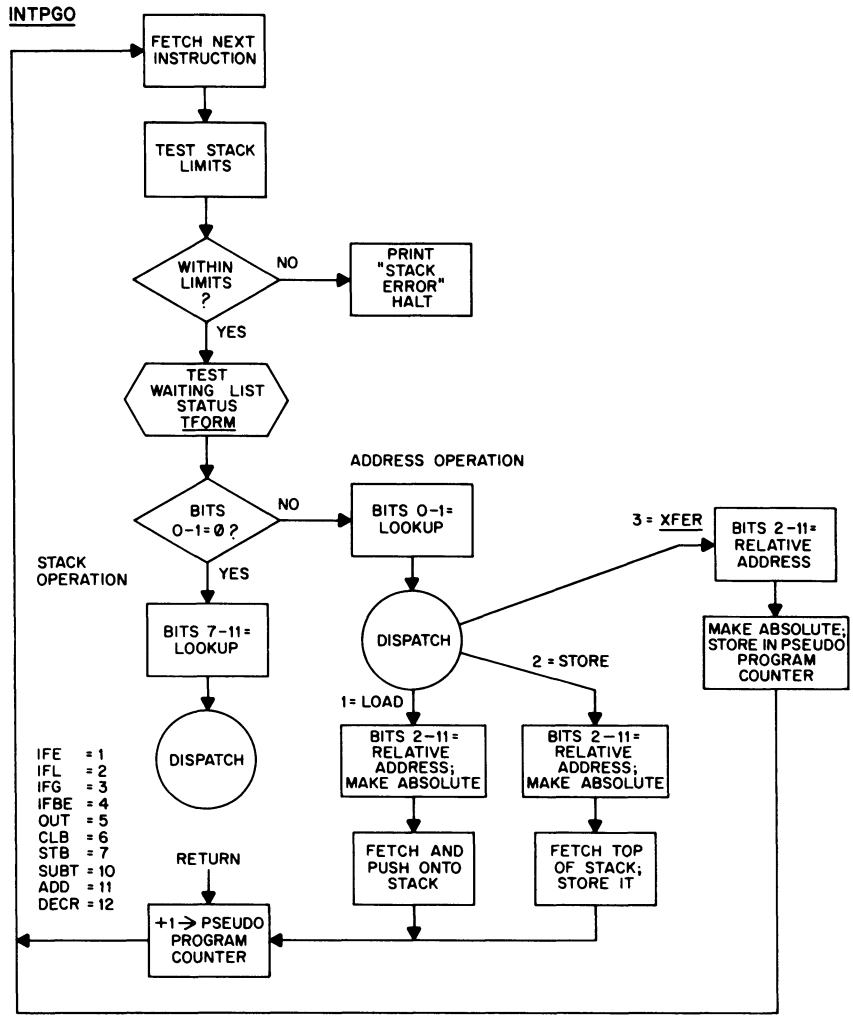


Figure A25 Arithmetic Interpreter Interrupted by Clock and I/O Devices

APPENDIX 2

PROGRAM EXAMPLES

PROBLEM 1: A SIMPLE PROGRAMMING PROBLEM

An in situ pressure, temperature, and salinity sensing instrument is lowered into the ocean. Data is transmitted along a single conductor cable and is brought into the computer using a serial buffer input.

We want to sample the ocean in the following manner:

1. From the surface to 100 meters, record at each meter the pressure, temperature, and salinity.

2. From 100 meters to 1000 meters, record the pressure, temperature, and salinity whenever the absolute change of temperature is greater than .05°C or the absolute change of salinity is greater than .02 ‰. Also record the pressure, temperature, and salinity every 100 meters from 100 meters to 1000 meters.

Let us assume that the oceanographic sensors have the following precision; that is, unity is equal to the following:

1 unit of pressure = 1 meter
1 unit of temperature = .01°C
1 unit of salinity = .01 ‰

A program to accomplish this sampling is written as follows:

```
BUFR :     PRES, TEMP, COND
FORM:      1, PRES, TEMP, COND
[    :     OUTP (1, DCTP)
1    :     IFGR PRES, 144; GOTO 2
     :     IFGR (PRES -@PRES), 1; OUTP (1, DCTP)
     :     GOTO 1
```

```
2     :     IFLS (PRES -@PRES), 144; IFLS (TEMP -@TEMP), 5;'
            IFLS (COND -@COND), 1; GOTO 2
      :     OUTP (1, DCTP); GOTO 2
      ]
            END
```

This program says, in effect:

BUFR: PRES, TEMP, COND

Three variables named PRES, TEMP, and COND are to be sampled using the serial buffer.

FORM: 1, PRES, TEMP, COND

Three variables named PRES, TEMP, and COND are to be outputted together.

:OUTP (1, DCTP)

This says output is to be recorded on magnetic tape.

1:IFGR PRES, 144; GOTO 2

This states that if the absolute change of pressure is greater than 100 (144_8), the control of the sampling will be transferred to statement number 2; otherwise, it will go to the next line.

:IFGR(PRES -@PRES), 1; OUTP (1, DCTP)

This line states that if the absolute change of the pressure between two successive readings is greater than 1, output onto magnetic tape according to Format 1; that is, OUTP (1, DCTP) which means store data on DECtape using Format 1; otherwise, go to the next line.

GOTO 1

This says to go to statement number 1 and test the environment again.

```
2:    IFLS (PRES -@PRES), 144; IFLS (TEMP -@TEMP), 5;'
      IFLS (COND -@COND), 1; GOTO 2
```

This states that if the absolute change of pressure is less than 100 meters or the absolute change of temperature is less than .05°C, or if the absolute change in salinity is less than .02 ‰ then go to statement number 2 which begins the tests over again. Otherwise go to the next line.

:OUTP (1, DCTP); GOTO 2

This says to output data onto magnetic tape and transfer control to statement number 2. The sampling and testing procedure begins again.

END

This last instruction is self explanatory.

As shown in the above description, the computer has been programmed to make logical decisions specified by the investigator in sampling the marine environment. It also has been used as a means of storing data. In the above instance, data has been stored on magnetic tape and can be used in other programs to determine variables such as Sigma T, anomaly of specific volume, and sound velocity. Table A1 shows a portion of the calculated output from stored data on magnetic tape transport number 1 that can be run immediately after the sample program.

TABLE A1 REDUCED DATA

INPUT SOURCE? T

OBSERVED VALUES

DEPTH	TEMP.	SALIN.	SIGMA-T	DELTA-A	SOUND-VEL
0000	8.35	34.17	+26.590	+145.53	+1483.4
0001	8.28	34.19	+26.616	+143.08	+1483.2
0002	8.20	34.21	+36.644	+140.45	+1482.9
0003	8.13	34.24	+26.678	+137.20	+1482.7
0004	8.05	34.26	+26.706	+134.59	+1482.4
0005	7.98	34.28	+26.732	+132.19	+1482.2
0006	7.91	34.31	+26.766	+128.98	+1482.0
0007	7.83	34.33	+26.793	+126.38	+1481.7
0008	7.76	34.35	+26.819	+123.94	+1481.4
0009	7.69	34.37	+26.845	+121.45	+1481.2
0010	7.61	34.39	+26.873	+118.89	+1480.9

PROBLEM 2: A MORE SOPHISTICATED PROGRAM

For a better demonstration of the flexibility of this programming technique, consider the following program.

An investigator desires to use a thermistor, pressure, and conductivity chain towed from an oceanographic vessel. He will sample at the same time a telemetering buoy that transmits

data from these current meters. In addition, he desires to obtain Loran lines of position and sample the ship's speed and ship's heading. These can be summarized as follows:

1. Log the time on magnetic tape every 50 meters of distance traveled. Ship's speed is 10 knots; it will cover 50 meters in approximately 9.70 sec.

2. Sample each thermistor, conductivity, and pressure sensor in the chain every half second. This defines the program time as QUNT: 62.

3. Sample the Loran, ship's speed, and ship's heading every 2 sec, thus, L1(4, L2(4, SP(4, HEAD(4.

4. Conditional Output - Since the near-surface values of temperature and conductivity will fluctuate the most, it might be most desirable to set thresholds so that relatively large changes of temperature and salinity will be stored. However, deeper values will not change as significantly, so small incremental changes have more meaning and thus should be outputted and stored. Arbitrary values have been chosen and are shown in Table A3.

Determination of Octal Constants to be Used in Testing - In order to test the variable it must be determined to what its unit value corresponds. This is found by dividing the range of the thermistor, pressure transducer, or other device by the precision of measurement; thus if the range of the thermistor is 20°C and the precision of measurement is 1 part in 2000, then each unit equals .01°C.

TABLE A2 SUMMARY

Variable	Range	Precision	UNITY Corresponds to
Thermistor	20°C	1:2000	= .01°C
Pressure	100 meters	1:2000	= .05 meters
Conductivity	20 ‰	1:2000	= .01 ‰
Vane	360°	1:120	= 3°
Compass	360°	1:120	= 3°
Rotor			= 1 centimeter per second

5. General Output Requirement

Every variable should be recorded on magnetic tape if the specified conditions are met.

Plot T0 versus S0 if it is recorded.

Type Current Meter Data in Decimal if it is recorded.

Punch T0, P0, and S0 if they are recorded.

By outlining the problem, the investigator will have thresholds established for recording changes in the variables listed in Tables A3, A4, and A5. Figure A26 shows the equipment needed to do the work.

The program listing to sample these variables and record those which exceed the established thresholds is given below.

```
ADCV:    T0(0), T1(1), T2(2), T3(3), T4(4), T5(5), P0(6),'
         P1(7), P2(10), P3(11), P4(12), P5(13), S0(14), S1(15),'
         S2(16), S3(17), S4(20), S5(21)
BUFR :   V1, C1, R1, V2, C2, R2, V3, C3, R3
DGIN:    L1(4, L2(4, SP(4, HEAD↑4
QUNT:    62
FORM:    1, T0, T1, T2, T3, T4, T5, P0, P1, P2, P3,'
         P4, P5, S0, S1, S2, S3, S4, S5
FORM:    2, T0, S0
FORM:    3, V1 , C1 , R1 , V2 , C2 , R2 , V3 ,'
         C3 , R3
FORM:    4, T0, P0, S0
FORM:    5, L1, L2, SP, HEAD, CLOK
         <5, DCTP, 22>
[    :   OUTP(1, DCTP); OUTP(2, PLDT); OUTP(3, TYPE); OUTP(4, PNCH)
3    :   IFEQ (V1+C2), 3; GOTO 1
         IFEQ (V2+C2), 2; GOTO 1
         IFEQ (V3+C3), 1; GOTO 1
         IFLS R1, 12; IFLS R2, 12; IFLS R3, 12; GOTO 2
1    :   OUTP (3, TYPE)
2    :   IFLS (T0 - @T0), 12; IFLS (P0 -@P0), 12; IFLS (S0 - @S0), 5;'
         IFLS (T1 -@T1), 7; IFLS (P1 -@P1), 12; IFLS (S1 -@S1), 5;'
         IFLS (T2 -@T2), 5; IFLS (P2 - @P2), 2; IFLS (S2 -@S2), 4;'
         IFLS (T3 -@T3), 3; IFLS (P3 -@P3), 6; IFLS (S3 -@S3), 3;'
         IFLS (T4 -@T4), 2; IFLS (P4 -@P4), 4; IFLS (S4 -@S4), 2;'
         IFLS (T5 -@T5), 2; IFLS (P5 -@P5), 2; IFLS (S5 -@S5), 1;'
         GOTO 2
     :   OUTP(1, DCTP); OUTP(2, PLDT); OUTP(4, PNCH); GOTO 3
]
         END
```

TABLE A3 MULTIPLEXER A-D CONVERTER ASSIGNMENT (ADCV)
(Data Originating in Thermistor Chain)

Depth in Meters	Thermistor Variable Name	Multi-plexer Channel #	Record if Absolute Change of	Pressure Variable Name	Multi-plexer Channel #	Record if Absolute Change of	Conductivity Variable Name	Multi-plexer Channel #	Record if Absolute Change of
0	T0	(0)	.1°C	P0	(6)	.5 meter	S0	(14)	.05 ‰
10	T1	(1)	.07°C	P1	(7)	.5 meter	S1	(15)	.05 ‰
20	T2	(2)	.05°C	P2	(10)	.5 meter	S2	(16)	.04 ‰
30	T3	(3)	.03°C	P3	(11)	.3 meter	S3	(17)	.03 ‰
40	T4	(4)	.02°C	P4	(12)	.2 meter	S4	(20)	.02 ‰
50	T5	(5)	.01°C	P5	(13)	.1 meter	S5	(21)	.01 ‰

TABLE A4 SERIAL DATA INPUT BUFFER ASSIGNMENT (BUFR)
(Data Originating in Moored Current Meter String)

	Vane Variable Name	Compass Variable Name	Record if	Rotor Variable Name	Record if
Current Meter 1	V1	C1	V1 + C1 = 9°	R1	R1 > 10
Current Meter 2	V2	C2	V2 + C2 = 6°	R2	R2 > 10
Current Meter 3	V3	C3	V3 + C3 = 3	R3	R3 > 10

TABLE A5 DIGITAL INPUT BUFFER ASSIGNMENT (DGIN)
(Data Originating in Ship's Instrumentation)

	Variable Name	Time Multiple	Gray Code ?
Loran Line	L1	(4	
Loran Line	L2	(4	
Ship's Speed	SP	(4	
Ship's Head	HEAD	†4	yes

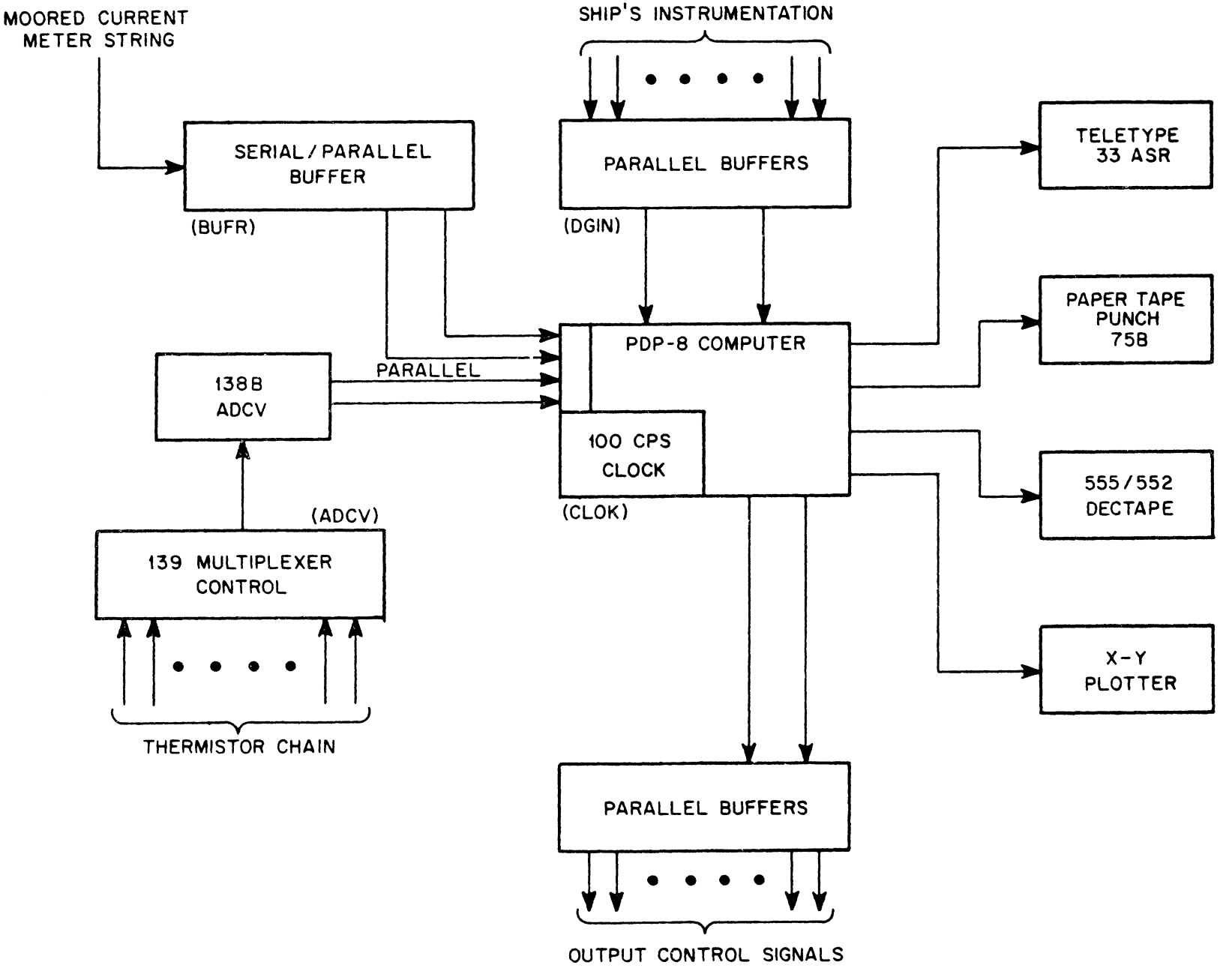


Figure A26 Equipment Configuration for Problem 2

DĀTAK ADDENDA

STANDARD DĀTAK CLOCK IOT'S

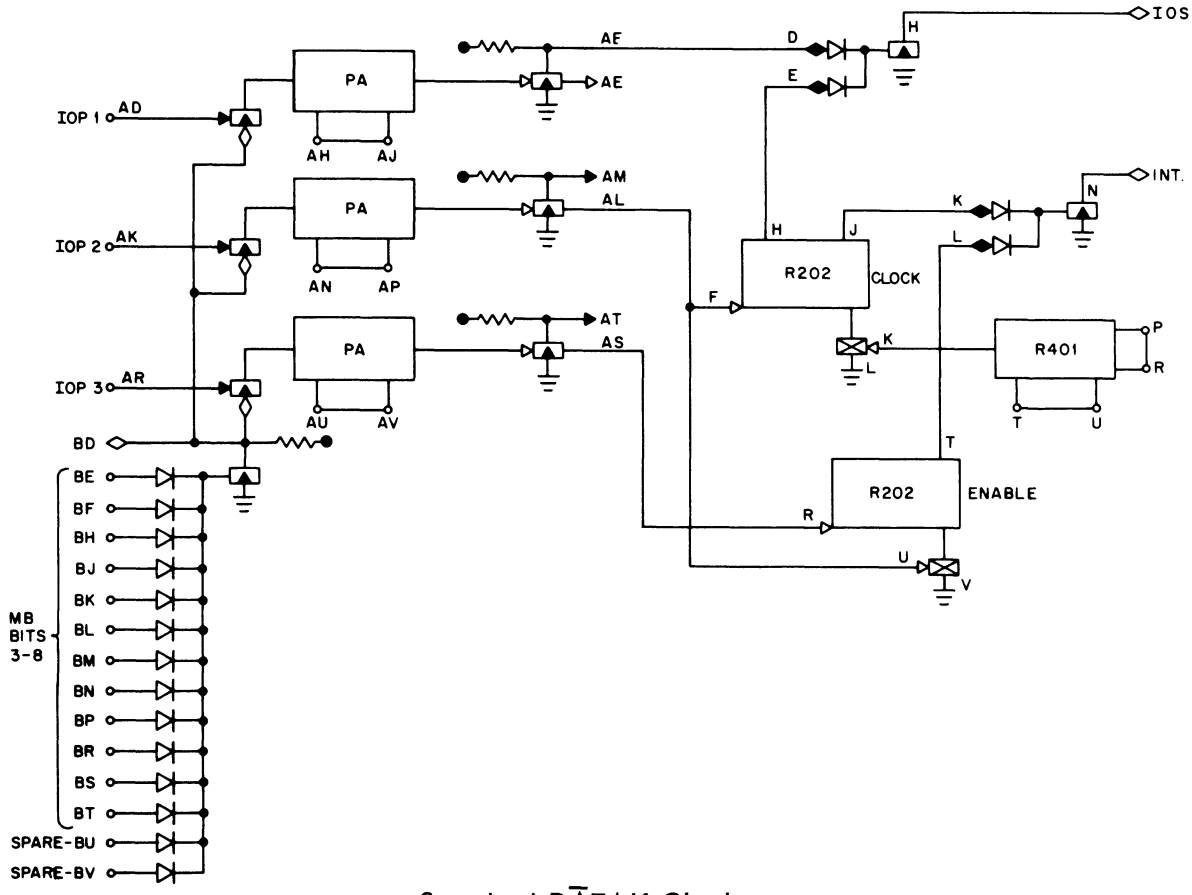
IOT 1 Skip 4 Clock flag = 0

IOT 2 Clear Clock flag and Connect Flag to interrupt

IOT 4 Disconnect Clock flag from interrrupt

INSTRUCTION LIST

1. Skip if Flag 0
2. Clear Flag and enable Clock
3. Disable Clock



Standard DĀTAK Clock

Refer to FLIP CHIP Catalog for further information about module logical design.

Washington, D. C. • Parsippany, N. J. • Los Angeles • Palo Alto • Chicago • Ann Arbor Pittsburgh • Denver • Huntsville • Orlando Carleton Place, Ont. • Reading, England • Paris, France • Munich, Germany • Sydney, Australia

5605 PRINTED IN U.S.A. 20-12/65