FIELD DATA REDUCTION SOFTWARE FOR UNIVAC AN/UYK-15 AND UNIVAC 1616 COMPUTERS: DESCRIPTION AND USER'S MANUAL

Naval Surface Weapons Center
White Oak, Silver Spring, Maryland

19 November 1975

NSWC/WOL/TR 75-68

NSWC TECHNICAL REPORT

WHITE OAK LABORATORY

FIELD DATA REDUCTION SOFTWARE FOR UNIVAC AN/UYK-15 AND UNIVAC 1616 COMPUTERS: DESCRIPTION OF USER'S MANUAL

BY
M.L. Warner
E.G. Jacques

19 NOVEMBER 1975

NAVAL SURFACE WEAPONS CENTER
WHITE OAK LABORATORY
SILVER SPRING, MARYLAND 20910

- Approved for public release; distribution unlimited

NAVAL SURFACE WEAPONS CENTER
WHITE OAK, SILVER SPRING, MARYLAND 20910

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC/WOL/TR 75-68	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Field Data Reduction Software for UNIVAC AN/UYK-15 and UNIVAC 1616 Computers: Description and User's Manual		5. TYPE OF REPORT & PERIOD COVERED Final Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) M. L. Warner and E. G. Jacques		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center White Oak Laboratory White Oak, Silver Spring, Maryland 20910		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS S4740/18723-4-301
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE 19 November 1975 13. NUMBER OF PAGES 70
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Reduction
Data Handling
Data Handling Equipment
AN/UYK-15
Computers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This software package describes a system that is used to gather and reduce a large amount of field test data. The method used to reduce the data onto a single 9-track magnetic tape is first described. Then a method is described to transform the reduced data into data that can be easily used by the engineers and scientists on the project.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

NSWC/WOL TR 75-68

19 November 1975

Field Data Reduction Software for UNIVAC AN/UYK-15 and UNIVAC 1616 Computers: Description and User's Manual

This publication should be of interest to those persons who have large amounts of field data to record and distribute to other people. This field data reduction scheme is readily adaptable to any number of inputs in any configuration. The software described herein was developed under the High Energy Laser Program at the Naval Surface Weapons Center, White Oak Laboratory, for the UNIVAC AN/UYK-15, and is upward compatible with the AN/UYK-20. This publication describes the software as it is on 19 November 1975. Although no major changes are anticipated, this software is being updated as need requires. A follow-on report that incorporates the results of operational experience is planned.

This publication is intended to aid in the use and modification of this software. Source code for this software may be acquired through the Naval Surface Weapons Center, White Oak Laboratory, Code WA-12. This work was done under Task Number S4740/18723-4-301.

CONTENTS

CONTENTS (Continued)

ILLUSTRATIONS

CONTENTS (Continued)

ILLUSTRATIONS

CONTENTS (Continued)

TABLES

1.0 INTRODUCTION

1.1 The data generated during the High Energy Laser (HEL) field test is reduced to the User's data format by two major software packages. They are the Field Data Reducer (FDR) package and the Fetch Package (FETCH). The FDR package is not only responsible for the complex data handling instrumentation system, but also responsible for extracting and merging the field test data onto a single standard nine-track magnetic tape (Mag Tape). The Fetch package, using the tape generated by the FDR, extracts the data requested by a project scientist (User). Then using the User's specified format, transfers the data onto another magnetic tape.

1.2 Both FDR and FETCH are written in a modular format for greater ease in programming and debugging. Tables 1.1 and 1.2 are a listing of the routines and subroutines used in the two packages. The FDR package is broken into two independent programs - PREFDR, which sets up the tables that reflect the configuration of the instrumentation system, and FDR, which does the data processing. FDR is itself broken into two major routines - RTDS, which sets up the data structure to be used during processing and CYCLE, which does the actual processing using subroutines to handle all discrete tasks which occur more than once, e.g., digital tape drive input and output (MTIO), program timing (CLOCK), program monitoring (MONTOR). FETCH is similarly broken into major routines: initialization (FETCHI), time and value up date (FETCHT), and calibration of the data (FETCHV). These routines make extensive use of the other subroutines listed in Table 1.2.

2.0 Field Data Reducer (FDR) Software

2.1 The programs PREFDR and FDR are responsible for describing and handling data from a complex instrumentation system that may have its configuration changed from time to time. Initially, PREFDR allows the user to describe the hardware configuration in tabular form. These tables are then saved for use by FDR. The FDR program then uses the defined hardware configuration to extract data from the input stream. The extracted data is tested for a change in value, sorted, and merged with data previously extracted. The final processed data stream is stored on a standard computer magnetic tape.

2.2 Figure 2.1 describes a typical hardware configuration. As shown in the figure the system is composed of three parts. The first part is the Pulse-Code Modulation (PCM) instrumentation system. The last partin between is an interface between the instrumentation and the computer system. Because of the completeness of the interface, the computer can control the entire data reduction process without operator assistance.

2.3 It is the purpose of the computer to generate a tape which can be used by other programs for analysis reasons. The format of this data tape and the method of implementation are the main topic of this document. The program should represent the data as accurately as possible and yet minimize the amount of storage necessary to represent that data. It is also desirable to minimize the amount of time necessary to carry out the above process. Therefore, a program which handles many types of data and stores these data on a minimum of tape in a minimum amount of time is desired.

TABLE 1.1

FDR ROUTINES

PREFDR (Preliminary setup for Field Data Reducer) - sets up tables reflecting the instrumentation system configuration

FDR (Field Data Reducer) - master program for data reduction

RTDS (Real Time Data Structure) - establishes the data structures used in FDR

IBR (Input Buffer Routine) - system loading label for IBI and IBU

IBI (Input Buffer Initialization) - sets up buffer data structure

IBU (Input Buffer Update) - determine which input data to process, checks its status and attaches the time at which the date was gathered to the data block

PCMIO (PCM input/output) - master routine for controlling the input data stream

CYCLE - does the real time processing of input data

STRIP - real time output of data onto the electrostatic printer instead of magnetic tape

MONTOR - Monitors and displays program status and allows the operator to interact with the FDR program

COMON (Common) - establishes a common storage area for linkage between FORTRAN and ULTRA routines

CONVRT - converts from binary to decimal and decimal to binary for several coding schemes (see 4.13.2)

CLOCK - real time clock control

LOGIC - supplies logic functions not available in the UYK-15 FORTRAN (see 4.13.4)

MTIO - real time magnetic tape input/output control routine

SCAN - assembly language routine which scans parameter strings

FUN (Find Unit) - retrieves the UNIT identification and characterization information associated with a channel

FNID (Find IDENT) - retrieves the IDENT identification information associated with a UNIT

FNCOD (Find CODE) - retrieves the data value for a particular CODE in an IDENT

WB - control routine for the Wide Band analog recorder

SM - control routine for the Switch Matrix

DC - routine to maintain the program Digital Clock and control the time code reader

Table 1.1 (cont'd)

S (synchronizer routine) - monitors and controls the bit and frame synchronizers

UNITP (Unit Processor) - returns a list of all IDENTS associated with the specified UNIT

IDP (Ident Processor) - returns a data value for each CODE in an IDENT

MRGR (merger) - merges input data for output on a single magnetic tape

MIUN (Merge input unit)

MOUN (Merge output unit)

TABLE 1.2

FETCH SUBROUTINES

FETCHI (FETCH initialization) - reads the data description tables from the IDATA tape

FETCHT (FETCH Time) - maintains the input data time

FETCHV (FETCH Value) - calibrates data values and outputs them in user's format

PRINT1 - routine for printing tables

PRINT2 - routine for printing user codes and hardware descriptions

MTIOF - magnetic tape input/output routine

MTIPE - checks for parity errors and controls data stream through MTIOF

DBLWRD - handles double word addition, subtraction, and storage

DFLOAT - changes a double word integer to floating point

IFIELD - removes and right justifies a specified bit field

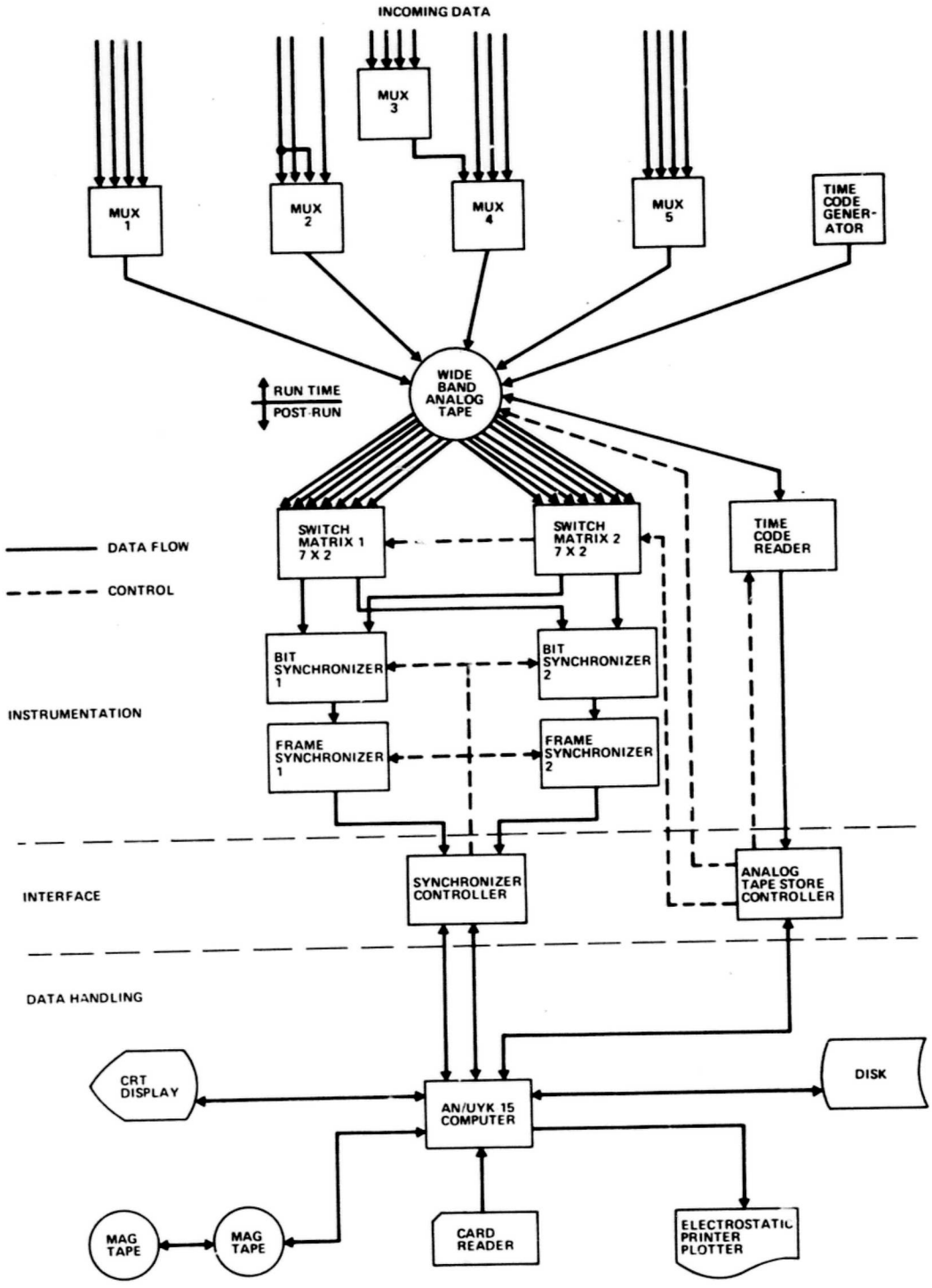


FIG. 2.1 A TYPICAL HARDWARE CONFIGURATION

2.4 One further consideration is that since the nature of an instrumentation system is to assist in diagnosis and documentation of a series of events, the data requirements will certainly change as time passes. Therefore, tremendous flexibility is needed in defining the data. The program should allow for changes to be made in the incoming data without changes to the actual software programs. With such a program, the users may easily change the characteristics of the incoming data with full knowledge that these changes can be accounted for by the software.

2.5 The data flow of the instrumentation/data handling system will now be described. Data from different types of sensors are multiplexed, digitized and converted into a PCM format (MUX). The MUX data are then recorded on one of 13 tracks of a wide-band (WB) analog recorder. The 14th track has a time code recorded on it from the Time Code Generator. The above events occur during a test run. The WB analog tape recorder bandwidth is necessary to record the widest of the MUX bandwidths. After the run has been completed, the instrumentation system is able to begin reduction and storage of the recorded data. The analog recorder can be played back at numerous speeds. The idea is to run the analog recorder back as fast as possible without overloading the computer. The switch matrix routes any channel of data to one of two bit synchronizers. The bit synchronizer will preprocess the WB information for use by the frame synchronizer. The frame synchronizer then converts the PCM stream into data words. This data is read into the computer, merged and reduced, then written on the digital tape for storage. This sequence continues until all the data on one track of the WB is exhausted, thus completing one pass. As the data from each pass is written on the digital tape, it is merged with data from all previous passes. The above process is continued for each pass until data is exhausted. The final pass tape, containing all of the data, may then be used for analysis purposes.

2.6 The data format chosen to minimize tape usage was to record data changes only when they occur. To protect a tape from a parity error and to allow searching, an absolute data value is stored on tape at regular intervals. The above recording methods will be referred to by the names CHANGE BLOCK (for the data change) and FIXED BLOCK (for the absolute data).

2.7 To enhance execution time of the data formating process, all of the routines involved in the real-time process were written in assembly language. Because many data items are sampled by the same multiplexer, all of these data items have the same characteristics. Thus by grouping data items, the execution time of the program can be minimized, since they all have the same characteristics when processed.

2.8 With the above ground rules, this report will describe an implementation of this data management and reduction scheme into a real-time program.

3.0 PREFDR Program Description

3.1 The instrumentation system, described in the Introduction to this manual, by necessity, must be allowed to change without changing the software. To accomplish this, a series of tables are used to describe the entire hardware setup of the instrumentation system. By modifying these tables, one may reflect

a hardware change in the system. PREFDR allows a user to modify these tables in as simple as possible a method. The program also checks for bounds on the parameters and number of parameters fed in by the user. Once the user thinks all of the tables have been set up properly, PREFDR checks linkage between tables to assure that there are no deficiencies. PREFDR allows the user to generate tables from scratch, modify existing tables, list tables, and output tables to a magnetic tape.

3.2 Figure 3.1 represents the tables needed to describe the hardware. The first table, HEADER, contains the test number and comments on the test and test conditions. The test number may have up to 20 alpha numeric characters. The remaining 20 items will be numeric characters which may be used by the test director to list informative test conditions.

3.3 The other tables describe the data flow and characteristics. Data items are called CODES; groups of CODES are called IDENTS; groups of IDENTS make up UNITS; and a UNIT is connected to a particular TRACK on the WB analog recorder; and finally a TRACK is fed into a computer channel (CHAN). With this series of tables, data can be described, grouped, and routed for input to the computer.

3.4 The channel table (CHTAB) contains a list of analog recorder track numbers (TRACK). This table defines the order in which the data should be extracted from the analog tape. Two consecutive table entries of zero means no more TRACK assignments.

3.5 The TRACK table (TRKTAB) defines which multiplexer (UNIT) is connected to a particular analog tape track. A UNIT equal to zero terminates the table.

3.6 The unit table (UNTAB) contains a complete description of a particular unit, i.e., multiplexer. This information includes a designation number (UNIT); the number of frames sampled per second (RATE); the number of bits per word (BPW); the number of words per frame (WPF); the number of least significant bits (LSB); a 32 bit frame sync. pattern (FSP) (for less than 32 bits the pattern is right justified and leading zeros inserted to fill the storage location); the number of significant bits in the FSP (BPFSP); and the alternate complement code word (ACC). The ACC words high order bit (15) declares that the FSP will be complemented every other frame if it is high (equals 1); otherwise no complement operation will be performed on FSP. The low order three bits of the ACC word (0-2) describe the format of the data from the analog recorder. A zero entry in UNIT terminates the table.

3.7 The ident table (IDTAB) defines group designators for data. By grouping data items that are similar in characteristics, i.e., word size, sample rate, it is hoped to minimize the size of the user's tape. The identifier (IDENT) designation is defined in this table along with the UNIT it represents and a special IDENT processor flag (IPROC). A zero entry in IDTAB terminates the table.

3.8 The code table (CODTAB) defines the characteristics of the data items (CODE). Information contained in the table is a code designator (CODE); an IDENT reference, a super/sub multiplexing descriptor (SS) (see Figure 3.2), the word within the data frame which represents the CODE (WORD); and a calibration pointer (CAL). If SS is zero, multiplexing is performed. If SS is positive, the low order byte

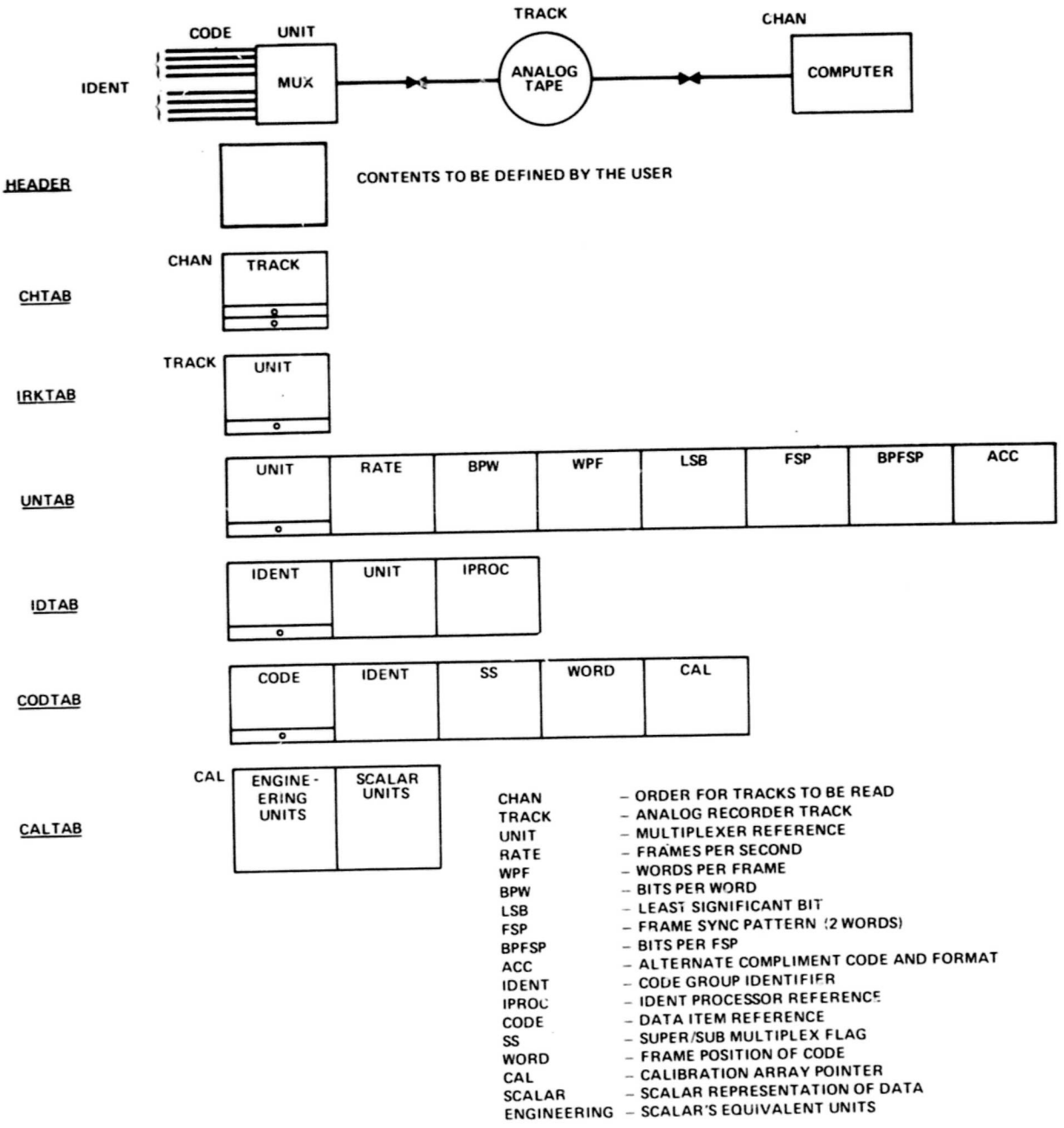


FIG. 3.1 THE FDR TABLES

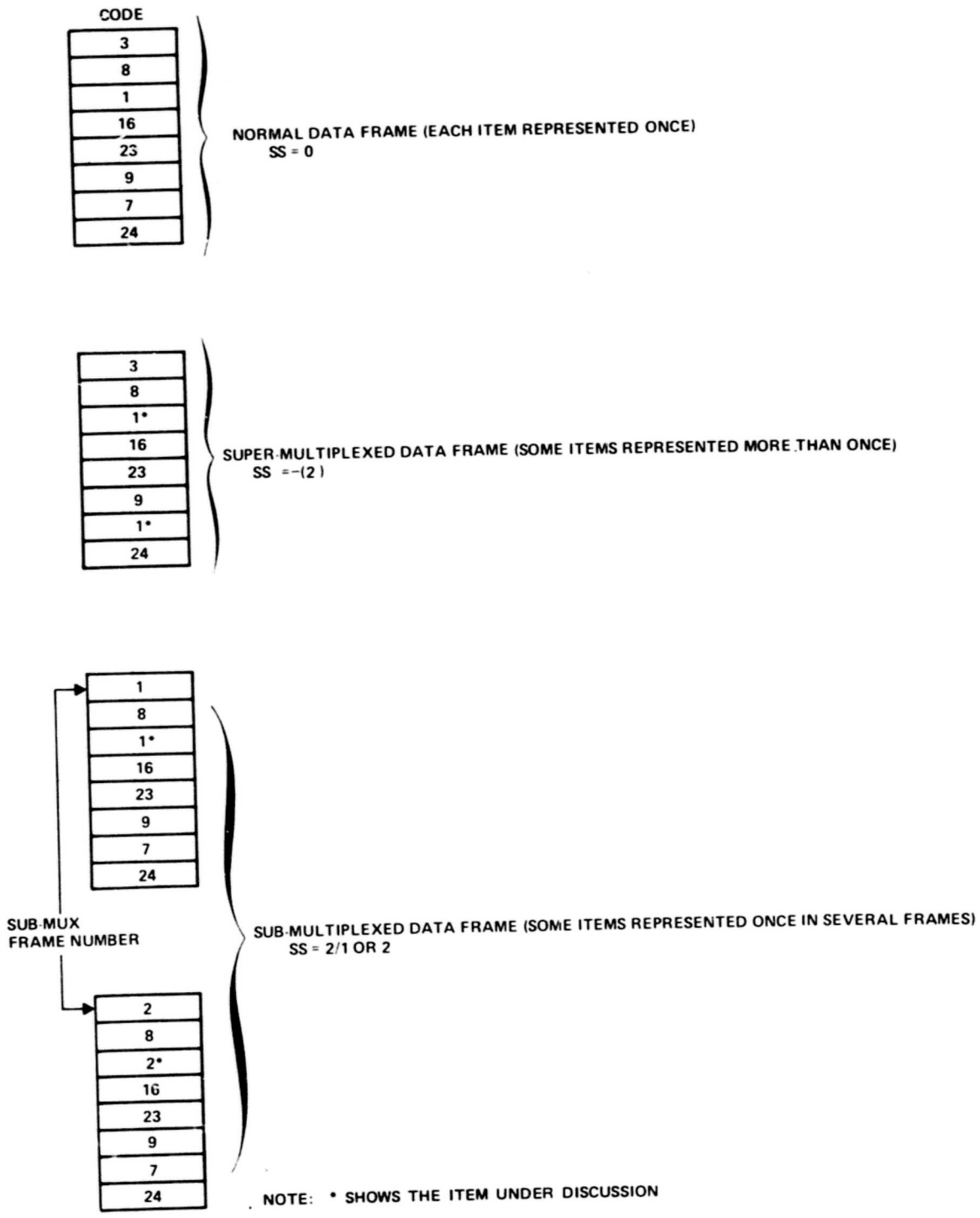


FIG. 3.2 MULTIPLEXING DESCRIPTORS

describes which frame the code occurred in and the high order byte describes the number of frames in a sub-multiplexed frame. If SS is negative, the absolute value of SS is the number of super multiplexed samples in a frame. The first zero entry in CODTAB terminates the table. If CAL is zero, the data is discrete; otherwise, CAL is a pointer to CALTAB.

3.9 The calibration table (CALTAB) contains variable length lists of calibration data. The end of a list is detected by a decreasing scalar value (Figure 3.3). The lists are the digital representation (SCALAR) of analog values (ENGINEERING) at a given point. Thus by using N entries of CALTAB, an N point calibration can be performed on the data (Figure 3.3). By using linear interpolation, a scalar number can be converted into a number of engineering units. The engineering units may be volts, degrees, pressure, etc. Without a calibration table, the digital representation of analog information may have no real-world significance.

3.10 Operationally, the user has a choice of nine commands (see Section 5.2), which may be typed into the keyboard. If PREFDR detects the proper number of parameters and the correct ranges for each parameter, it accepts that call and modifies the tables appropriately. If an error is detected, PREFDR generates an error message. There are calls which allow the user to exit the program, list the tables on the printer, and output the tables to a tape.

3.11 The programs necessary to load PREFDR are as follows:

PREFDR - Main FORTRAN routine which edits tables.

SCAN - Ultra routine which scans parameter strings.

4.0 FDR Program Description

4.1 FDR is the main data handling program for the instrumentation system. It is composed of many routines and executes in two phases, whose main distinction is the operational environment. The first phase is a process of setting up tables, data structures, initialization, etc. This phase is not time critical and can be written in FORTRAN. The second phase is the actual real-time data handling process. Because this process is time critical, it is written in assembly language. Together these two phases take turns setting up the program properly for a particular pass over the WB analog recorder, then extracting data from the WB recorder for merging and storage.

4.2 The first phase may simply be described as bookkeeping. Common linkages between assembly routine and FDR are first established. Program variables are next initialized. Then operator interaction begins via the keyboard. The operator may select a time window on each track of data for playback. Next the operator can select the number of on-line Bit/Frame synchronizer combinations. The pass number can also be specified in cases where the operator wishes to restart a partially completed run. Next the operator specifies the record and playback speeds of the analog recorder. From this point the operator specifies whether he wishes to strip out data, or store the data on Mag Tape. In the case of storing the data, the FDR then proceeds setting up the run. If strip out is desired, the operator must specify the time scale and range of the data to be stripped out.

CALTAB

CALIBRATION INDEX	SCALAR VALUE	ENGINEERING VALUE	
1	0	-100	2 POINT CALIBRATION
2	255	+100	
3	0	-500	3 POINT CALIBRATION
4	128	0	
5	255	+500	
6	0	-100	6 POINT CALIBRATION
7	10	0	
8	100	+10	
9	150	+20	
10	200	+30	
11	255	+40	
12	0	+50	4 POINT CALIBRATION
13	100	+60	
14	200	+70	
15	255	+75	
16	0	-0.05	9 POINT CALIBRATION
17	200	0	
18	400	+0.05	
19	800	+1.0	
20	1600	+2.0	
21	2000	+3.0	
22	2500	+5.0	
23	3000	+7.0	
24	4095	+10.0	

FIGURE 3.3 THE CALIBRATION TABLE

4.3 The first phase then continues by clearing the synchronizers, rewinding the analog recorder, and rewinding both digital tape units. FDR assigns the input and output digital tape units for the particular pass. Figure 4.1 shows the input/output configuration for various passes. If a termination flag is set, the program returns to the operating system at this point after outputting the unit number of the unit holding the user's tape to the operator. Otherwise the FDR program assigns digital input and output units as a function of which pass is in process. The output tape is now rewound for table processing. The tables are read in from the input unit and output to the output unit. If an error occurs it is displayed to the operator.

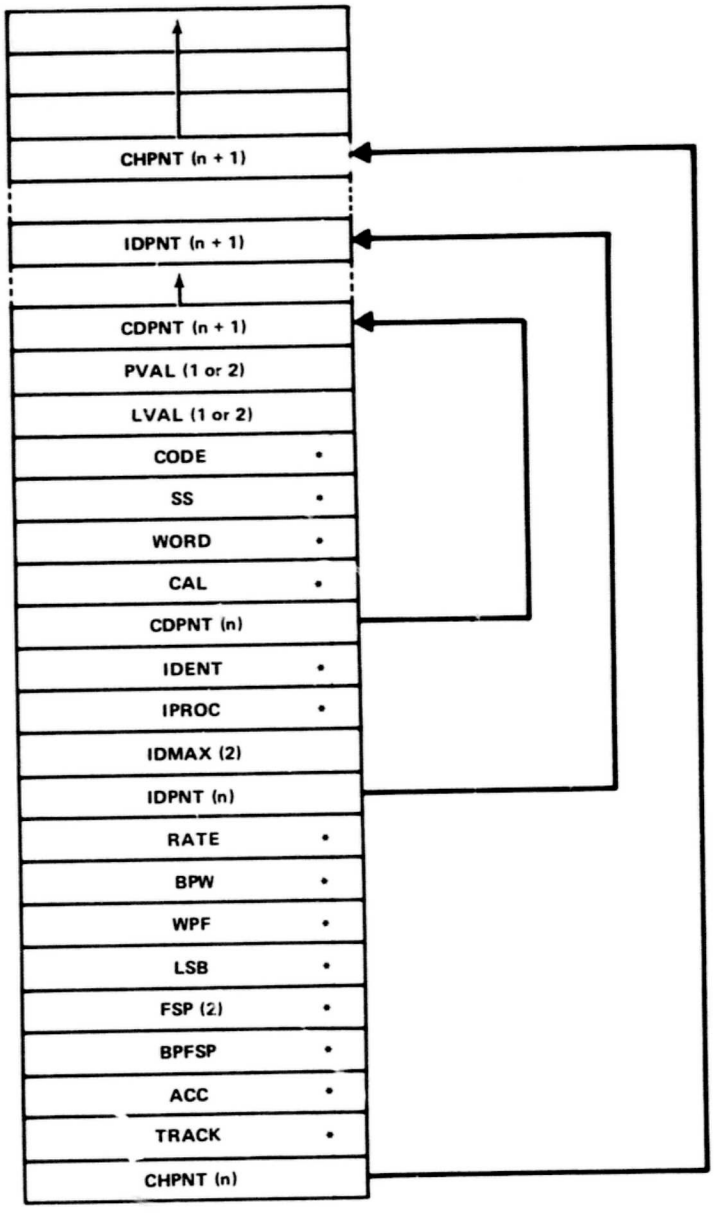
4.4 At this point FDR sets up a data structure describing the data flow for each Bit/Frame synchronizer combination. This data structure will later allow the real-time routine to determine how to input a data frame and reformat the data and where to store the data. Now FDR also sets up the input buffers so that the data for each Bit/Frame synchronizer combination may be read into core in a continuous manner. A monitor program for system status check and display and operator intervention is also initialized at this time. FDR then initializes the strip-out routines if that option is requested. Up until this point FDR has been doing general program initialization and computer peripheral setups. From this point FDR begins to initialize instrumentation units.

4.5 FDR begins setting up the instrumentation units by setting up the Bit/Frame Synchronizer combinations, setting the switch matrix, then making sure the Time Code Reader is in remote. Then a high-speed search is started on the analog recorder for a time several seconds before the earliest time on the tracks to be read. When this point has been reached, the time code reader's input filter is set up for the proper bandwidth. Now the analog recorder is turned on for a read. When the recorder has phase locked, the time code reader starts reading time. When the reader has no errors, the Bit/Frame synchronizer combinations are checked for sync. errors. If there are no sync. errors, read operations are commenced on each Bit/Frame synchronizer combination.

4.6 FDR then calls the real-time data processor (CYCLE and STRIP), which handles data until the data on all input channels has been exhausted. Upon returning from the real-time data processor, FDR checks the status of the process. If the process was stopped because of a data merging backlog, a read on the recorder is restarted and the real-time process is again started. If the program cannot keep up with the incoming data (data bound) the entire process is restarted at the next slower WB tape speed. Other statuses represent normal return or operator intervention. The operator can terminate a pass early, he can abort the entire process and restart from the beginning of FDR, or he can restart the latest pass.

4.7 The second phase of FDR involves the use of several assembly language subroutines. These routines shall be discussed individually. Their purposes are for table formation/table lookup (RTDS), data base management (IBR), device control (PCMIO), real-time data processing (CYCLE and STRIP), process monitor (MONTOR), and various utility functions (COMON, CONVRT, CLOCK, LOGIC and MTIO).

4.8 RTDS consists of a major routine and several subroutines used to manage the formation and use of the data structure used during program execution. Using its calling parameters - a frame sync. number, and a table pointer - RTDS will use the



*SAME DEFINITION AS PREFDR TABLES

CHPNT - POINTER TO NEXT CHANNEL IN DATA STRUCTURE
IDPNT - POINTER TO NEXT IDENT IN DATA STRUCTURE
CDPNT - POINTER TO NEXT CODE IN DATA STRUCTURE
IDMAX - TIME OF NEXT COMPLETE UP-DATE OF IDENT
LVAL - LAST VALUE OF CODE
PVAL - PRESENT VALUE OF CODE

FIG. 4.1 THE DATA STRUCTURE FOR FDR

information contained in the tables generated by PREFDR to form a real-time data structure. The data structure is built in such a way as to allow ease of use once it is entered. Figure 4.1 is a simplistic representation of the data structure. The synchronizer channel number of interest is found by linking n (channel number) times through the channel pointers (CHPNT). If a zero is encountered, no more channels are connected. Once a channel is located, the next nine items encountered describe the multiplexer. After this is a series of IDENT pointers. The next four items after the IDPNT describe the IDENT. The CDPNT follows linking all of the codes associated with a particular IDENT. Within RTDS are several routines which are used to scan the data structure. The routines are:

a. FUN, which discloses the UNIT information associated with the channel under consideration.

b. FNID, which discloses the IDENT information associated with the UNIT under consideration, and

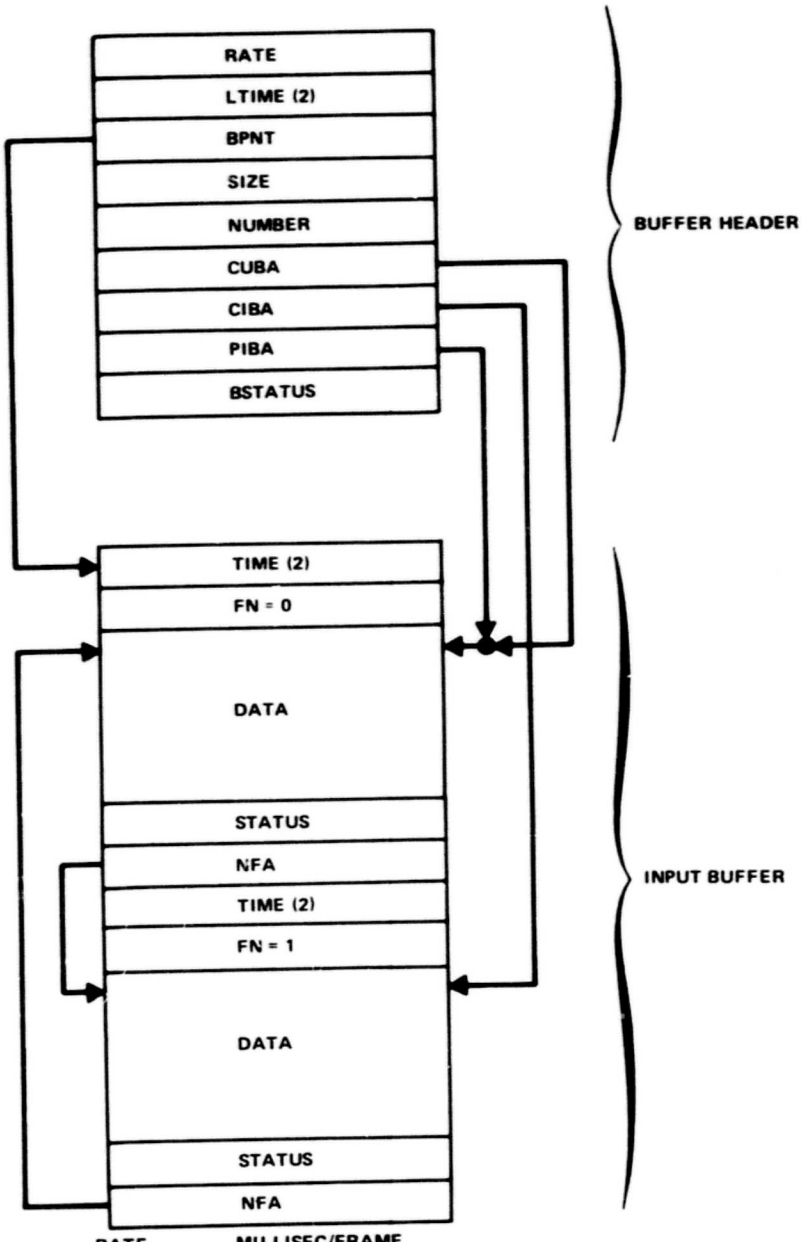
c. FNCOD, which discloses the CODE information associated with the IDENT.

When any of these routines returns a zero value, that particular information is exahusted (i.e., a zero pointer). It may be noted that the data structure is up-side-down (Figure 4.1). This is due to the core optimization routine that assigns unused core as input buffers. RTDS also modifies the PREFDR Table CODTAB by adding to the WORD column a high order byte which indicates the position within an IDENT that that code represents.

4.9 The next set of routines involves data base management (IBR) of the input buffers. Because the host computer has a very powerful I/O structure, adopting a data structure (Figure 4.2) for the input buffers allows the computer channel to fill a buffer, then automatically go to the next input buffer without program intervention. As shown in Figure 4.2 a storage area of about 2000 words is divided into a header and individual buffers. Each buffer contains one frame of data from a frame synchronizer and is linked to the next buffer in succession by a pointer so that the buffers form a continuous loop. The buffer header is a small ten word table which allows IBR to keep track of happenings and point to the input buffer area. The input buffer frame pointers allow the I/O channel to continuously run with no program intervention.

4.9.1 IBR consists of two routines: IBI and IBU. IBI sets up the input data structure. IBU finds the next unprocessed input buffer, computes current status, checks if the program is data bound, and checks for data termination. IBU data decides which input buffer should be processed provided more than one channel is connected to the computer.

4.10 The next necessary routine is for device control (PCMIO). This routine is used to initialize and operate all non-standard devices connected to the host computer. The devices controlled and their respective control routines (parentheses) are the Wide Band Analog Recorder (WB), the switch matrix (SM), the digital clock (time code reader) (DC), and the bit and frame synchronizers (S). If PCMIO initiates operations on the above devices and the operation has not been completed within one second, an error is output to the operator.



RATE	-	MILLISEC/FRAME
LTIME	-	TIME OF LAST BUFFER PROCESSED
BPNT	-	POINTER TO BUFFER AREA
SIZE	-	NUMBER OF DATA WORDS
NUMBER	-	NUMBER OF LINKED INPUT BUFFERS
CUBA	-	CURRENT UNPROCESSED BUFFER ADDRESS
CIBA	-	CURRENT INPUT BUFFER ADDRESS
PIBA	-	PREVIOUS INPUT BUFFER ADDRESS
BSTATUS	-	INPUT BUFFER STATUS
TIME	-	TIME DATA READ
DATA	-	VALUES OF INFORMATION READ
STATUS	-	STATUS OF SYNCHRONIZERS
NFA	-	NEXT FRAME ADDRESS
FN	-	FRAME NUMBER

FIG. 4.2 THE INPUT BUFFER DATA STRUCTURE

4.10.1 WB controls the analog recorder by sending a command to it then reading its status. If the command and status do not agree, the command is reinitiated until it is complied with, then the status is returned to the calling routine.

4.10.2 SM is controlled by giving it the number of channels connected. SM then calls FUN to determine what track is associated with a particular channel and outputs the proper commands to achieve the desired configuration. No status is returned to the user from SM.

4.10.3 DC has several modes of operation. DC can return an immediate time from the clock. It can cause the time to be updated whenever the time changes. DC will continue updating the time until another operation is initiated on its channel. It can setup the time code Reader's input filter (This is necessary when the playback speed of the WB is different from the record speed.). Finally DC can master clear the controller.

4.10.4 S, the final section of PCMIO, concerns the synchronizers. This section can request status from the Bit/Frame synchronizer and master clear the controller. This section can also write instructions to the bit and frame synchronizers by making calls to FUN for the necessary information. Finally, a read operation can be initiated on a channel. This causes a continuous read of data into the computer input buffers described in IBR. This read operation can only be terminated by executing another function on that channel. Otherwise the channel will read as long as data is available and DC will continue the time updating until another operation is initiated on its channel.

4.11 The routine responsible for the incoming real-time data is CYCLE, which contains several subroutines which perform operations on the data (UNITP, IDP, MRGR, MIUN, and MCUN). CYCLE together with a routine for displaying data on a raster scan device (STRIP) perform the actual data handling functions. CYCLE calls IBU for an unprocessed buffer and its status, checks the time of the buffer against the time interval during which the data should be processed, checks system status for operator or abnormal happenings, and checks the status of the actual frame of data. If there is no data available MONTOR is called and a system status is displayed. If data is available, CYCLE calls FUN with the channel the data was retrieved from making all of the UNIT parameters available. UNITP is then called and proceeds to breakdown the data frame into IDENTS by calling FNID. Once FNID is called, the ident is processed using IDP. IDP then breaks the IDENT into CODE's using calls to FNCOD. IDP accumulates information for each CODE in an IDENT. Then if the accumulated data block (one frame) is to be stripped out, STRIP is called; otherwise, the data is given to a routine which merges input data with the BLOCK and outputs the resulting data (MRGR). If the input data is exhausted and no strip out was being performed any data remaining on the input unit is copied onto the output tape unit. MRGR operates on the data blocks by comparing their respective times. The result is an output stream of blocks with monotonically increasing times associated with them. MRGR also decides whether a data BLOCK should be represented as an actual value (fixed block) or as a change from the previous value (change block). The fixed block assures that if a parity error occurs, a full value of the data is available at regular intervals. MRGR also generates a special time block containing the absolute time of day. The resulting output tape format is described in Figure 4.3. The tape format is the same no matter how many passes are necessary to

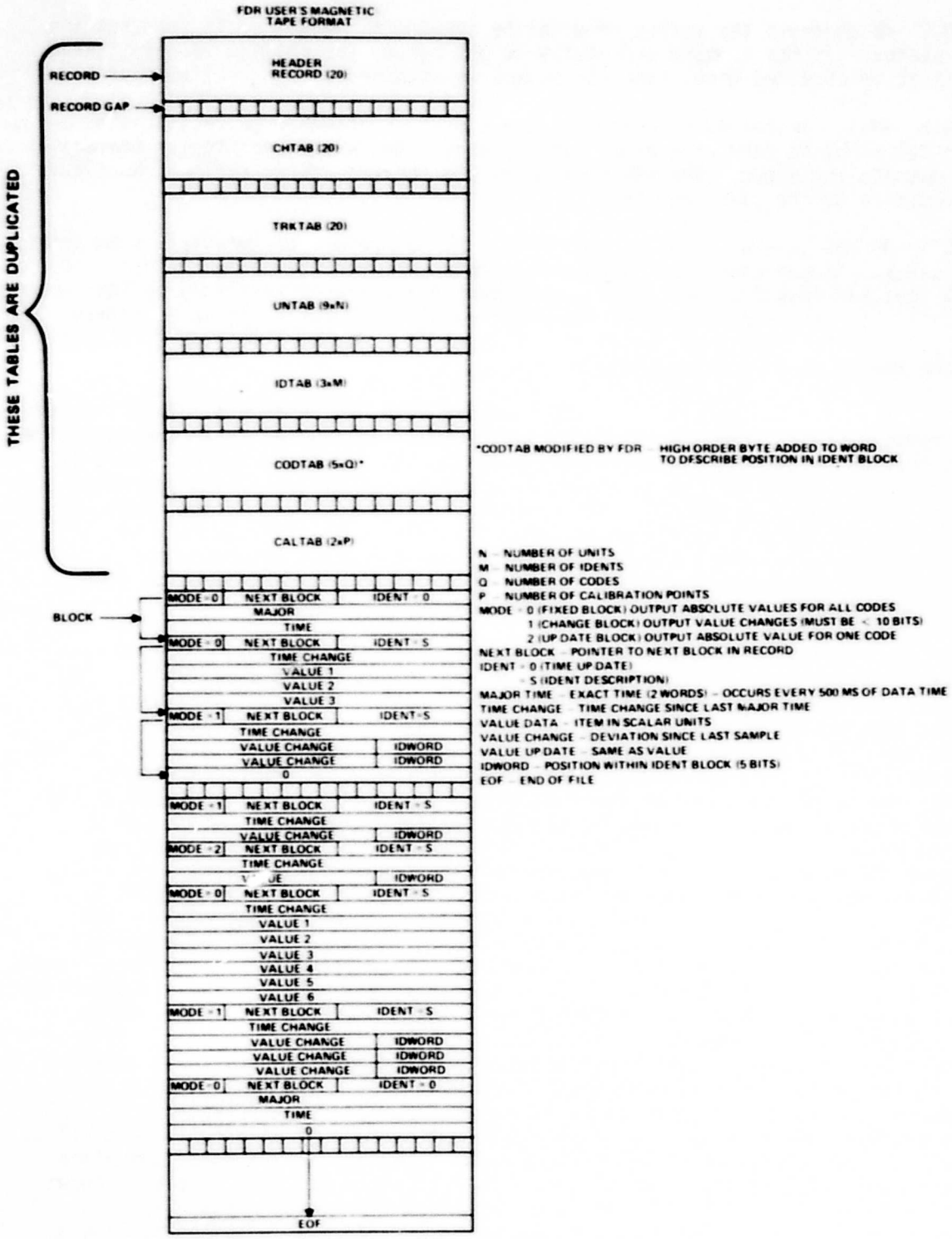


FIG. 4.3 FDR MAGNETIC TAPE FORMAT

generate the tape. As explained previously, the tables at the beginning of the tape give a full description of the tape's meaning. Once the data is output, it is not possible to distinguish what data came from which pass.

4.12 MONTOR is called by CYCLE when no data is ready for processing. MONTOR allows the operator to interact with the process by pushing the BREAK button on the CRT and entering a command (Section 6.2.2) which will be executed when CYCLE runs out of input buffers ready for processing. The routine also displays system parameters so the operator has information necessary for his interaction decisions. Since MONTOR will not stop the data processing tasks, it is not critical operationally.

4.13 The various utility functions used by the routines previously described include COMON, CONVRT, CLOCK, LOGIC, and MTIO.

4.13.1 COMON makes possible data linkages between FORTRAN and assembly language routines.

4.13.2 CONVRT is composed of the four routines listed below:

a. BINTIM converts the digital clock's BCD formatted time into the computer's binary format.

b. DECTIM converts the binary time into hours, minutes, seconds, and milliseconds.

c. HTOMS converts the time in hours, minutes, seconds, into milliseconds.

d. BINBCD converts double precision binary to eight-digit BCD.

4.13.3 CLOCK is a routine which makes the host computer's real-time clock seem like several programmable clocks for use by other FDR routines. This is important for time critical functions, etc.

4.13.4 The LOGIC routine allows FORTRAN to call several functions of a logical nature. The functions are:

a. IAND - logical and

b. IOR - logical or

c. IXOR - logical exclusive or

d. IALS - algebaric left shift

e. IARS - algebaric right shift (sign extension)

f. ILRS - logical right shift

g. ICLR - circular or end around shift

This routine is necessary because the host computer's FORTRAN has no internal logic or shift functions.

4.13.5 MTIO is responsible for initiating all commands to the digital magnetic tape drives and keeping track of the status of tape drive I/O operations. After initiating an operation to the requested magnetic tape unit, MTIO returns control back to the user. This action decreases the time required to input and output data to the tape units. Using the FORTRAN magnetic tape input/output routines would make execution of FDR too slow because after initiating an operation FORTRAN does not return control to the user until the input/output units have completed the operation.

5.0 User's Guide for PREFDR

5.1 Program Setup

5.1.1 FORTRAN Logical Unit Assignments

F1 - T1 or None - Input tape Unit

F2 - T2 - Output tape Unit

F3 - HP or KB - Listing Device

F4 - KB or CR - Command Input Device

F5 - KB or HP - Command Reply

5.1.2 Programs Required (Language)

Program	(Language)
PREFDR	(FORTRAN)
SCAN	(ULTRA)
(FORTRAN LIBRARY)	

5.2 Operator Commands/Inputs (see 5.2.1 for parameter definitions)

EXIT - Returns control to system

OUTPUT - Outputs tables to F2 and checks for bad linkage

LIST - Lists tables on F3

CHANNEL (CNUM, TRACK) - Tells what track on the analog tape recorder a channel is assigned to

TRACK (TNUM, UMT)- Tells what track a multiplexer is assigned to

UNIT (UNUM, RATE, BPW, WPF, LSB, FSP (2 words), BPFSP, ACC, FORMAT) IDENT (IDNM, UNIT, IPROC) - Lists the operational parameters of a unit (multiplexer)

CODE (CNUM, IDENT, SS, WORD, CAL) - Describes the code word's place in the data structure

CALIBRATION (CALNUM, EUMTS, SUITS) - Forms a calibration table by assigning engineering units to scalar units

5.2.1 In the above:

CNUM is the channel number

TRACK is the track to be connected to

TNUM is the track number

UNIT is the unit associated with track

UNUM is the unit number

RATE is the number of frames per second

BPW is the bits per word

WPF is the words per frame

LSB is the least significant bit

FSP is the frame sync pattern (11 digit octal number)

BPFSP is the bits per FSP

ACC is the alternate complement code

FORMAT is the code of the form of PCM

CODE	PCM FORMAT
1	(BIØ-S)
2	(BIØ-M)
3	(BIØ-1)
4	(RZ)
5	(NRZ-S)
6	(NRZ-M)
7	(NRZ-L)

INUM is the IDENT designator

IPROC is the IDENT processor flag designating special processing that must be done on that IDENT

CNUM is the CODE designator

IDENT is the IDENT reference

SS is the type of multiplexing

0 - normal frame

+(A/B) - A frames per subframe; code in frame B

-(C) - C super-frames per frame

WORD is the word in the data frame

CAL is the calibration reference

CALNUM is the calibration index

EUNITS is the engineering units

SUNITS is the scalar units

6.0 User's Guide for FDR

6.1 Program Setup

6.1.1 FORTRAN Logical Unit Assignments

F1 - KB - Command Input Unit

F2 - KB - Command Output Unit

F3 - T1 - Input Tape Unit (At Pass 0)

F4 - T2 - Output Tape Unit (At Pass 0)

F5 - EQ - Equipment Unit

F6 - S1 - Synchronizer 1 Unit

F7 - S2 - Synchronizer 2 Unit

F8 - HP - Strip-Out Unit

6.1.2 Programs Needed

Program	(Language)
FDR	(FORTRAN)

Program	(Language)
CLOCK	(ULTRA)
COMON	(ULTRA)
CONVRT	(ULTRA)
CYCLE	(ULTRA)
IBR	(ULTRA)
LOGIC	(ULTRA)
MONTOR	(ULTRA)
MTIO	(ULTRA)
PCMIO	(ULTRA)
RTDS	(ULTRA)
STRIP	(ULTRA)
FORTRAN LIBRARY	

6.2 Operator Commands/Inputs

6.2.1 Operator Inputs

The time window for each track is displayed on F2. The window may be changed by using the input command

AA, B1, B2, B3, S1, S2, S3

where:

AA is the Track number

B1-B3 are the begin time in hours, minutes, seconds

S1-S3 are the stop time in hours, minutes, seconds

The other operator inputs are:

NUMBER OF SYNCHRONIZER CHANNELS = A

A = 1 or 2 (number of synchronizer channels on line)

PASS NUMBER = B

B = 0-13 (if restarting, at what pass)

RECORD SPEED = C

C = 7.5 thru 120. (speed of analog recorder)

PLAYBACK SPEED = C

STRIP-OUT = D

D = Yes (want strip-out)

D = No (want user's tape)

TIME SCALE = E

E = seconds/inch of strip-out paper

RANGE = F

F = inches/axis (1-10) if F is negative, do not fill between data points (i.e., data points only - no curve)

6.2.2 Operator Commands

ABORT - Abort pass and restart from beginning

TERMINATE - Stop this pass and go to next

RESTART - Restart this pass from the beginning

6.2.3 FDR System Status. System status is displayed in the scratch pad of the CRT.

F4 = AA, F3 = BB, S1 = CC, S2 = DD, DB = EE, SYSTEM = FFFFFF

where:

AA is the number of parity errors on the digital tape drive designated F4

BB is the number of parity errors on the digital tape drive designated F3

CC is the number of sync. errors on synchronizer 1 (S1)

DD is the number of sync. errors on synchronizer 2 (S2)

EE is the number of data bound (DB) errors

FFFFFF is the discrete system status in octal code. The binary bits have the following significance when high:

BIT-0 Not used

BIT-1 Abort pass start all over

BIT-2 Terminate pass and go to next pass

BIT-3 Restart pass from beginning

BIT-4 CHANNEL table error

BIT-5 UNIT table error

BIT-6 IDENT table error

BIT-7 MTIO error

BIT-8 PCMIO error

BIT-9 Stop command for back-log on input tape

BIT-10 No data on synchronizer 1

BIT-11 No data on synchronizer 2

BIT-12 DB error, stop, slow down, and restart

BIT-13 Not used

BIT-14 Not used

BIT-15 Not used

6.3 Assembly Language Subroutines

NOTE: More specific information may be obtained through use of the program listings.

CLOCK

CALL CLOCKI(0) - Initialize clocks

CALL CLOCKF(NUM) - Turn off clock num

CALL CLOCK(NUM, INT, INTR) - Turn on clock NUM, execute INTR (the interupt sequence) in INT millisecond

COMON

CALL COMON(NUM, FELEM) - Set entry NUM address of FELEM

CONVRT

CALL DECTIM(TIM1, TIM2) - Convert time in milliseconds (TIM1) to time in hours (TIM2)

CALL BINTIM(BCD, BIN) - Convert BCD time (BCD) to binary time (BIN)

CALL HTOMS(HRS, MS) - Convert time in hours (HRS) to time in milliseconds (MS)

CALL BINBCD (BIN, BCD) - Convert binary number (BIN) to number (BCD)

CYCLE

CALL CYCLE (NFS, ITEM, ERROR) - Process NFS sync. channels between time limits specified by the user and return any errors (ERROR)

JLM UNIP - Process unit

JLM IDP - Process IDENT

JLM MRGR - Merge data

JLR RO, MIUN - Move data from input buffer

JLR RO, MOUN - Move data to output buffer

IBR

CALL IBI (FS, ERROR) - Set up input buffer for sync. FS

CALL IBU (NFS, FS, PARA) - Test NFS syncs. for first available frame (FS)

LOGIC

IAND (A, B) - A*B

IOR (A, B) - A + B

IXOR (A, B) - A ⊕ B

IALS (A, B) - A*(2**B)

IARS (A, B) - A/(2**B)

ILRS (A, B) - IARS with no sign extension

ICLS (A, B) - IALS with overflow inserted at LSB

MTIO

CALL MTIO (FUN, UN, FWA, SIZE) - Do MT operation FUN on UN

CALL MTIO (FUN = 8, UN, STATUS) - Get MT status

CALL XMTIO (FUN, UN, FWA, SIZE) - Execute and wait for MT FUN completion

PCMIO

CALL PCMIO (UNIT, FUN, LIST)

UNIT = 13 - Analog tape command (FUN)

UNIT = 14 - Switch matrix command (FUN)

UNIT = 15 - Digital clock command (FUN)

UNIT = 16 or 17 - Synchronizer command (FUN)

MONTOR

CALL MONTOR (FUN) - Monitor command (FUN)

CALL FORMAT - Encode

RTDS

CALL RTDS (FS, CTP, STATUS) - Create data structure

CALL FUN (FS, FUNA) - Unit parameters for FS

CALL FNID (FNIDA) - Ident parameters

CALL FNCOD (FCODA) - Code parameters

STRIP

CALL STRIP (FUN, PAREM) - Set up and strip-out data

SCAN

CALL SCAN (TYPE, MESS, CHARS, VALVE) - Extract next parameter

CALL IO (FUN, UN, FWA, SIZE) - Execute and wait for magnetic tape completion

6.4 FORTRAN Routines

PREFDR - Generate tables describing hardware

FDR - Initializes parameter for real-time data reduction

7.0 Introduction for User's Data Tape Software

7.1 FETCH, the user's data tape software package was written to free the user from the tedious process of decoding and managing the reduced data from the FDR output tape (IDATA). Inadequate data due to a party error (PE). is automatically intercepted and thrown out, and the user's working data is updated using the next set of valid data. With the calls listed below, all the pertinent data from the instrumentation and data handling tape put out by FDR (IDATA) are easily

accessible to the user. The general format of this data tape is shown in Figure 4.3.

7.2 The FETCH Software consists of three major subroutines:

a. FETCHI - The initialization routine reads the first 15 records in the instrumentation and data handling data tape (IDATA). From this information, the tables required by the FETCH programs are setup, and the beginning time on the IDATA Tape is calculated. Also a test identification number is returned to the user. Finally, the user's HEL codes are ordered and are put in the working code value table.

b. FETCHT - The time update routine keeps the working code value table current for the time change specified by the user.

c. FETCHV - The code value routine uses the calibration table to calculate the code's value in engineering units from the scalar units of the IDATA.

7.3 There are two optional print routines. These subroutines need be part of the FETCH package only if the user wants a copy of that particular print routine's subject matter. The routines are:

a. PRINT1 - This routine will print any or all of the beginning tables (Figure 1.1).

b. PRINT2 - This routine will print a list of the user's codes along with the hardware description of the codes.

7.4 There are seven supportive programs:

a. MTIOF - Magnetic Tape Input/Output Routine

b. MTIPE - Controls MTIOF, checks for parity errors (PE) and adjusts the data stream when a PE is sensed.

c. DBLWRD - Adds, subtracts, compliments and stores all double words (32 bits).

d. DFLOAT - Changes double word integer to floating point words.

e. IFIELD - Removes from a word and right justifies any bit field specified.

f. LOGIC - allows FORTRAN to call logical and shift functions.

7.5 The test identification number returned to the user via FETCHI should be recorded on all the user's data. This number is a master reference number and may be used, at some later data, to trace data to a specific test.

7.6 The remaining chapters will describe, in detail, the implementation of the FETCH software.

8.0 FETCH Program Description

8.1 The User's Main Program

8.1.1 The FETCH subroutine package requires approximately 12,000 words of memory. This total includes a one column 3750 word array in a common storage block named ISTORE. This block of memory is used to store the tables and various data management pointers required in the FETCH package. The two optional print routines require 1570 and 550 words, respectively.

8.2 FETCHI - The Initialization Subroutine

8.2.1 The initialization subroutine, FETCHI, reads in the code list provided by the user, forms the various tables required by FETCH, returns a test identification number and supplies an absolute starting time.

8.2.2 After rewinding the input IDATA type, the first seven records are read. These records describe the physical setup of the hardware configuration for the instrumentation system. If no PE occurred in the first seven records, the next seven records are discarded, or if a PE occurred the next set of duplicate tables are read into memory. Once the tables are in memory, columns one thru five of the Working Code Value Table (Table 8.1) are set up using the user's code list and the first seven tables.

8.2.3 Next, the first record of good, i.e., contains no PE, block data (Figure 4.3) is read into the buffer. The time of the first code value on the IDATA tape is calculated; returned to the users in hours, minutes, seconds and milliseconds and stored for use by the other FETCH routines.

8.2.4 Finally, the test identification number, actually the first word in the Test Information Table (the HEADER record of Figure 3.1) is returned to the user.

8.3 FETCHT - The Time and Value Update Routine

8.3.1 The time and value update subroutine, FETCHT, requires for a time change in multiples of 10 microseconds as an input. Since the user's time increment may not exactly correspond to the time changes on the IDATA tape, a corrected time value, which corresponds to the actual tape time increment, is returned to the user.

8.3.2 The FETCHT subroutine compares the time increment supplied by the user with the time on the IDATA tape. A correction to the time movement is made for each major time block (Figure 4.3) encountered. As the time is compared, the routine continually updates the value of all the codes in the Working Code Value Table (Table 8.1). When the time on the IDATA tape has been incremented equal to or greater than the user's time increment the processing stops and the corrected time is returned to the user.

8.3.3 Another form of time change may be used. If the user calls FETCHT with the argument equal to minus one, the time change will be automatically set equal to the time change required to read the next block of data (Figure 4.3). Then all blocks with a time equal to this will be processed, and the value of the time step will be returned to the user.

8.3.4 The buffer pointer is increased by this subroutine. When the pointer reaches the end of the buffer or the end of the data in the buffer, the buffer

Table 8.1

WORKING CODE VALUE TABLE

CODE	IDENT	WORD** IN IDENT/FRAME		BITS PER WORD	CALIBRATION TABLE INDEX	SCALAR VALUE* (2 WORDS)
1	1	1	1	10	12	000000 000167
2	1	2	2	10	6	177777 177611
3	1	3	3	10	3	000000 000011
4	3	3	4	10	3	000000 001011
8	3	1	5	18	12	000001 001201
9	3	2	6	12	6	000000 000056
10	3	4	7	17	0	000000 100000
51	4	1	8	8	16	177777 177773
.	.			.	.	. .
.	.			.	.	. .
.	.			.	.	. .
364	2	1	9	14	101	000000 000000

*Octal Value
**Upper and Lower ByTe
i.e. 11 corresponds to 257

pointer is reindexed and the control chain (MTIPE and MTIOF) for an IDATA tape to be read into the FETCH buffer is activated. A minus one returned to the user signifies that the end of the file has been encountered.

8.4 FETCHV - The Code Value Routine

8.4.1 For the code specified the code value subroutine, FETCHV, will, using the calibration table, change the scalar units (columns 6 and 7 of Figure 2.1) to engineering units. The value will be returned to the user as a floating point value.

8.4.1.1 The scalar units are changed to engineering units by linear interpolation using a n-point calibration table (Figure 3.3). Once the code is found in the Working Code Value Table (Table 8.1), the calibration table reference column will indicate the starting point in the calibration table. From this point the scalar units of the calibration are compared to the scalar units of the code value in the Working Code Value Table. When the calibration table value is less than the code value, a linear interpolation is performed using the scalar and engineering units which bracket the code value. The interpolation is performed in floating point arithmetic and the value is returned as a floating point number. If the bits per word of the Working Code Value Table indicates a double word value, the same operation is performed, only double word values are used.

8.4.1.2 A zero value in the calibration table reference column indicates the scalar value in the Working Code Value Table is the correct value, i.e., there are no engineering units associated with this code value. The scalar value is changed to a floating point number and returned as the code value.

8.4.2 If a minus one code is used the number of parity errors during the run will be returned to the user.

8.5 MTIPE and MTIOF - The Input/Output Control Routines

8.5.1 The magnetic tape input parity error control is the MTIPE subroutine. Every read command must go through MTIPE.

8.5.1.1 To indicate that the beginning tables are being read into memory, the length of the buffer argument is negative. If a PE occurs a PE flag is returned to the calling routine, FETCHI. The PE flag results in reading the second set of tables into memory. If another PE occurs the program stops.

8.5.1.2 If a PE is detected in any block of data, MTIPE will discard all the data in the buffer by requesting another read from the magnetic tape to the buffer. A search for a major time block (Figure 4.3) is begun. When found, the buffer pointer is indexed and the program sequence continues from the major time block.

8.5.1.3 When an end of file (EOF) is sensed an EOF flag is returned to the user's main program as the FETCH subroutines return control to the main program.

8.5.2 The magnetic tape input/output subroutine, MTIOF, controls the magnetic tape units. MTIOF is capable of executing all the tape drive functions available

to the AN/UYK-15. The only functions used in the FETCH package are read and rewind. The read call is similar to the FORTRAN READ statement, except no error messages are outputted and the processing does not stop. Instead error flags are sent to the calling programs. MTIOF is composed of two major subdivisions, which are the execution of the calling command and evaluation of the status of the call.

8.5.2.1 The execution of the call is accomplished by storing the calling parameters into the IO$$ package (reference (a)). The packet as used in the HEL software (Figure 8.1a) has one difference from reference (a) in that the skip count field is also used as a parity error control field. A typical parameter storage for the FETCH package is shown in Figure 8.1b. Once the package is filled IO$$ is called and the packet is executed.

8.5.2.2 The status of the logical unit is checked before the packet is executed. When this operation has been executed, the status of the operation is again checked. If the status code indicates a PE, a minus two status flag is returned via status and an EOF returns a minus one. A read operation will return in status the number of words read; otherwise, a successful operation other than a read will return a status equal to zero.

8.6 DFLOAT, DBLWRD, IFIELD and LOGIC - General Utility Routines

8.6.1 DFLOAT (Double Float) takes a two word integer and floats it. The output is a single precision floating point number (1 sign bit, 7 bit exponent and 24 bit fractional mantissa).

8.6.1.1 If the input is negative, it is complimented and a negative flag set. Once a positive number is assured the high order word (bits 16 to 31) are multiplied by 65,536. If there is a bit in the 2^{15} position of the lower order word, it is eliminated and the DFLOAT value corrected accordingly. Finally, the value of the lower order work is added to DFLOAT and, if there is a negative flag the DFLOAT value is complimented.

8.6.2 The double word (32 bits) routine, DBLWRD, provides the various integer double word functions listed below:

a. An add function, IDADD

b. A subtract function, IDSUB

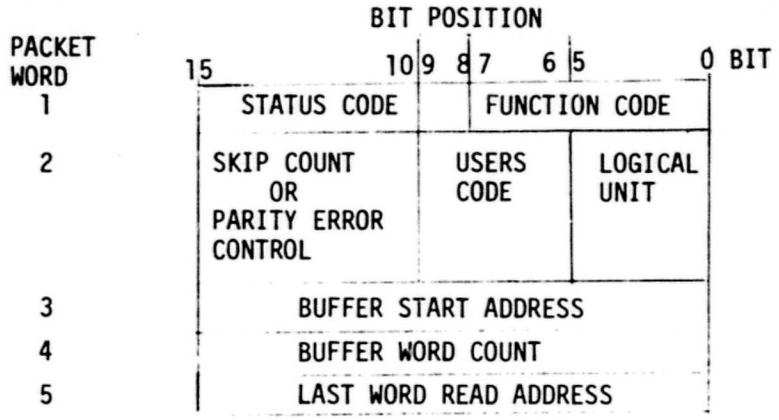
c. twos compliment function, IDBAR

d. A double word store function, IDSTOR

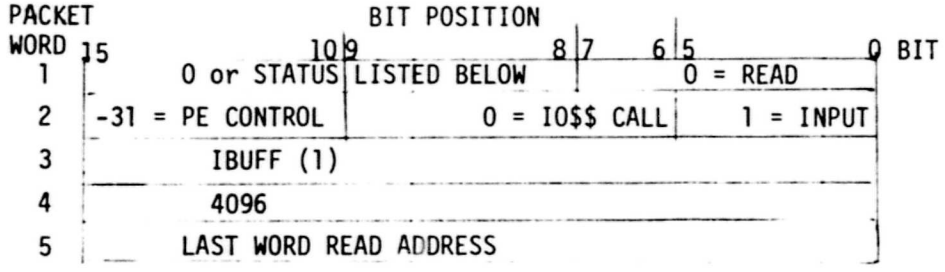
e. A time calculation function, ITMDIV

8.6.2.1 The only function not self-explanitory is ITMDIV. This is a special function used by FETCHI to convert the double word major time, which is in milliseconds, to hours, minutes, seconds and milliseconds.

(a) User's Handbook of UNIVAC 1616 Level 1 Support Software Manual



(a)



STATUS CODE

0-OPERATION COMPLETE
1-PARAMETER PACKET ERROR
2-END OF FILE
3-BUSY
Ø377-INPUT/OUTPUT IN PROCESS

(b)

Figure 8.1 IO$$ PARAMETER PACKET

8.6.2.2 Except for determining the storage address, the coding of the double word package is straight forward, i.e., the function requested is determined, the calling parameters are retrieved, the storage address and the return address are calculated, the operation is performed, and, finally, the return sequence is completed. At this time, a short background discussion is needed to show how a double word answer is stored. The AN/UYK-15 FORTRAN complier uses a store register 2 (R2) instruction to store the final answer returned by any function. The following is one coding sequence used to call a function.

JLM FUNCTION	Jump link memory to function called
N	Number of parameters in the call
A1	Address
A2	of
.	the
.	Parameters
AN	
S RM, AJ	Store register M, an argument of this program, in the next address (AJ).
L, RK, AK	Load register K (used to modify the R2 storage address) with the value in address, AK, which is the result of the I + 1 in the FORTRAN coding for the NUMB (I + 1) storage address. The address of NUMB is in AJ.
S R2, AJ, RK	Store register 2 in address determined by AJ modified by the value in register K.

In the above example all addresses AK and AT, are in the first address below the store or load instruction. In the double word storage instructions a problem arises because the sequence above will store only one register, and two register storage is needed to store two words. DBLWRD determines the R2 storage address (base address plus modifier) by searching the coding of the calling routine, shown above. To do this DBLWRD separates the instructions from the addresses and searches the instructions until a store register 2 instruction is found. Then the modifier is determined and added to the base address, this address is incremented by one and register 3, the lower order word (bits 15-0) of the double word, is stored in this address. Register 2, the highest order word (bits 31-16) of the double word, is stored by the compiler coding.

8.6.3 The bit field extractor, IFIELD, removes the bit field specified from the word specified and right justifies it. This is accomplished by forming a right justified mask of ones the length of the bit field. Then all the bits in the word are shifted right until the bit field is right justified. Finally, the mask and the modified word are ANDed together to form the answer.

8.6.4 The logic and shift function, LOGIC, is explained in paragraphs 4.13 and 6.3.

8.7 PRINT1 and PRINT2 - THE OPTIONAL PRINT ROUTINES

8.7.1 The beginning table print routine, PRINT1, is used to provide a hard copy of the tables for the users. Any or all the tables may be printed. If no hard copy is required, this subroutine is not coded into the computer. There are two print controls. The first controls the table(s) to be printed. The second controls the listing devices, the printer and the input terminal's CRT (cathode ray tube). If the printer is selected each individual table is started at the top of a page, and the printing continues until done. If the CRT is selected the display is filled then the program pauses until the operator decides to continue the table listing.

8.7.2 The Working Code Value Table may be printed using the PRINT2 subroutine. Depending on the print code selected, the code value may be printed as a two word integer, as a two word octal value, or as a floating point number. The listing device may be either the printer or the CRT. The only difference is that when the CRT is selected the program pauses at the end of the table. The table is always started at the top of the page. Like PRINT1, this subroutine is optional.

9.0 User's Guide to FETCH

9.1 Program Setup

9.1.1 Main Program Special Requirements - Dimension and assign to a blank common area a 3750 word array called ISTORE, i.e., COMMON ISTORE (3750 words).

9.1.2 FORTRAN Logical Unit Assignments

F1 - HP (high speed printer) or KB (keyboard) - Error listing device

F2 - KB - Table listing device

F3 - HP - Table listing device

9.1.3 Programs Required (Language)

Program	(Language)
FETCHI	(FORTRAN)
FETCHT	(FORTRAN)
FETCHV	(FORTRAN)
IFIELD	(FORTRAN)
MTIPE	(FORTRAN)

Program	(Language)
DEFLOAT	(FORTRAN)
PRINT1	(FORTRAN) - Optional
PRINT2	(FORTRAN) - Optional
DBLWRD	(FORTRAN)
MTIOF	(FORTRAN)
LOGIC	(FORTRAN)

(FORTRAN LIBRARY)

9.2 Major Subroutine Calls

9.2.1 FETCHI (IDTEST, NUMB, LIST, INLTIM)

where:

IDTEST is the test identification number

NUMB is the number of items in the list

LIST is the list of code words

INLTIM is the initial time array (4) used to hold the hours (1), minutes (2), seconds (3), and milliseconds (4).

9.2.1.1 This subroutine must be the first FETCH routine called. The test identification number is returned and should be recorded with all data used.

9.2.1.2 Error Messages

9.2.1.2.1 If a code, which is in the LIST, is not found on the input IDATA tape, the error message is:

CODE XXXXXX NOT FOUND

where:

XXXXXX is the code identification number

The program pauses. If the user decides to continue the code, it will be ignored and NUMB will be decreased by one.

9.2.1.2.2 If both sets of beginning tables have PE, the error message is:

STOP TABPE

Control is returned to the system.

9.2.2 FETCHT (DELTIM)

where:

DELTIM is the users increment of time in milliseconds.

The maximum value of DELTIM is 32767 milliseconds.

9.2.2.1 This subroutine will return a corrected DELTIM which corresponds to the actual time incremented on the IDATA tape.

9.2.2.2 If DELTIM equals -1.0, the time increment will be equal to the time between consecutive blocks on the data tape. If there are several blocks with identical time, all will be processed. The exception to the above is when a major time block is the next block to be processed. It will be processed and then the next block's time will be used to calculate the time increment. In all cases a corrected time increment will be returned to the user.

9.2.2.3 If a minus one (-1.0) is returned to the main program, the EOF has been sensed. The user should check for the EOF in his program.

9.2.3 FETCHV (ICODE, VALUE)

where:

ICODE is the code where value is required

VALUE is the value in single precision float point format

9.2.3.1 If ICODE was not in the legal LIST of FETCHI, a

STOP NOCODE

error message will result, and control will return to the system.

9.2.3.2 If ICODE equals -1, the number of PE will be returned via VALUE to the user.

9.3 Special Supporting Subroutine Calls

9.3.1 MTIPE (IUNIT, NAME, IAMT)

where:

IUNIT is the magnetic tape input unit

NAME is the buffer's name

IAMT is the length of the buffer

If AMT is negative the beginning tables are being read; if positive, the block data (Figure 4.3).

9.4 General Use Routine Calls

9.4.1 MTIOF (FUNCODE, IUNIT, NAME, IAMT, STATUS)

where:

FUNCODE is the function code, i.e.,

0-Read

1-Write

2-Initiatize

3-Pass Record Forward

4-Pass Record Backwards

5-Pass File Forward

6-Finalize

7-Rewind

IUNIT is the magnetic tape unit

NAME is the address of the storage location

IAMT is the length of the buffer to be read or written

STATUS is the status of the magnetic tape system, i.e.,

.EQ. 0 - Operation Complete

.GT. 0 - Number of words read

.EQ. -1 - End of File

.EQ. -2 - Parity Error

Other forms of the call are:

MTIOF (FUNCODE, IUNIT, NAME, IAMT)

MTIOF (FUNCODE, IUNIT, STATUS)

MTIOF (FUNCODE, IUNIT)

The last two calls cannot be used for a read or a write.

9.4.2 DFLOAT (NUMBER)

where:

NUMBER is the first word address of a double word integer.

This function changes the double length (32 bit) number to a single precision (sign bit + 7 bit exponent + 24 bit mantissa) floating point number.

9.4.3 DBLWRD is a double word function package. The calling form is: FUNCTION (N(I), M(J)) or FUNCTION (L(K))

where:

FUNCTION is the double word function, i.e.,

IDADD - add (N(I), N(I+1)) - (M(J), M(J+1))

IDSUB - subtract (N(I), N(I+1)) - (M(J+1))

IDBAR - twos compliment $\overline{(L(K), L(K+1))}$

IDSTOR - STORE (L(K), L(K+1))

ITMDIV - Special time function (N(I), N(I+1))/(M(J), M(J+1)) Modulus two storage total + remainder to (N(I), N(I+1)).

These functions must have a two word storage location specified, i.e., X(1) = IDADD (N(I), M(J)) will have the answer stored in X(1) and X(2); however, X(1) = IDSUB(IDADD(N(I), M(J)), (N(I)) will not work because IDADD (N(I), M(J)) has no specified storage location. The AN/UYK-15 FORTRAN compiler will not give any error messages for the double word functions.

9.4.4 IFIELD (NAME, HBIT, LBIT)

where:

NAME is the address

HBIT is the high order bit of the bit field

LBIT is the low order bit of the bit field

This function removes from the word NAME the bit field beginning with and including the highest order bit to and including the lowest order bit. The bit field is also right justified.

9.4.5 LOGIC is a logic and shift function package discussed in paragraph 6.3.

9.5 Optional Print Subroutine Calls

9.5.1 PRINT1 (ITPCODE, ISORP)

where:

ITPCOD is the table print code, i.e.,

1 - Test Information Table

2 - Channel Table

3 - Track Table

4 - Unit Table

5 - Code Group Identification Table

6 - Data Item Reference Table

7 - Calibration Table

8 - All the tables above

ISORP is the CRT(=0) or the printer(=1). If the CRT is selected, the subroutine will PAUSE at the end of each page.

9.5.2 PRINT2 (IVALCT, ISORP)

where:

IVALCT is the code value format control, i.e.,

0 - Integer

1 - Floating point

2 - Octal

ISORP is the same as in 3.5.1.

This subroutine will print the Working Code Value Table which will include all the codes listed in the FETCH1 call. If the keyboard is selected the subroutine will PAUSE at the end of the table.

10.0 Program Flow Charts

10.1 Figures 10.1 - 10.24 are the flow charts for each program previously discussed.

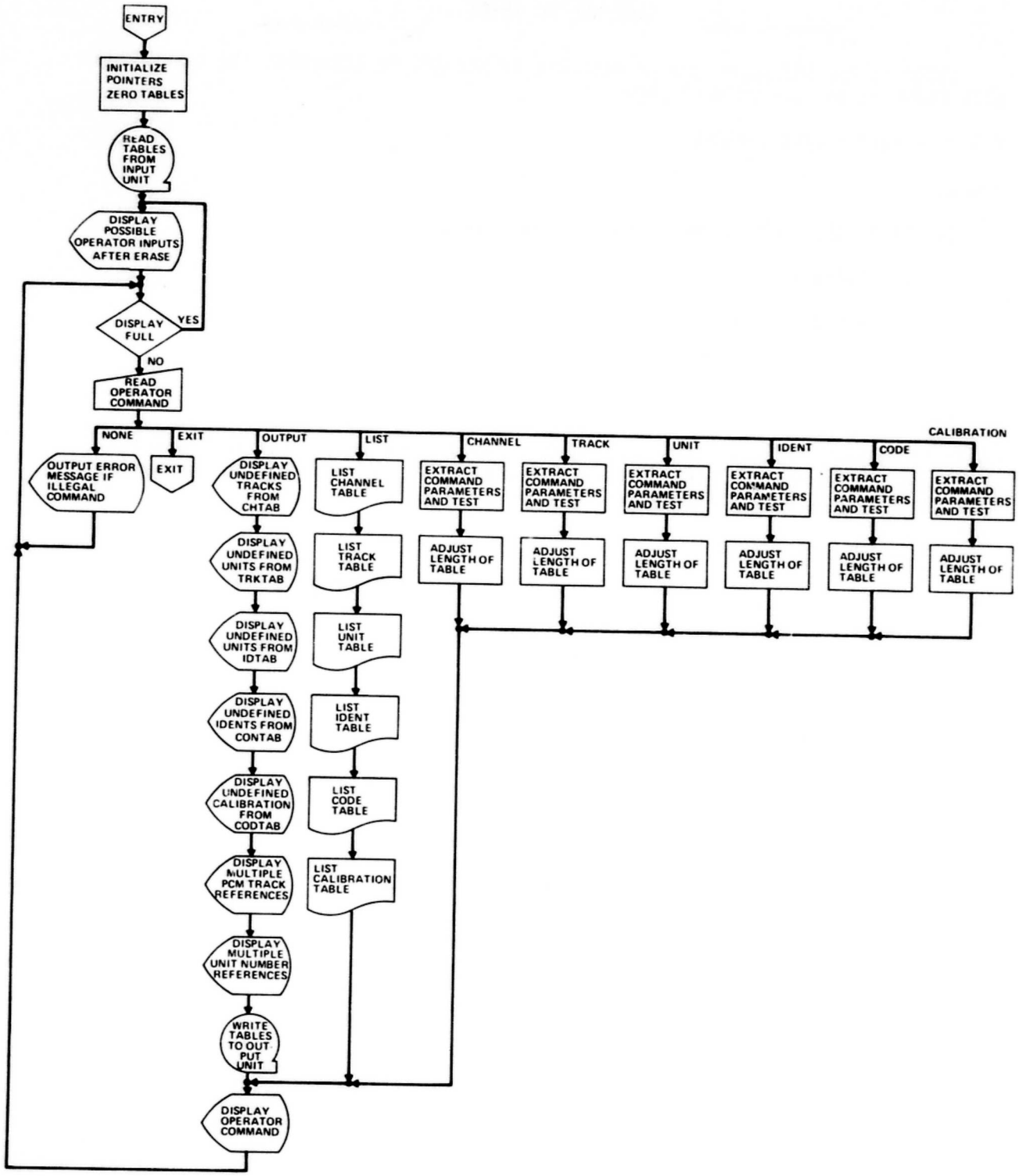


FIG. 10.1 PREFDR FLOW CHART

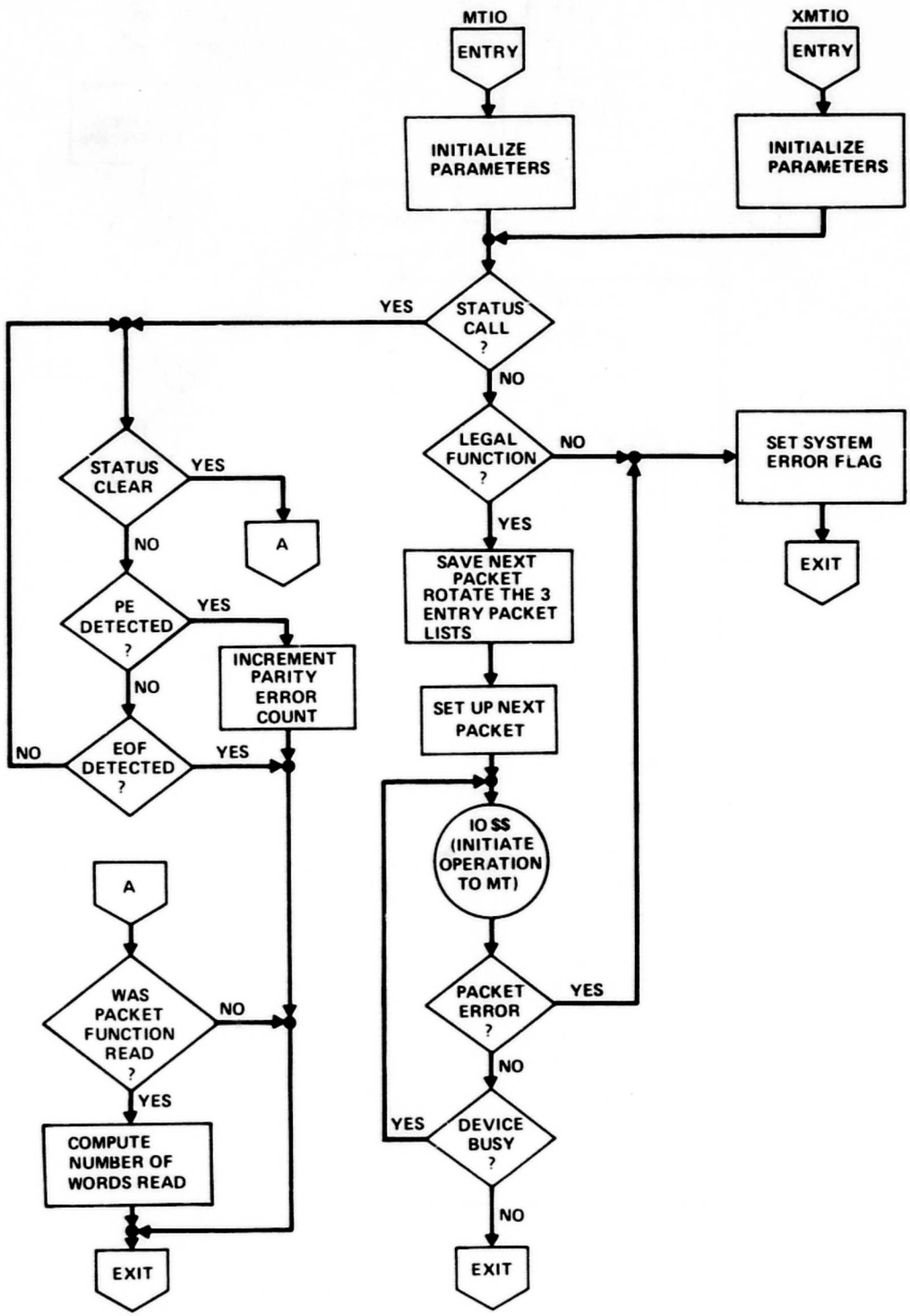


FIG. 10.2 MTIO FLOW CHART

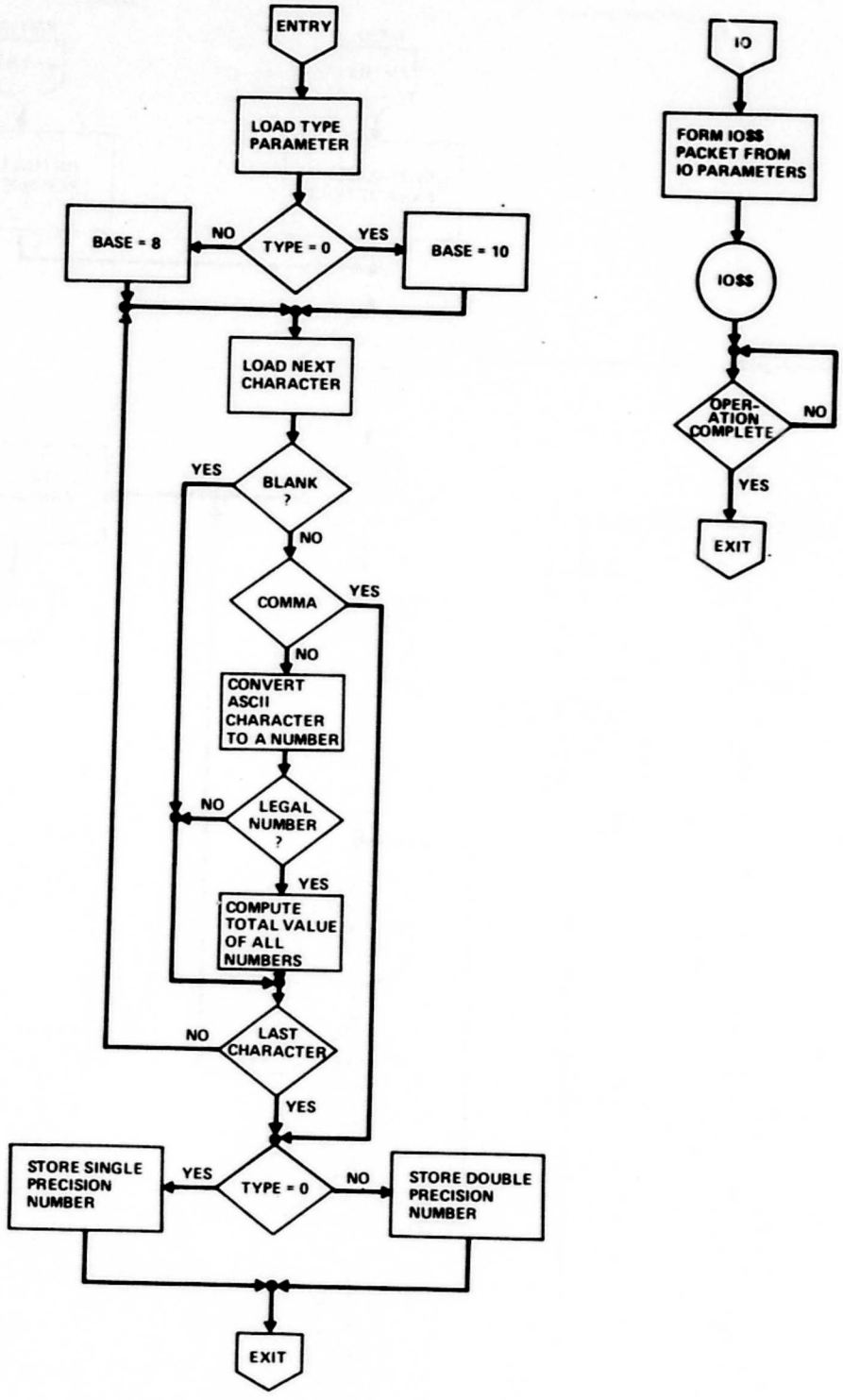


FIG. 10.3 SCAN FLOW CHART

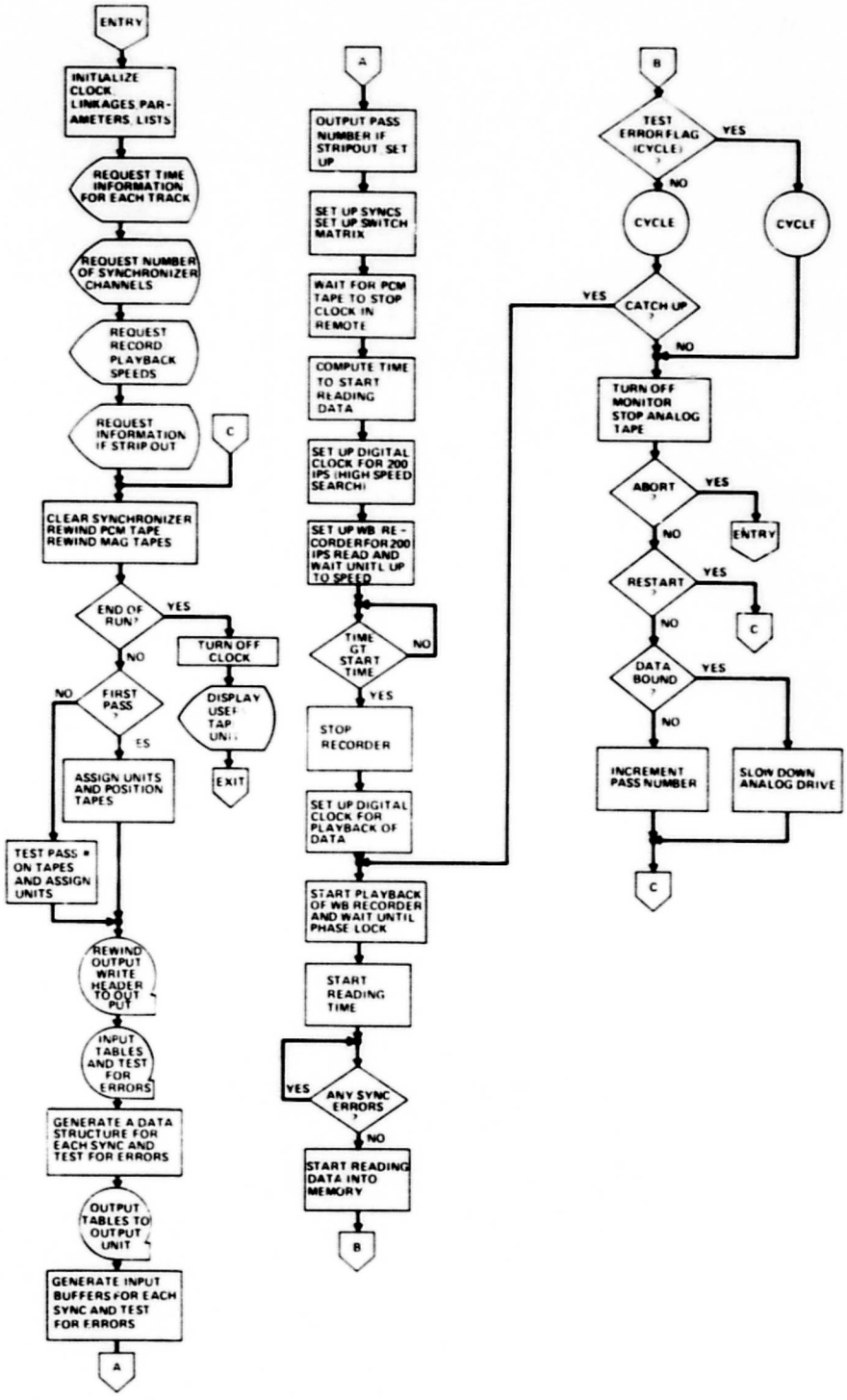


FIG. 10.4 FDR FLOW CHART

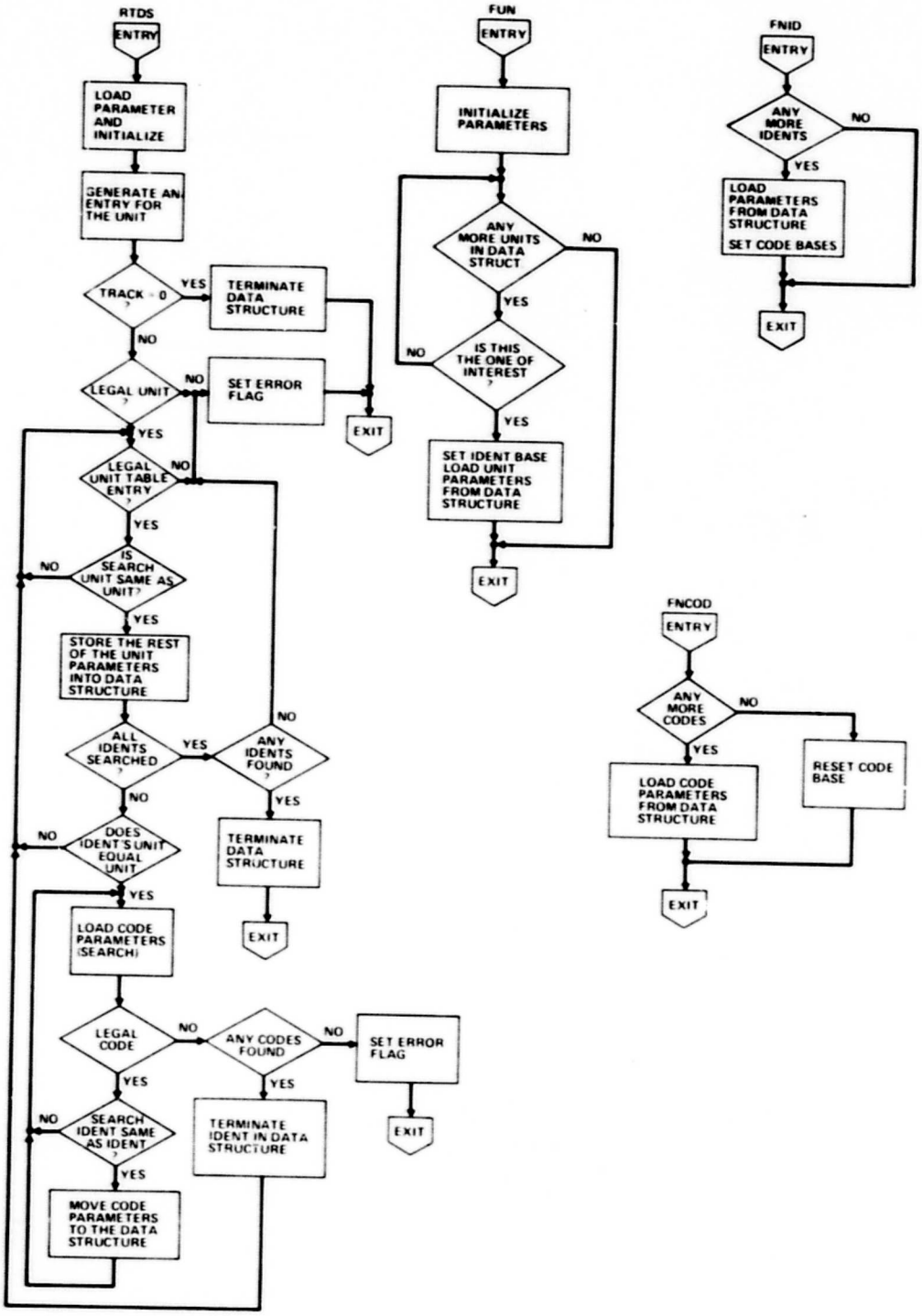


FIG. 10.5 RTDS FLOW CHART

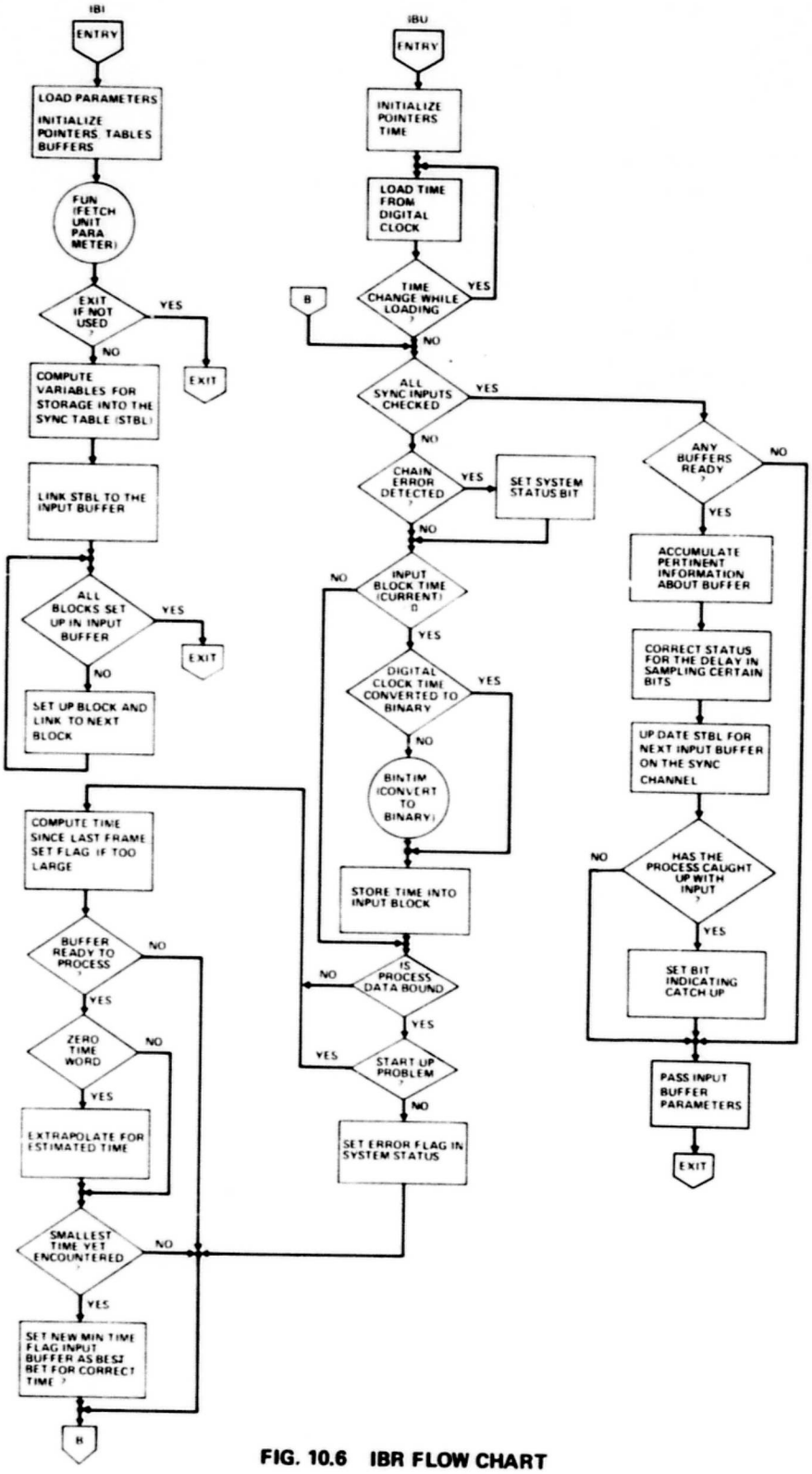


FIG. 10.6 IBR FLOW CHART

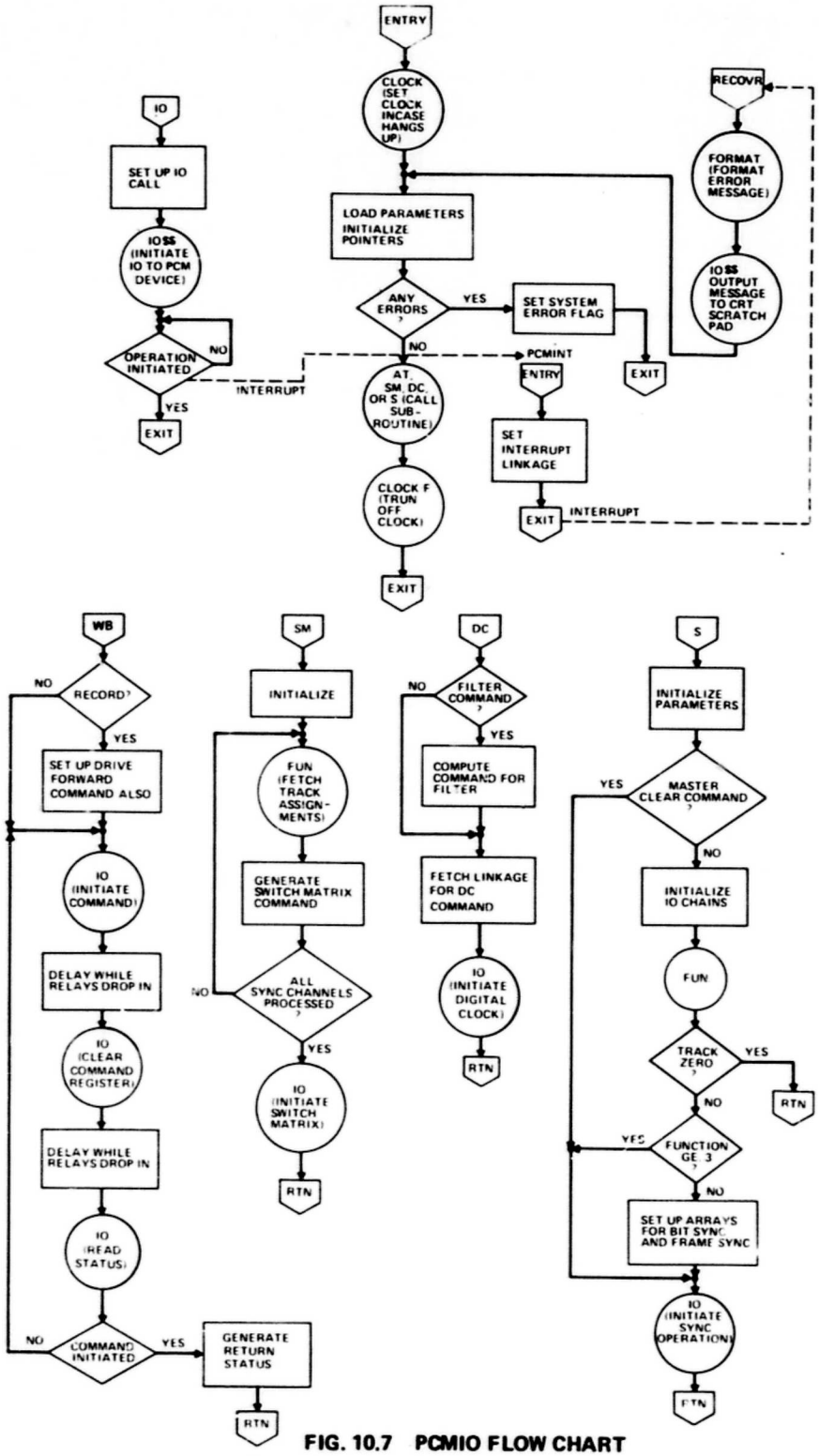


FIG. 10.7 PCMIO FLOW CHART

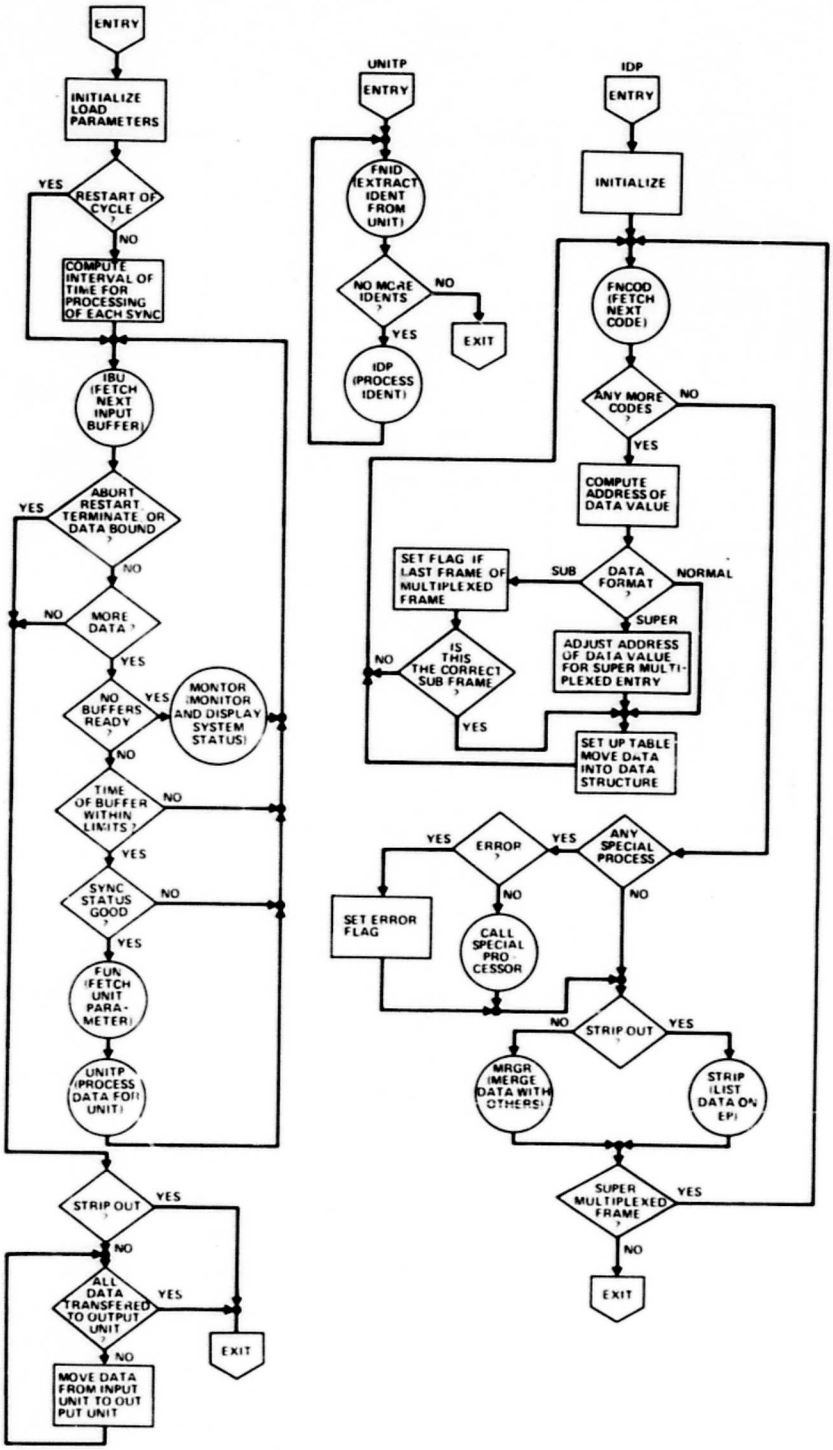


FIG. 10.8 (a) CYCLE FLOW CHART

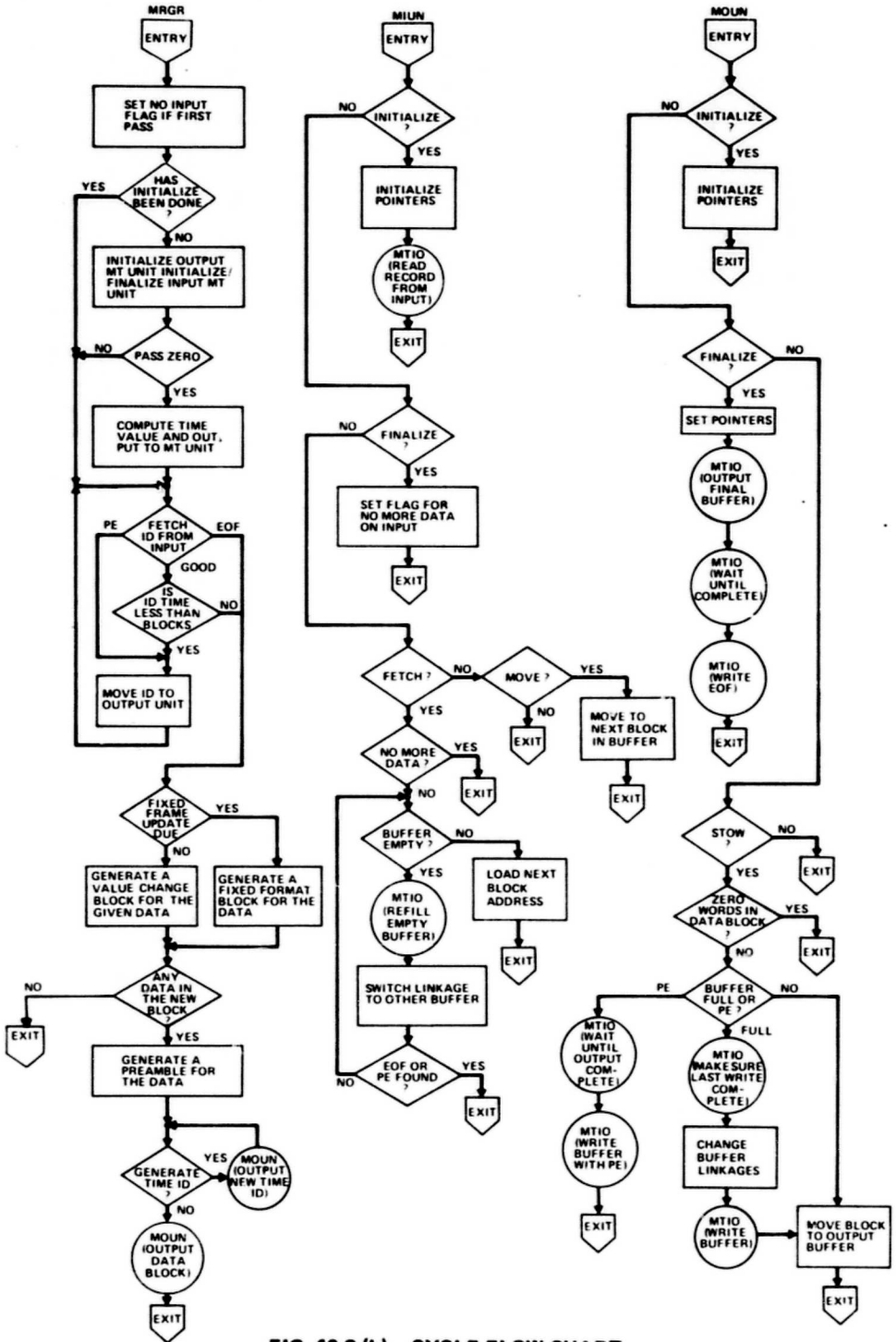


FIG. 10.8 (b) CYCLE FLOW CHART

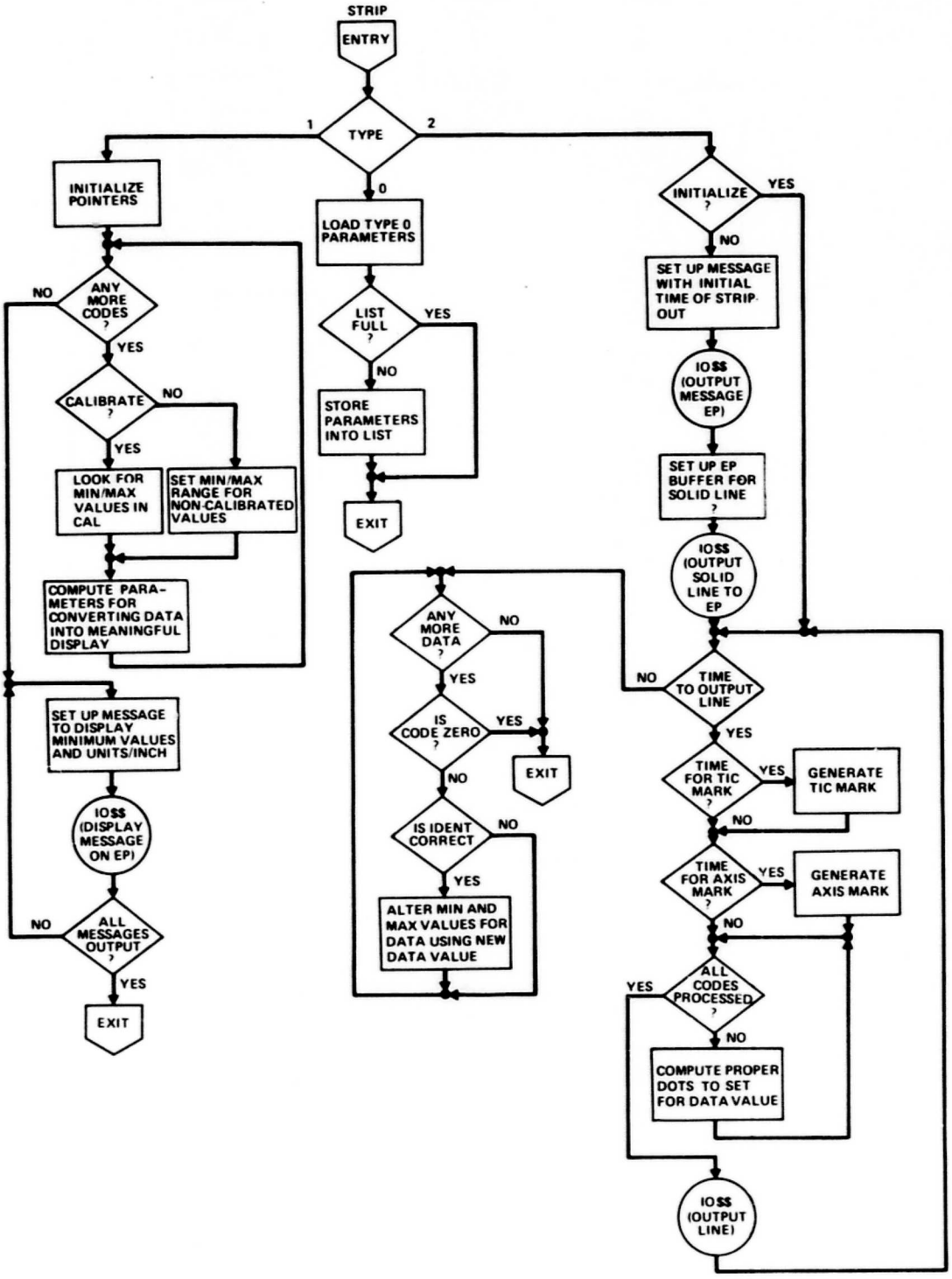


FIG. 10.9 STRIP FLOW CHART

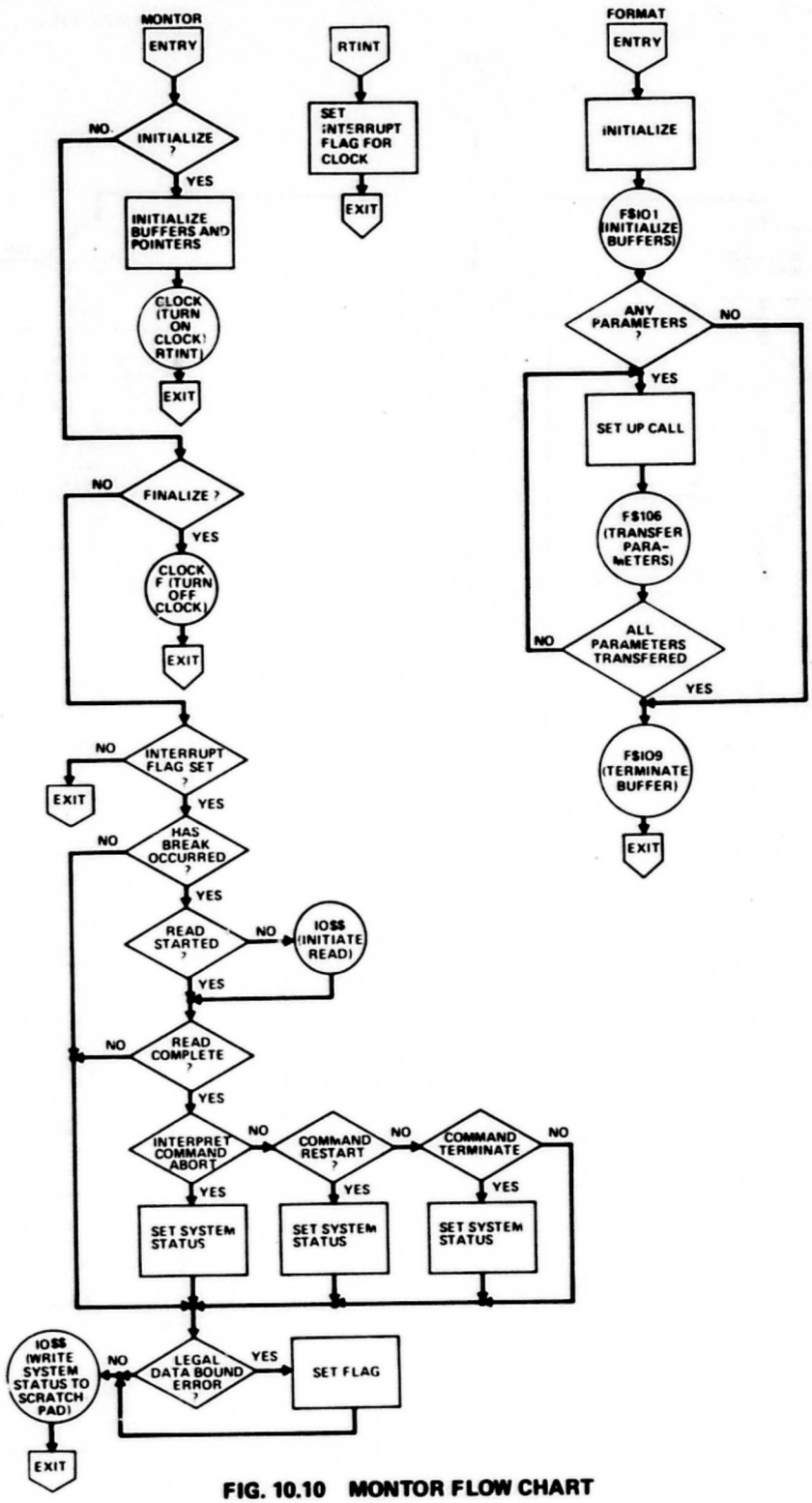


FIG. 10.10 MONTOR FLOW CHART

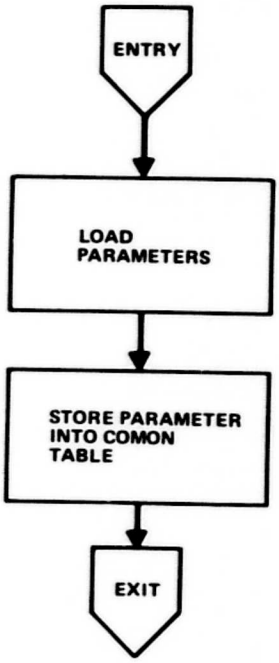


FIG. 10.11 COMON FLOW CHART

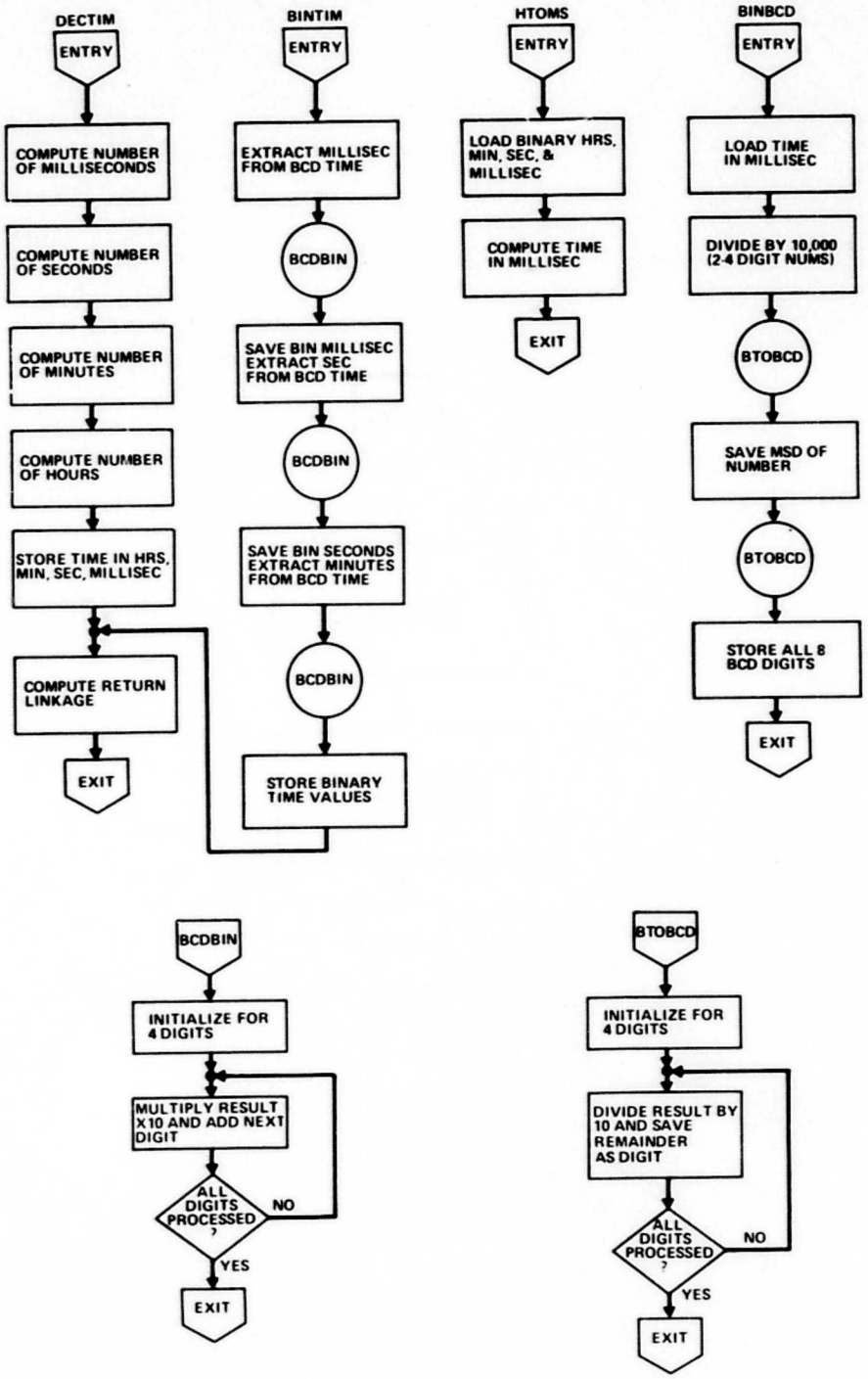
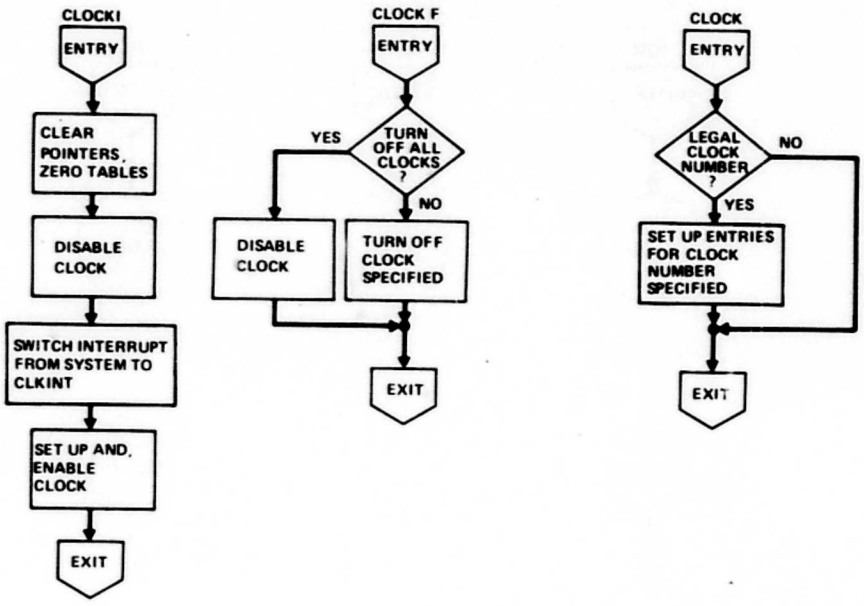


FIG. 10.12 CONVRT FLOW CHART



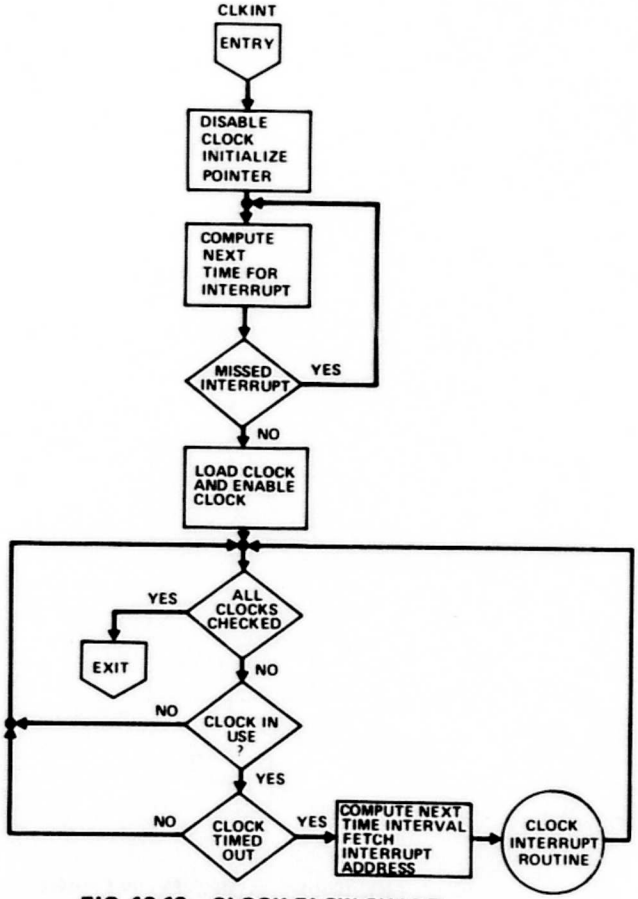


FIG. 12.13 CLOCK FLOW CHART

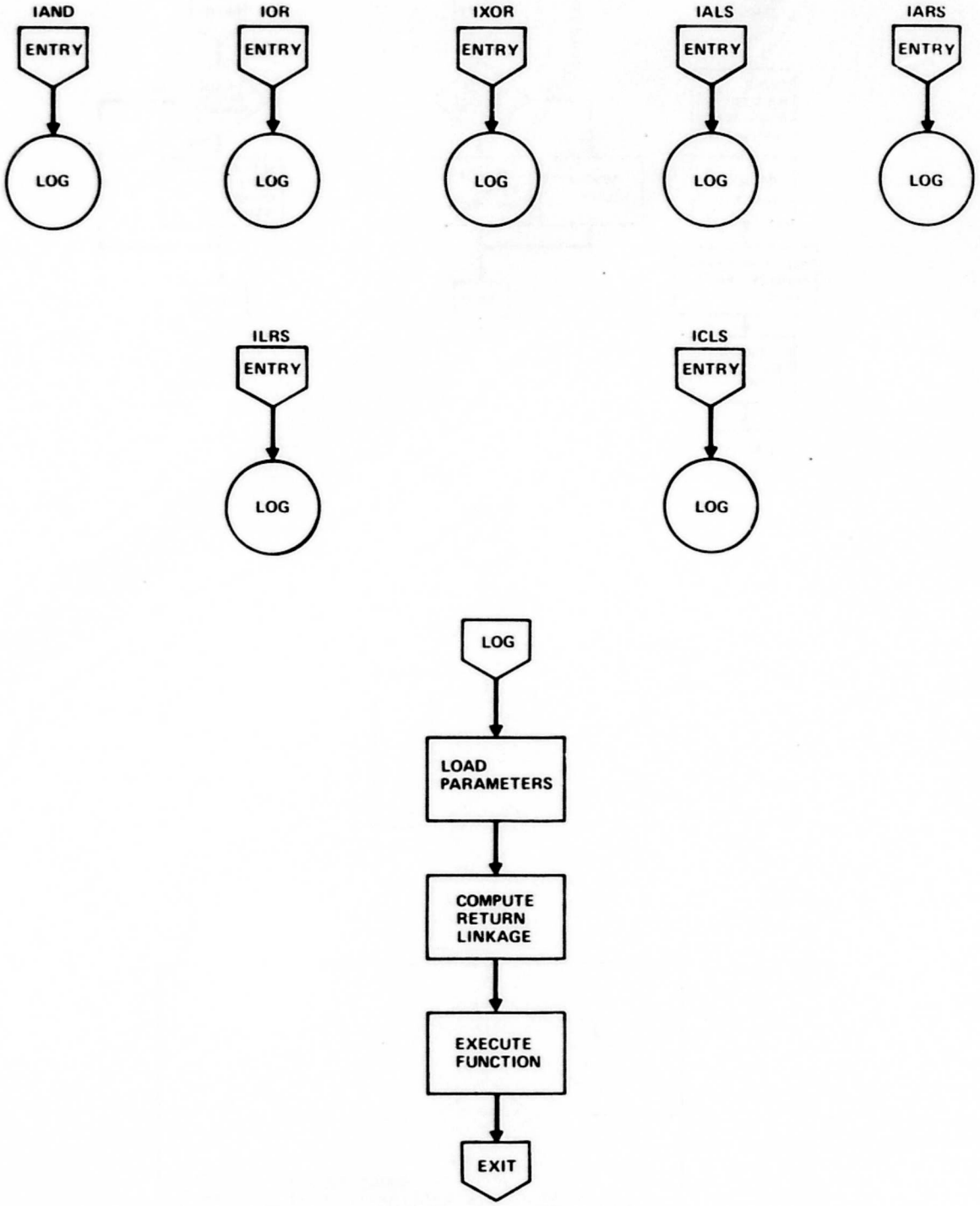
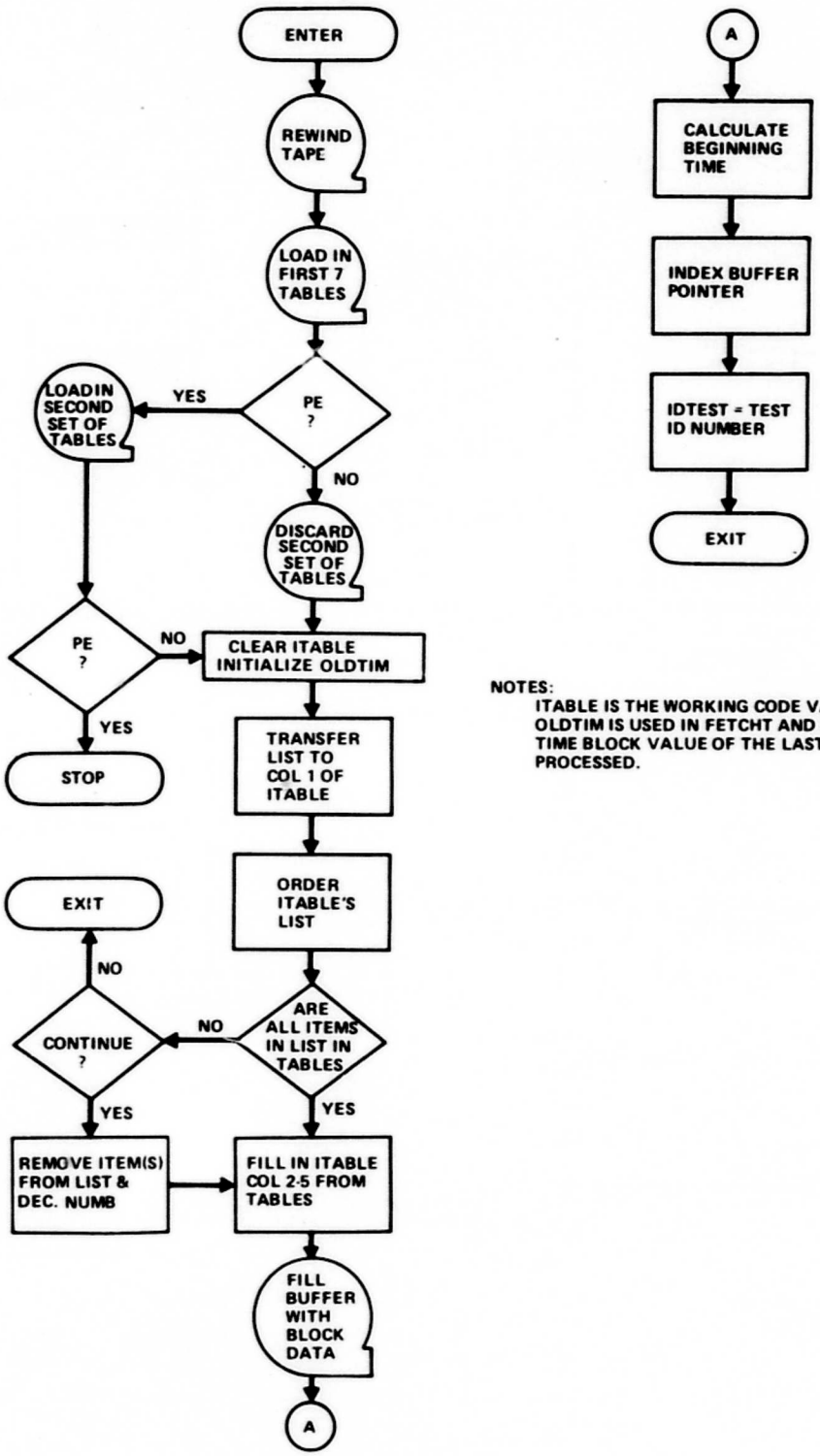


FIG. 10.14 LOGIC FLOW CHART



NOTES:
ITABLE IS THE WORKING CODE VALUE TABLE
OLDTIM IS USED IN FETCHT AND IS THE IDATA TIME BLOCK VALUE OF THE LAST BLOCK PROCESSED.

FIG. 10.15 FETCHI FLOW CHART

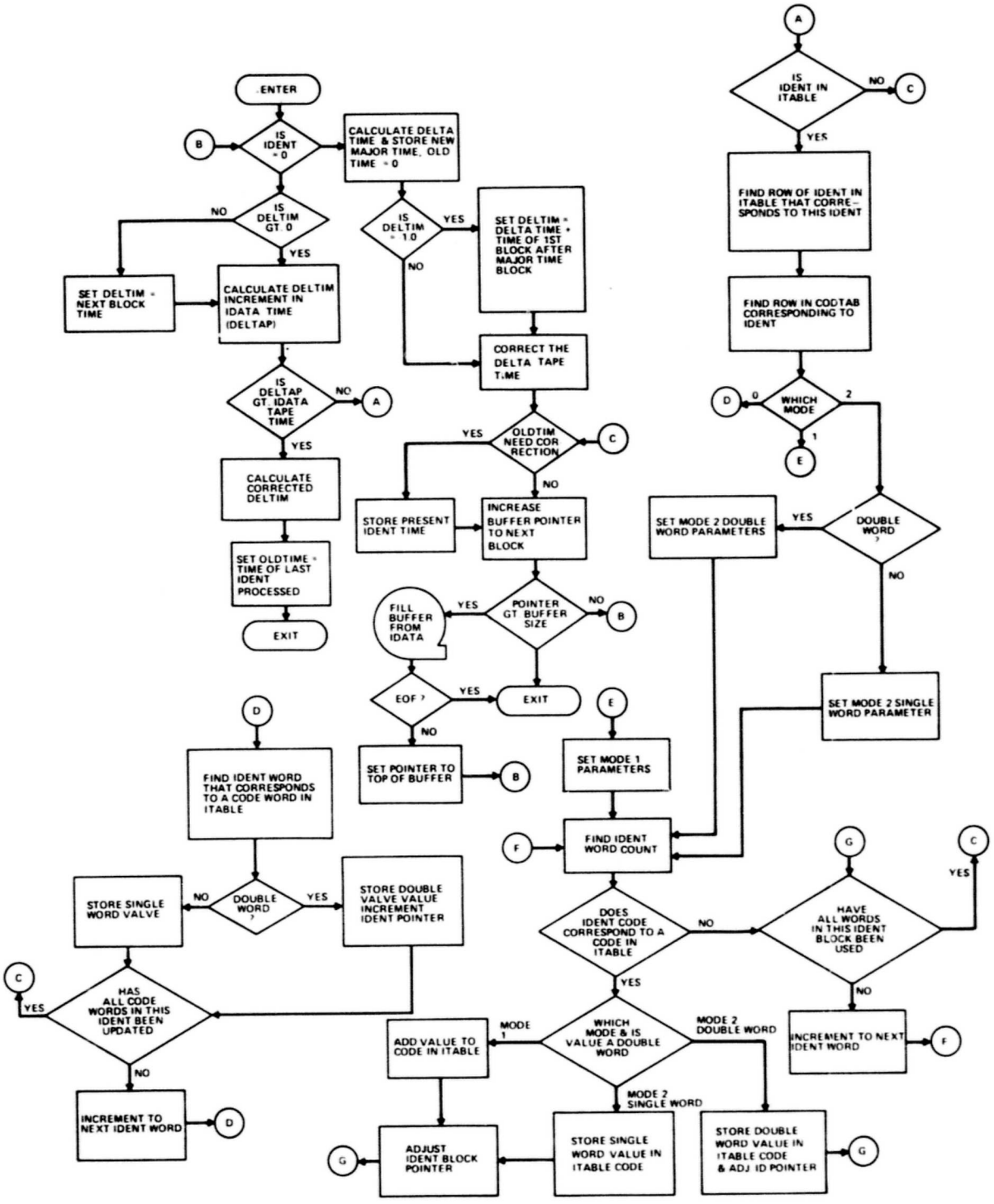


FIG. 10.16 FETCHT FLOW CHART

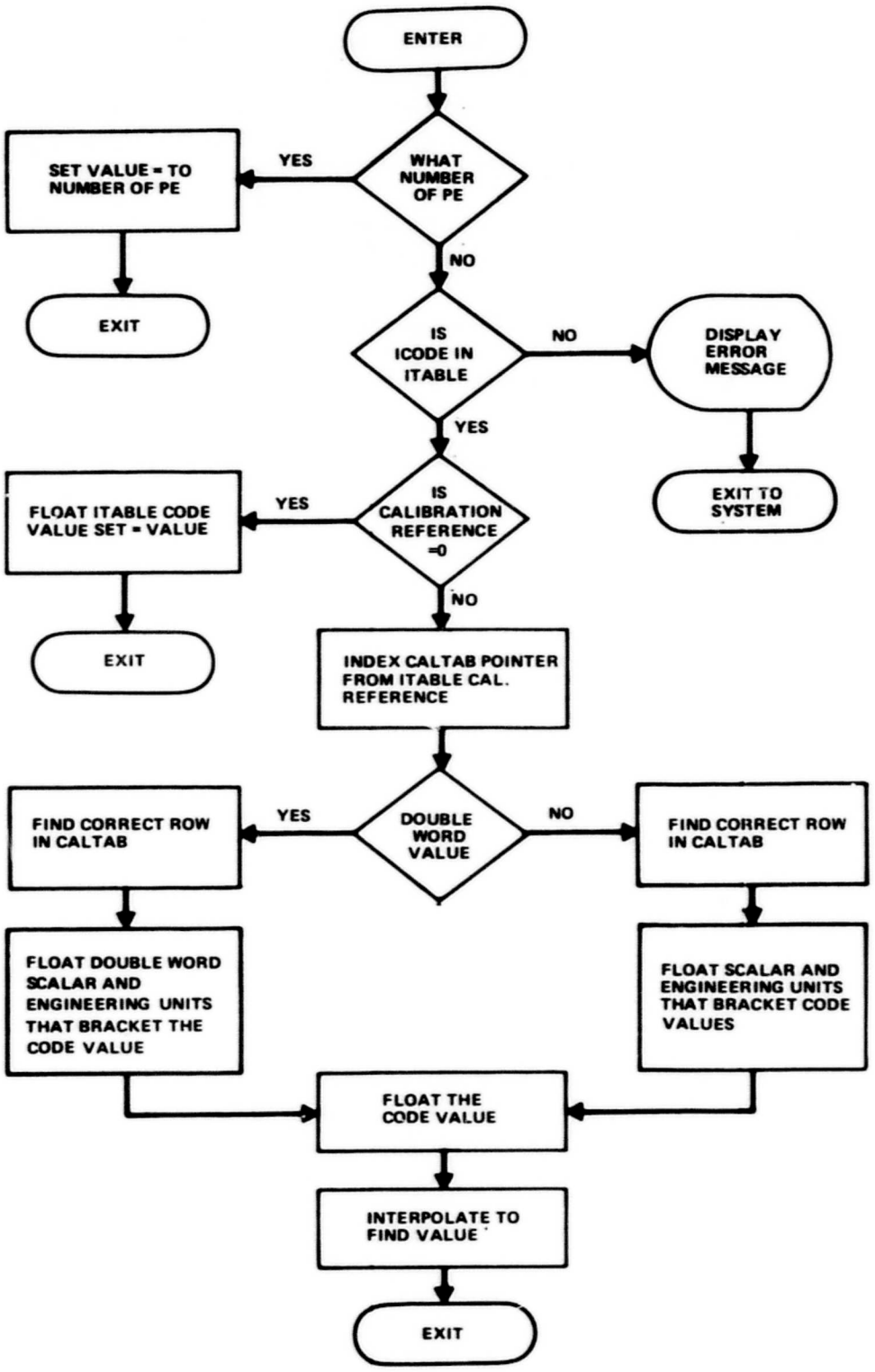


FIG. 10.17 FETCHV FLOW CHART

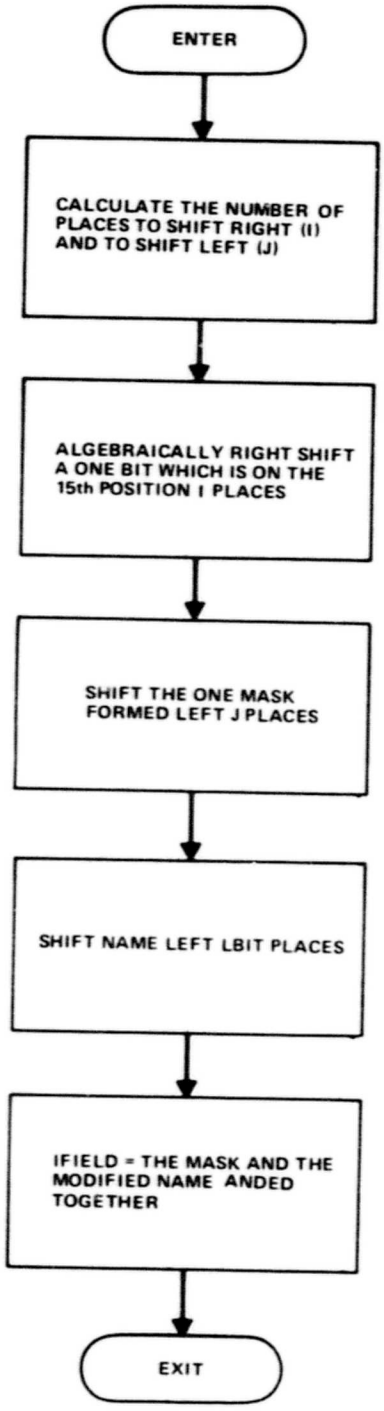


FIG. 10.18 IFIELD FLOW CHART

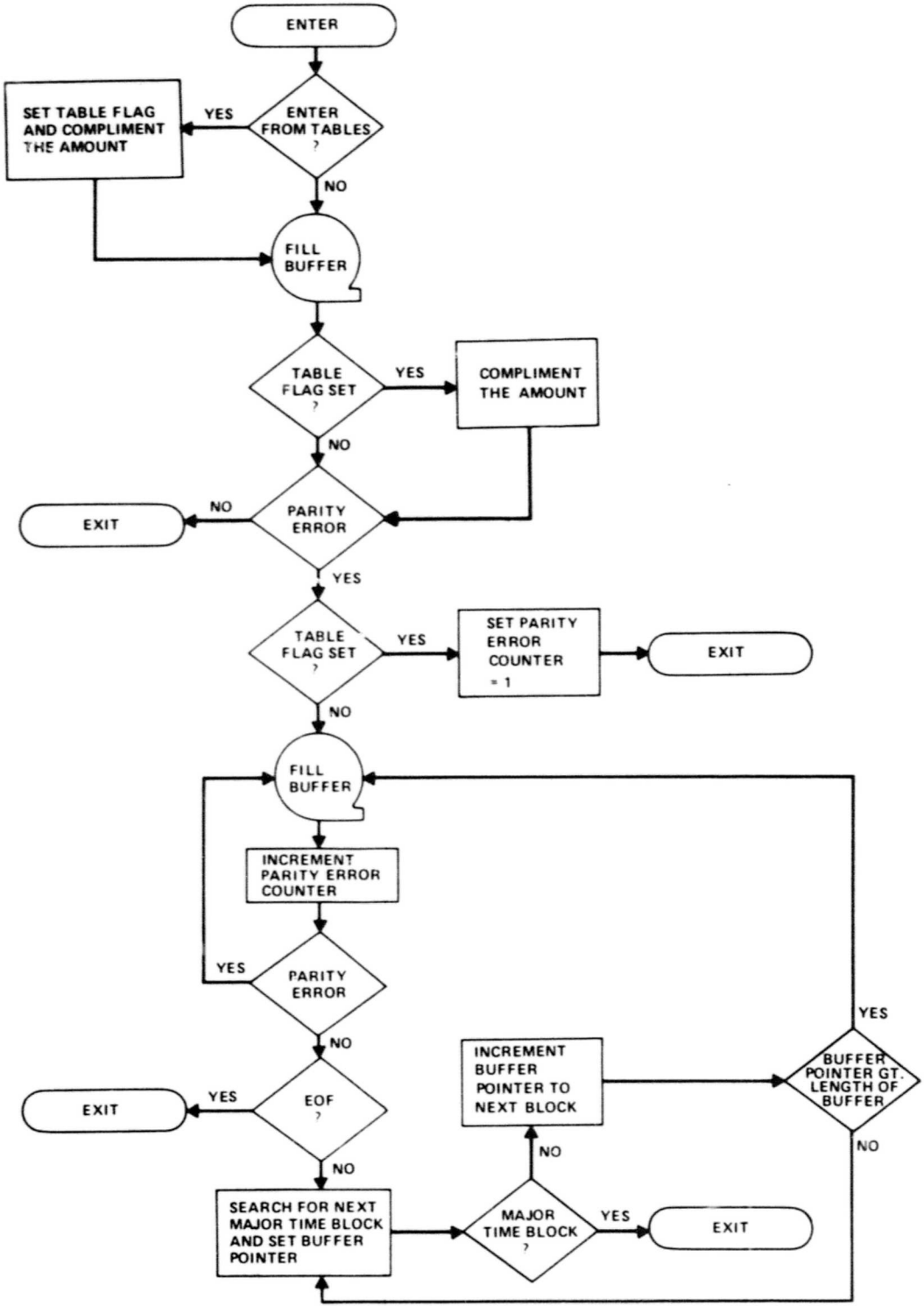


FIG. 10.19 MTIPE FLOW CHART

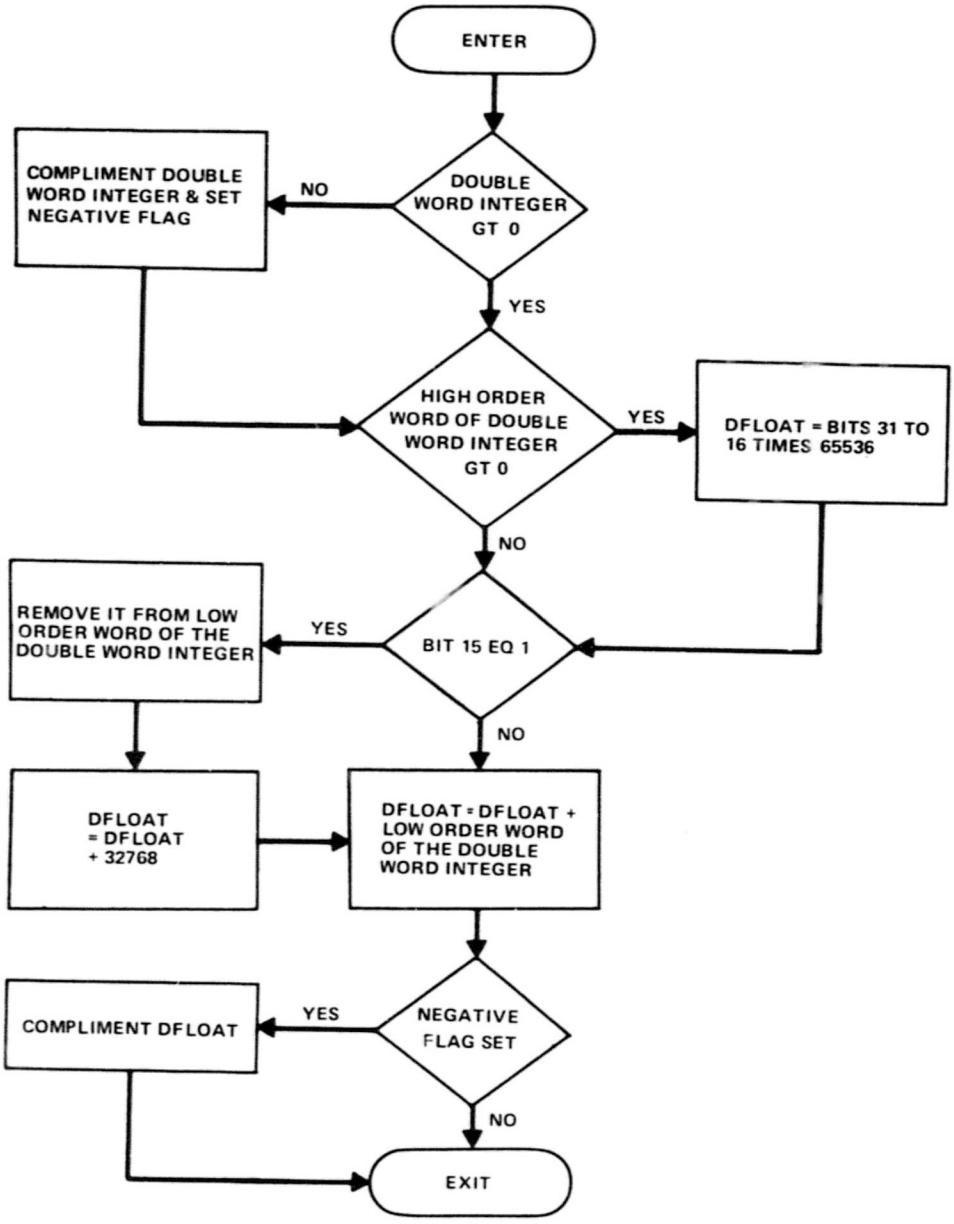


FIG. 10.20 DFLOAT FLOW CHART

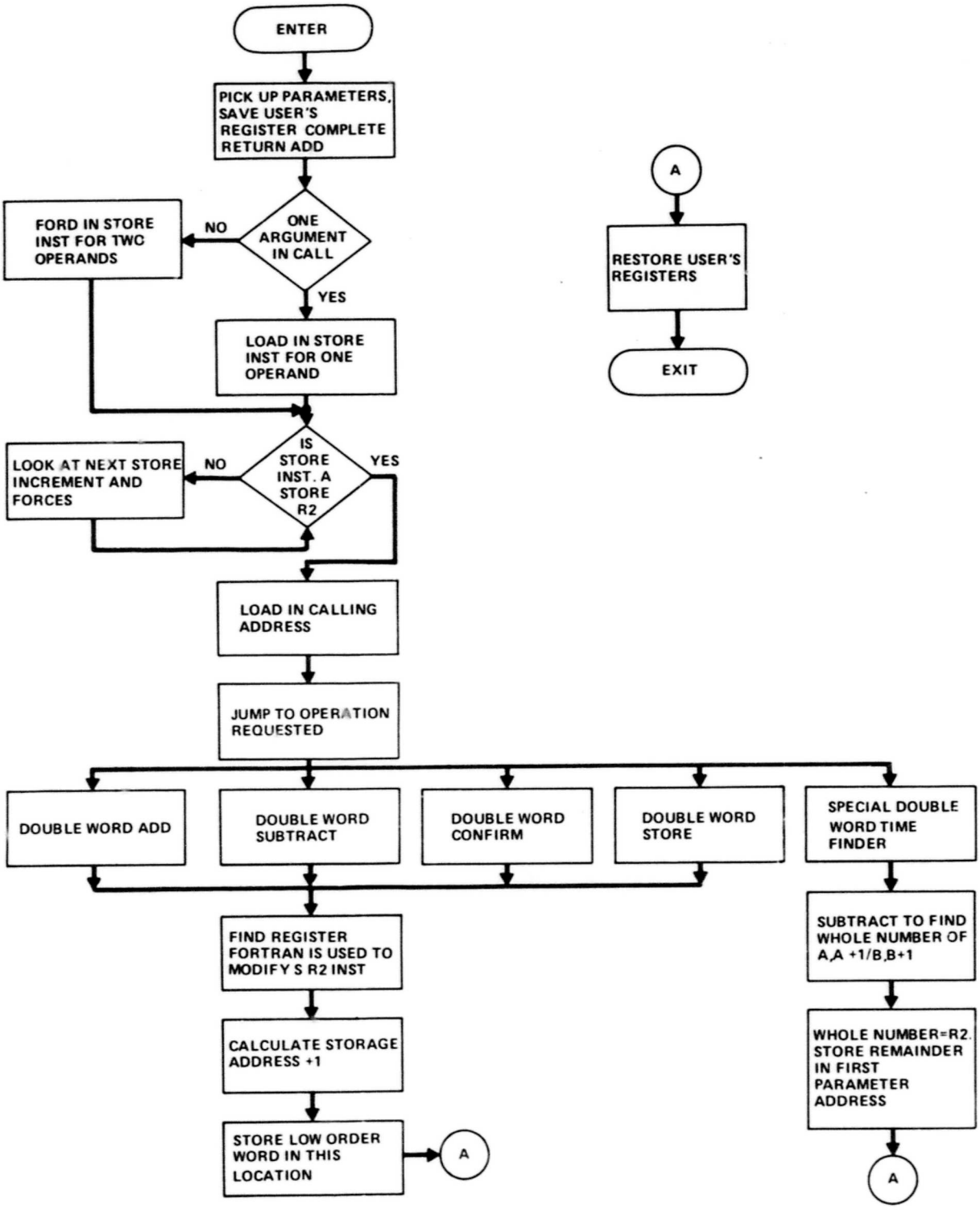


FIG. 10.21 DBLWRD FLOW CHART

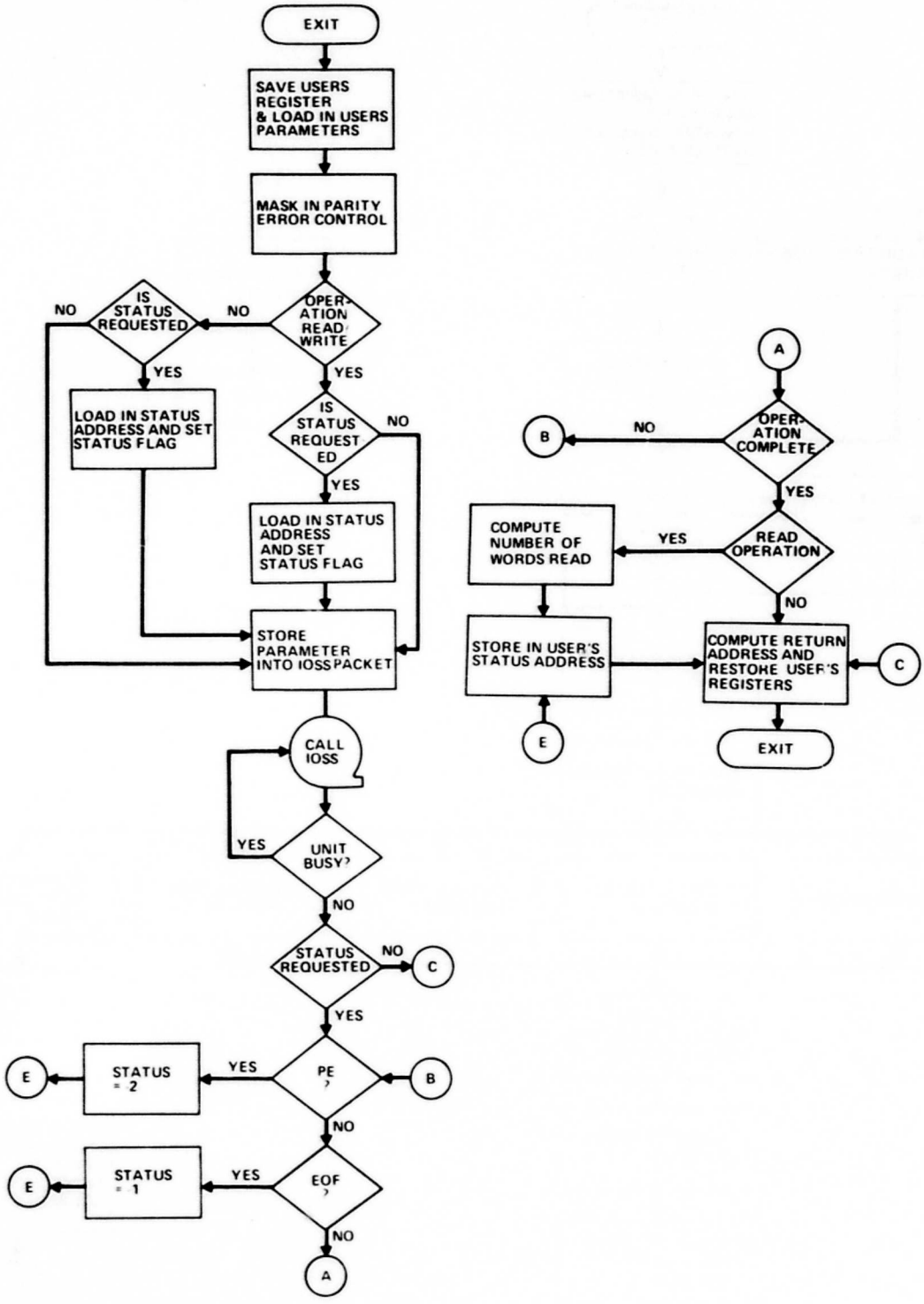


FIG. 10.22 MTIOF FLOW CHART

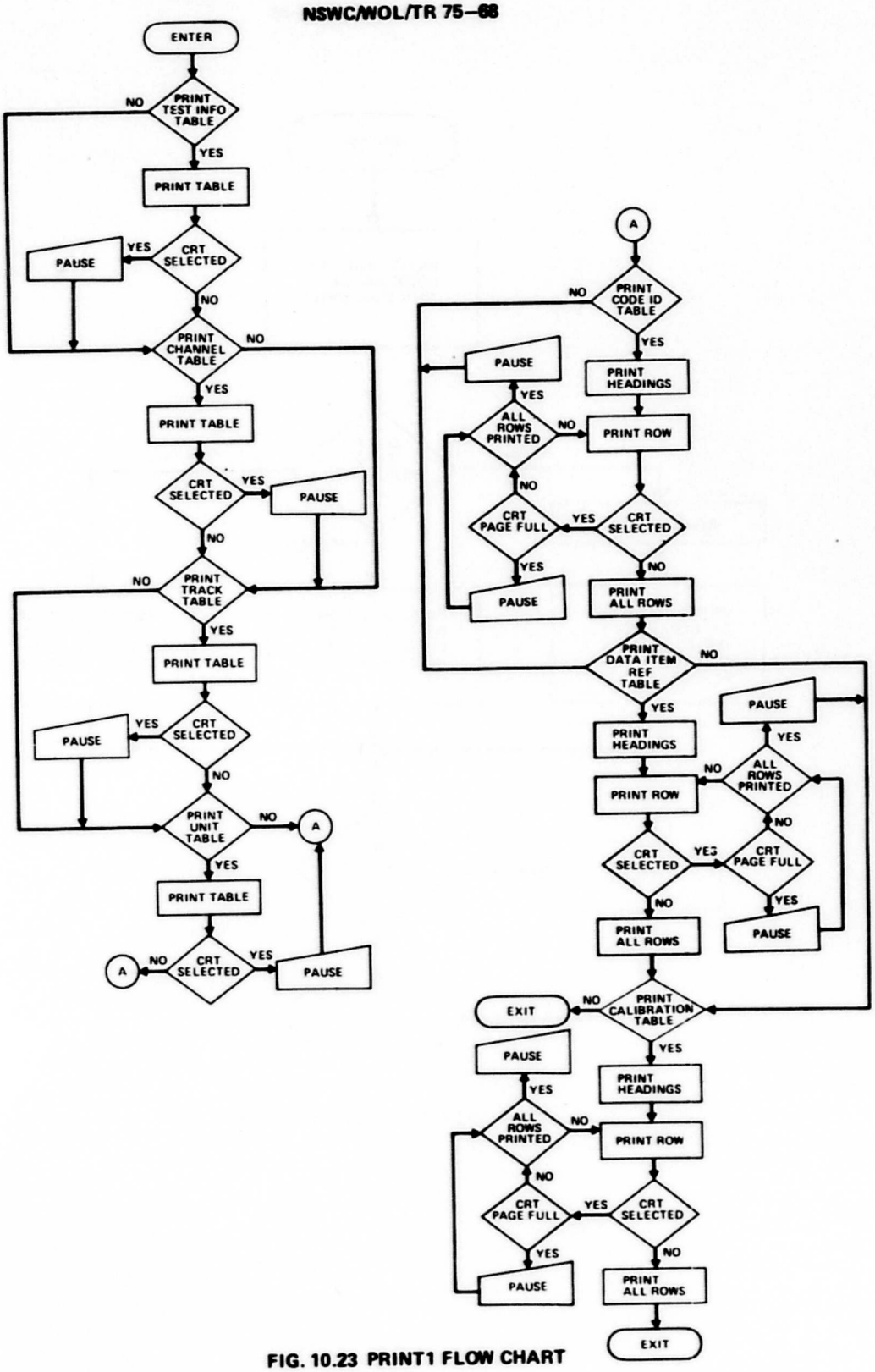


FIG. 10.23 PRINT1 FLOW CHART

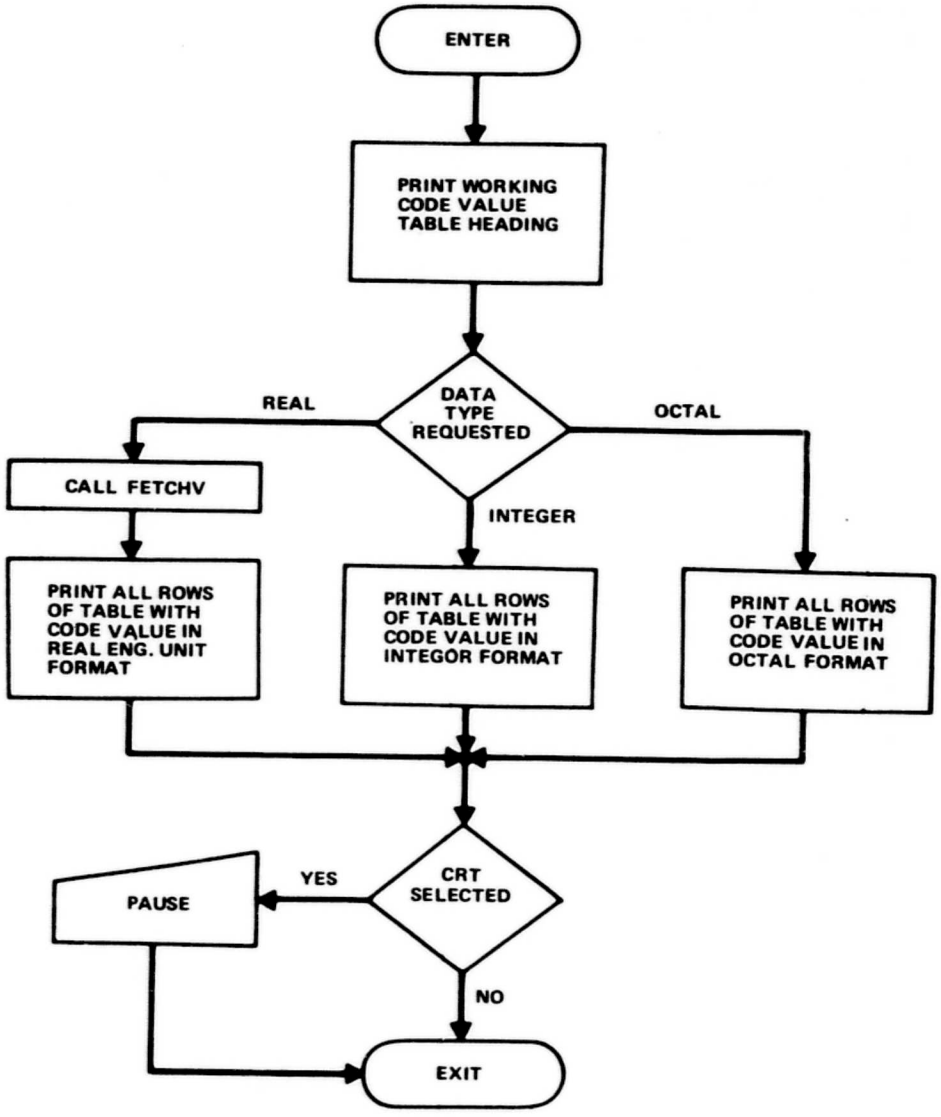


FIG. 10.24 PRINT 2 FLOW CHART