FACTS TIME SHARING MONITOR

Robert E. Enos, Eng.

A THESIS

In

The Faculty

Of

Engineering

Presented in Partial Fulfillment of the Requirements for

the Degree of Master of Engineering at

Sir George Williams University

Montreal, Canada

AUGUST 1972

FACTS TIME SHARING MONITOR

ABSTRACT

A Time Sharing Monitor with the capability of controlling 8 Test Consoles from a Centralized Computer has been developed. A foreground/background philosphy has been designed, with the foreground being able to operate on 1 to 7 Users, while the background processes Compilation, Editing or File manipulation. The Monitor is coded using Digital Equipment Co. basic Assembler language, on their PDP-9 Computer with the Priority Structure based on the Automatic Priority Interrupt facility.

The Monitor features have been verified by software simulations. A prototype manufacturing test system for circuit packs, consisting of two test consoles and three TTY's, is in operation under Monitor Control.

ACKNOWLEDGEMENTS

The author wishes to thank the following individuals for their invaluable help.

Mr. G.M. Martin of Sir George Williams University, who consented to act as the author's adviser and provided helpful suggestions on the scope and subject matter of this Thesis.

Mr. A.G. Vuckovic, Eng., formally the author's Department Manager at the Northern Electric Company, who gave the author assignments which eventually led to this paper. He also provided valuable guidance.

Mr. F. Benedetti, Eng., of the Northern Electric Company, who designed the necessary Hardware Modification required by the author's software.

Mr. R.K.K. Chiu, Eng., and Mr. E. Beneteau, of the Northern Electric, who are writing the User Software packages that will use the author's Monitor, for there help in debugging the Monitor.

Mr. G. Veilleux, of the Northern Electric Company, who assisted in the Editing of the source programs.

Miss S. Enos who gratiously accepted to type the Thesis.

CONTENTS

ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

There are a large number of Computer Systems with varying degrees of software package sophistication available on the Market. They include large Business Computers[1] (superscript number refers to bibliography) which can operate in the Batch, Foreground-Background or the Multiprogramming modes. These systems are designed to optimize computer processing time, C.P.U. time, and in the latter two cases the operating system will mix jobs to optimize peripheral utilization. The users can access the computer system by either giving their programs to an operator for processing, or in some cases, can input their programs via a terminal.

Computer systems specifically designed for Time sharing will optimize user response via terminals. Time Slicing[2] is one technique used to accomplish this. A Quantum of time is granted to each user on a round robin basis or according to some priority scheme based on program size, I/O usage, etc. Normally, the actual C.P.U. time and peripheral usage tends to be inefficient because of the swapping overhead and idle computer processing when an I/O operation is initiated by a user during his time slice.

Process control Time Share systems, as the IBM 1800 System[3], work in a Foreground-Background configuration. The Foreground is accessed when a flag is received from the process hardware, with Idle time being used for program development or processing in the Background. The monitor loads the Foreground with one of the real time software packages and the flag is serviced. This system and systems like the

RSX PDP-15[4], assume that all the Real Time Programs are developed prior to the final installation of the total process control. Modification of these programs and new Real Time programs have to be debugged in an off line environment.

This project was to design a Time Sharing System (FACTS MONITOR) that would permit Real Time process control of up to 8 tests stations, on a priority basis, while allowing for Editing, Compilation, and Data transfer processing. The Author started to design this Monitor in September of 1970 and the final stages of debugging were completed in August of 1972.

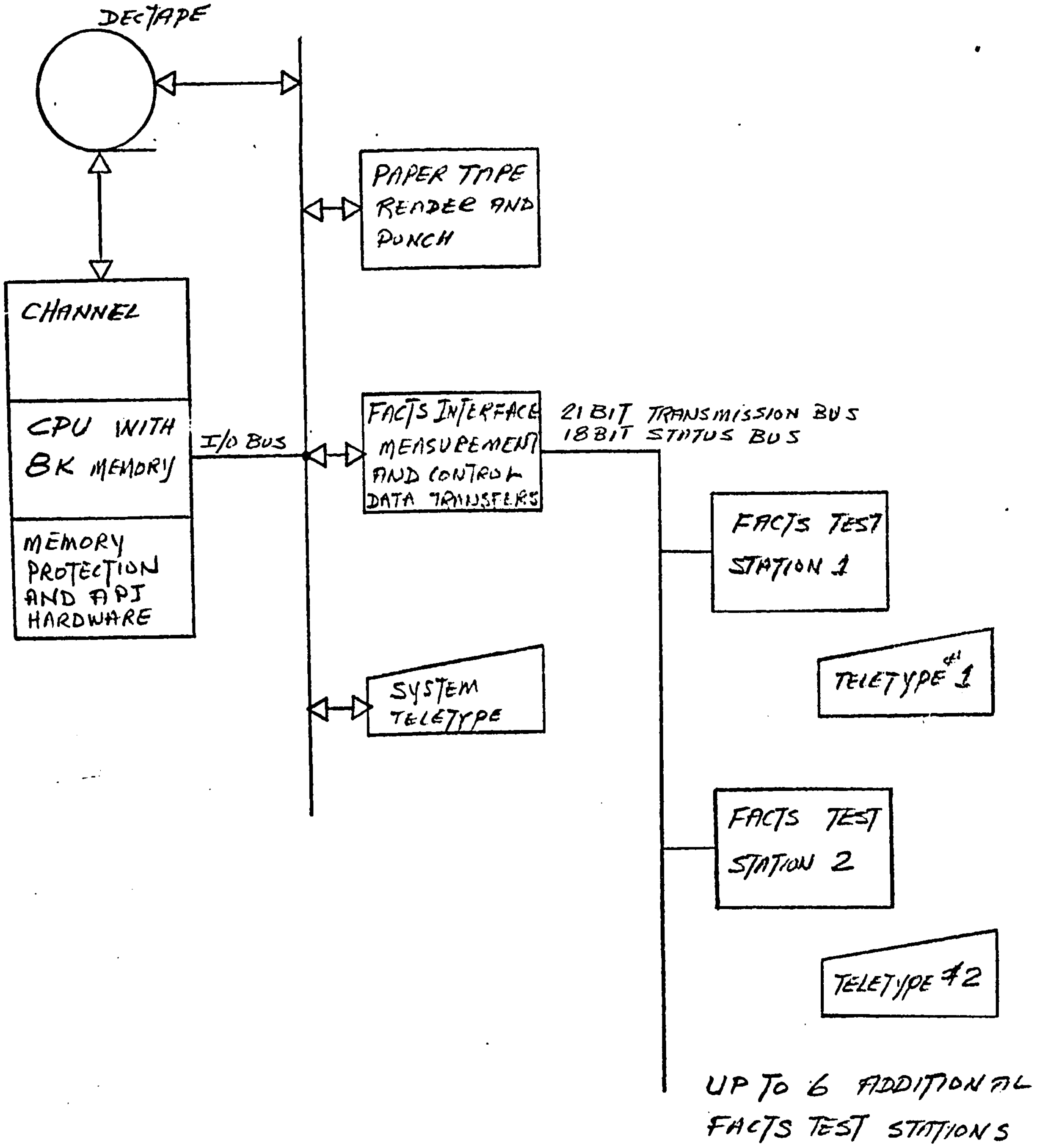
The FACTS (Flexible Access Computerized Test System) software was developed to permit on line process control of up to seven test positions in the Foreground, while permitting on line Editing, TESTRAN compiling and Background Task processing in the Background. A user can Edit a Test Program and Compile in the background and subsequently request processing in the foreground of the object program, while up to six other users are being controlled. Operator response, optimization of hardware peripheral usage, and testing speed are the basic criteria built into the Test Set hardware and the FACTS Monitor software. Figure 1.1 shows a diagram of the Hardware configuration of the FACTS System.

The Monitor was designed to control the I/O and Scheduling of users for the specific testing of relay, analogue and digital circuit modules, but it is applicable to any control process that can be interfaced to a FACTS type Test Station (refer to Appendix "1").

This Thesis presents the FACTS Time Sharing Monitor and I/O Handlers required to control up to eight FACTS TEST STATIONS.

Chapter 2, gives an introduction into the general philosophy of a Time Sharing Monitor. Chapter 3, defines the total FACTS software package and priority structure of the Monitor. The Monitor Executive and Scheduler are described in detail. Chapter 4 presents the I/O Handlers, their features and how they are accessed by the user.

Chapter 5, defines the System User Programs, Op/Sys, Testran Compiler, Editor, BTP, Background TASK Programs, and how they relate to the Monitor. Chapter 6, exposes the verification testing that was performed to demonstrate the features of the Monitor. It includes a description of the Op/Sys and Background Programs used to evaluate the Monitor. Chapter 7, summarizes the detail call procedure for user usage of the I/O Handlers. It also elaborates on the mnemonics for addressing specific handlers in the present I/O package.



CONFIGURATION OF FACTS HARDWARE

FIGURE 1.1

CHAPTER 2

THE STRUCTURE OF A TIME SHARE MONITOR

As Computer Technology evolved through the interaction of users, programmers, and hardware designers, the structure of a Monitor has been generally defined. The Monitor supervises user program running and Computer hardware. It resides in a protected area of core memory and it Monitors all interrupts whether software of hardware generated.

Memory protection for the Monitor is accomplished by suitable hardware that causes an Interrupt if user software tries to access the Monitor, and in some cases, other user areas of core. The Monitor then decides on the legality of the operation. Hardware Interrupts whether internally generated (indication of Hardware malfunctions) or raised by peripherals, are processed by the Monitor via an assortment of device handlers.

The requirements that.a Monitor is desinged to optimize are:

1. Efficient Utilization of Hardware.
2. Common Handling of I/O Devices.
3. Easy User Handling of Machine.
4. User and File Protection.

These requirements are satisfied in varying degrees by all Monitors, depending on the specific application. To achieve these characteristics, the following major component of Monitors have been defined:

1. I/O Control (Interrupt Handling).
2. Job Scheduler and Initiator.

3. Job Supervision (Memory Protection).
4. Error Handling.
5. Operator Communication.
6. Control Program Library.
7. Accounting and Report Keeping.

2.1 - I/O CONTROL

The Interrupt feature, input/output control hardware and software, is what permits time sharing in a computer. There are three main sections in I/O control: the housekeeping functions associated with processing of an Interrupt, the servicing of the Interrupt by a suitable handler, and the initializing of an I/O operation.

When a flag is raised by a peripheral interface, the Computer Interrupt hardware initiates an Interrupt request in the Computer. Depending on the previously set condition, and the sophistication of the hardware, the Interrupt request will be eventually granted. The contents of active registers, different status registers, etc., will be saved in order to be able to restore the interrupted program. The type of Interrupt will be decoded and the proper servicing Handler will be accessed.

There are three generic groups of Interrupts that require handlers. Interrupts that are caused by internal malfunctions of the Computer, such as parity errors in memory or power interruption, internal control Interrupts - i.e. memory protect, clock, etc., Interrupts, and peripheral device Interrupts. The handlers for internal malfunction flags attempt to keep the Computer up by executing several recovery procedures, and inform the operator of the fault. Internal control Interrupts could be used to trigger the scheduling of users, the requesting of an I/O operation, etc. The peripheral handlers perform the action initiated by the flag. The actions include the beginning of data transfers, the completion of transfers, and therefore the initialization of releasing the peripheral from this particular user, etc.

The initialization of I/O operations is triggered by a user requesting an I/O operation via the Monitor. These handlers preprocess the data to be written , or read and prepare the I/O devices to perform the required operation.

2.2 - SCHEDULER

The Scheduler algorithm determines which user Program will be run next. The scheduling criteria is based on predetermined priority levels or calculated priority levels with reference to size and type of Program, estimated run time, availability of core, and real time service to the user. Usually, one particular parameter is optimized such as service to the user, efficient use of C.P.U. time, or efficient use of the peripheral devices. These parameters are not usually compatible.

The Scheduler is entered after an I/O operation is initialized or terminated, or when the watchdog timer causes an Interrupt.

2.3 - JOB SUPERVISION

The Job Supervision Program monitors the jobs as they are being processed in core. It keeps track of the time used per job. If there are errors created by the user program, it will initiate the necessary operator messages and abort the job if required.

All Interrupts are handled by the Job Supervision Program. When an Interrupt is granted, the active registers are stored and the service routine is entered. Recovery processing after the Interrupt is part of the Job Supervision Program.

2.4 - ERROR HANDLING

When an error is detected, usually via an error Interrupt, a number of actions are possible depending on the failure. In the case of a peripheral malfunction, the data transfer is attempted several times, and if the error persists, the peripheral device is "downed", the user is aborted, and the operator notified. If there is a memory failure, the user is aborted, and the memory is exercised with a maintenance program and the defective section is "downed", if possible, and the operator notified. If basic C.P.U. functions are inoperative, recovery is only possible with a dual processor arrangment, which are synchronized and compared logically one to the other.

2.5 - OPERATOR COMMUNICATION

User message to condition peripherals for user programs, error diagnostics on downed hardware, etc., are processed via the operator communication software package. This also permits the operator to manually initialize the system, abort user programs that are defective, or take action when deemed necessary.

2.6 - CONTROL OF PROGRAM LIBRARY

The Program Library is accessed when a compilation is taking place and the user program calls for a subroutine which is only available on the Program Library. This software initiates the directory search of the Program Library and transfers the required subprogram to the user area of core.

2.7 - ACCOUNTING AND RECORD KEEPING

The C.P.U. time accumulated by the user is calculated by adding the watchdog timer quanta times during which the user is processing. In addition, peripheral usage and channel connection times are also added. The software bills the user and keeps statistical records so that the Computer efficiency can be analyzed.

CHAPTER 3

STRUCTURE OF THE FACTS SOFTWARE

The FACTS Monitor is designed to optimize the speed of controlling up to 7 Test Sets while providing user access to do background processing. There are four System User Programs (SUP) defined that are accessible via the Monitor. The FACTS Operating System (Op/Sys), which resides in the Foreground, processes compiled user test programs. The test programs appear as data which are operated on to control Test Sets. The Operating System is re-entrant to the extent that after a request for I/O is initiated, another user can be processed via the Operating System. The next Op/Sys user gets service if no background request is present. If a background request is made, control will be transferred to it when the Op/Sys user makes an I/O request, or if all Op/Sys users are in an I/O wait condition. When an Op/Sys user I/O is complete, it will interrupt the background program and restart processing.

The Testran Compiler has priority in the background. There is one Testran Compiler per Test Set. It compiles the Testran Test programs into ELEMENTAL COMMANDS (EC) which are stored on mass storage and are the input data to the Operating System. The Editor System User Program is next in priority. The Editor permits on line Editing of Testran Programs. The Background Task System User Program (BTP) has the lowest priority. It permits on line housekeeping functions such as transferring files from one device to another.

The priority structure is built around the hardware API[6], Automatic Priority Interrupt, facility in the PDP-9[7] computer, API has four priority hardware levels permitting eight devices, with a unique trap address in memory for each device, per level. In addition there are four software API levels. The hardware levels are assigned as shown in Appendix 6 (levels 0 - 3) and software levels are assigned as follows:

SOFTWARE API LEVEL	
4	Raised upon entry, from a user program or terminated I/O operation into the Monitor.
5	Raised by the completion of an I/O operation.
6	Raised for user service via the operating system.
7	Raised for user service via the compilers, Editor, or BTP.

The FACTS Monitor processes at the API level four and all Monitor request must originate from below this level; since Monitor request from above API level four are not closed out and recovery would not be possible[19].

When an I/O operation, which is processed at API levels 0 to 3 is complete, a request for service at API level 5 is initialized. The processing at API level 5, in turn, requests service at API level 4 to update the status of the hardware handling availability.

Operating System users are queued by the Monitor and are given processing time on a first come first served basis. This gives service to the fastest Test Sets more frequently, commensurate with the speed of testing.

Background processing is granted at API level 7 with priority given to the Testran Compilers. If a compile request is received by the Monitor via a TTY Keyboard, the particular Testran compiler for that Test Set is loaded from mass storage and the Testran program is compiled. If another background user program is in process, it will be swapped out of core upon its entry into the Monitor for an I/O operation and the Testran Compiler will be subsequently loaded. After the compilation is complete, and no other compiler request is waiting, the previous background program will be reloaded into core. The Editor could be swapped out of core either when an I/O request is received by the Monitor or when a TTY I/O operation via the Editor is in process. If the BTP is processing and an Editor request is made, the BTP will be swapped out of core and the Editor will be loaded.

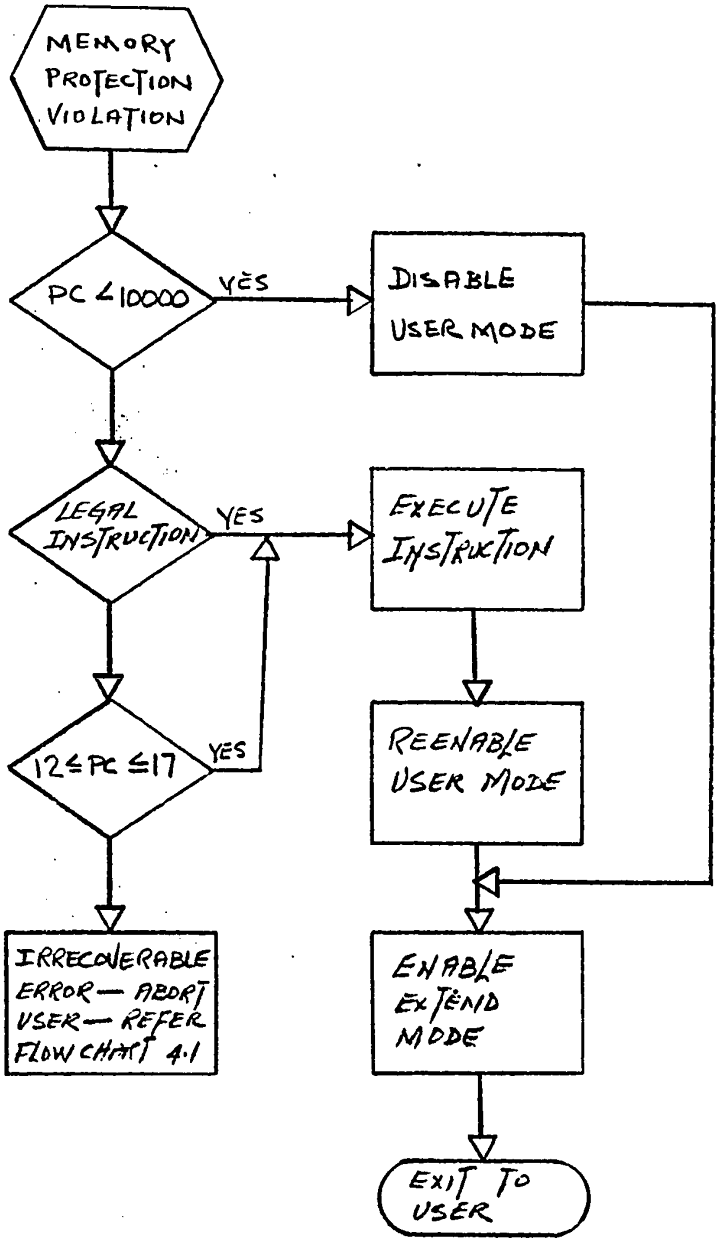
Memory protection and the trapping of illegal Instruction software was designed using the Memory Protection Hardware of the PDP-9 computer. A boundary register and instruction register Monitors each instruction that is executed and breaks to the Monitor when a violation occurs.

3.1 - MONITOR EXECUTIVE PROGRAM

The Monitor is entered upon the execution of a CAL[5] instruction and priority is raised to API level 4. The CAL instruction could be executed as a result of a Memory Protection violation, by a user requesting an I/O or Monitor operation.

3.1.1 - Memory Protection Processing

When the Memory Protection Hardware detects a violation a CAL Instruction is executed and the Violation Flag is set. Upon entry into the Monitor the Violation Flag is interrogated, and if set, the following sequence is processed (refer to flow chart 3.1).



MEMORY PROTECTION PROCESSING FLOW CHART 3.1

If the trapped instruction address is less than 10000, where the Monitor is resident, the User Mode is disabled and control returns to the interrupted program. This occurs when a system program user is being loaded or swapped, which is done at API level 6. If trapping occurs because a user is accessing data in the protected area of core, the instruction is permitted providing it is a LAC[5], XOR[5], or XCT[5] instruction. If trapping occurs by user manipulation of Auto Index registers 12 to 17 all instructions are permitted. All other conditions are illegal and the user executing the illegal instruction will be aborted. The following irrecoverable error message will be typed; "ILCAL" followed by the core location where the violation occurred, in Octal.

3.1.2 - I/O Initiator

The format for the instruction for requesting an I/O operation is as follows:

```
PC          CAL          A B C
PC + 1      BUFFER
```

The CAL instruction causes a JMS[5], Jump Store, to absolute location 20 in core storing the PC + 1 in location 20 with bit 0 indicating the state of the link, bit 1 indicating the state of the Memory Extension Mode[5], and bit 2 indicating if the User Mode (Memory Protection) is enabled.

A Specifies the system user program number in bits 5 to 7.

B Specifies the handler number in bits 8 to 14.

C Specifies the FACTS station number in bits 15 to 17.

Buffer specifies the address where data will be retrieved or stored.

Upon entry into the Monitor the CAL data is processed as shown in Flow Chart 3.2. Legality checks are made on the data in the CAL instruction. For the single threaded handlers, if the I/O request is legal the CAL data, Buffer address and user return pointer is stored in "XGENQ" queue and the device busy status register "XBUSYR" is interrogated. If it is free the device handler is entered and the busy register updated. If busy, the status of the number of users waiting for that specific device handler is updated in table "XSTGEN".

The format of the Busy Status register "XBUSYR" is:

BIT	DESCRIPTION
0	Disk
1	Dectape
2	Card Reader
3	Paper Tape Reader
4	Paper Tape Punch
5	Analogue to Digital Converter
6	Not defined
7	Not defined
8	Not defined
9	Not defined
10	User TTY 8
11	User TTY 7
12	User TTY 6
13	User TTY 5
14	User TTY 4
15	User TTY 3

BIT	DESCRIPTION
16	User TTY 2
17	User TTY 1

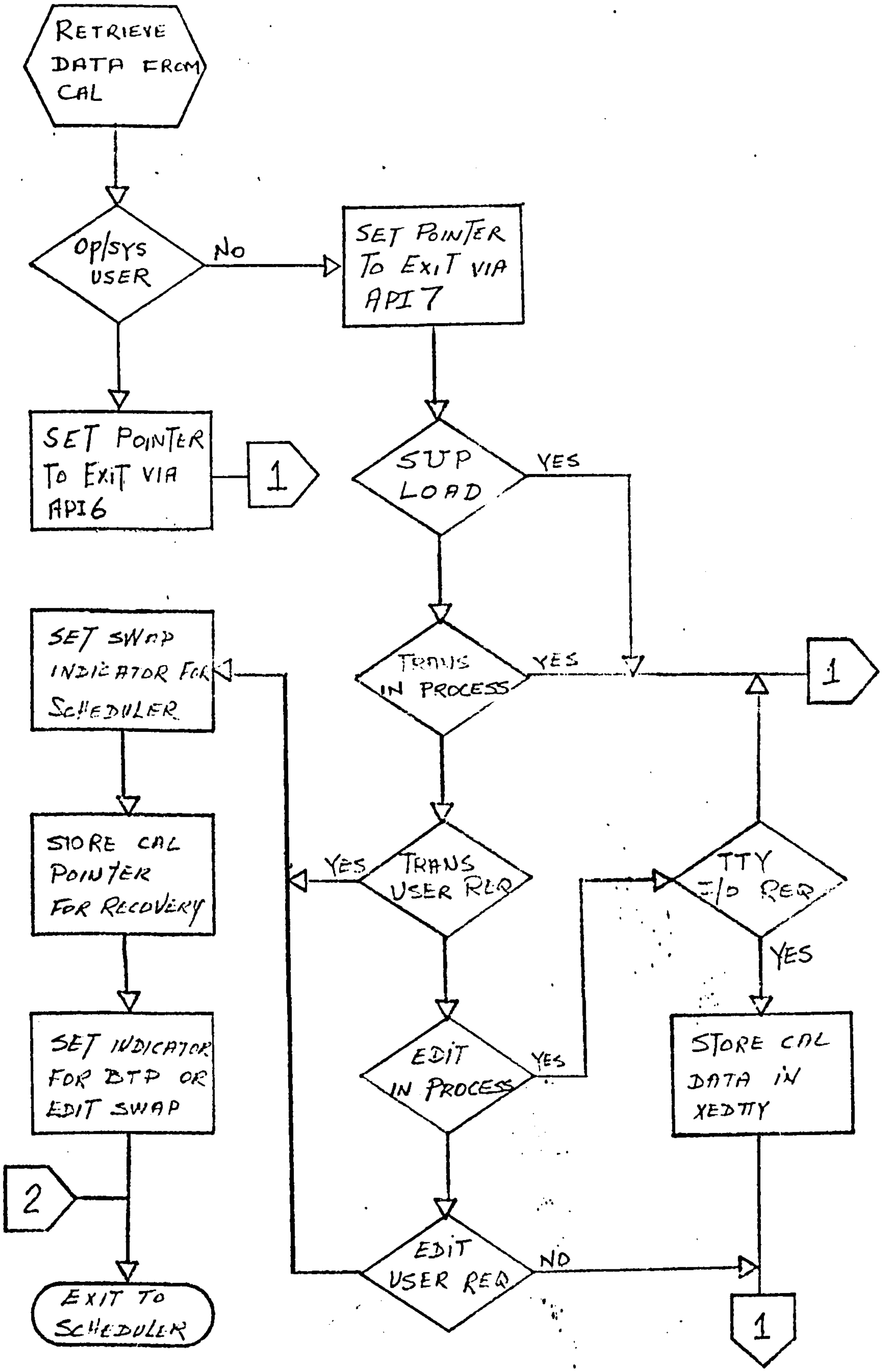
The status of the number of users waiting for a device handler, in "XGENQ" queue, is loaded in table "XSTGEN" as a function of the handler number. The number of bits in a register of table "XSTGEN"represents the number of users waiting.

For the teletype, Keyboard and Print, I/O request data is stored in tables in the Teletype handler directly. For the System TTY there is also an eight entry queue "XSYSTQ" for multiple I/O calls.

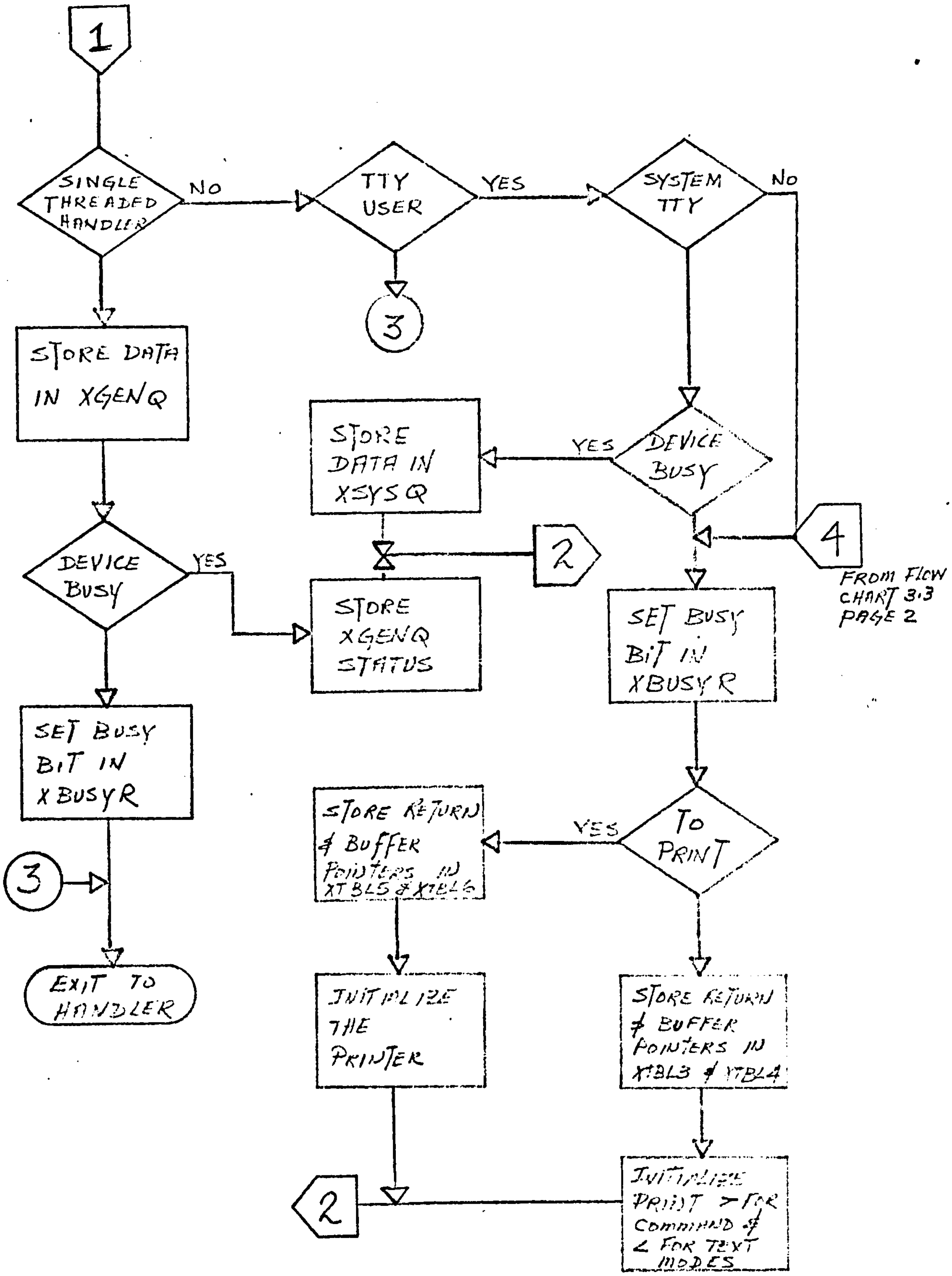
For the multi-threaded handlers, such as the Real Time Handler, CAL data is accessed by the handler upon entry from the I/O Initiator.

The user relinquishes control upon the request for an I/O operation except for Transmissions of data to a FACTS Test Station. In this case control is returned to the user immediately after the Transmission.

If the I/O request is from the background, and a compilation is in process, the request is granted. If the user is the Editor and there has been no compilation request, the I/O is processed, but if a compilation request has been registered, the background Swap routine is entered. If a BTP requests an I/O operation, and no other background request has been registered, the I/O is processed, but if a background request has been registered, the background Swap routine is entered. The Swap routine stores the CAL pointer of the BTP and/or the Editor program, for user recovery, and sets software flags for the initiation



I/O INITIATOR FLOW CHART 3.2 PAGE 1 of 2



I/O INITIATOR FLOW CHART 3.2 PAGE 2 of 2 of swapping by the Scheduler.

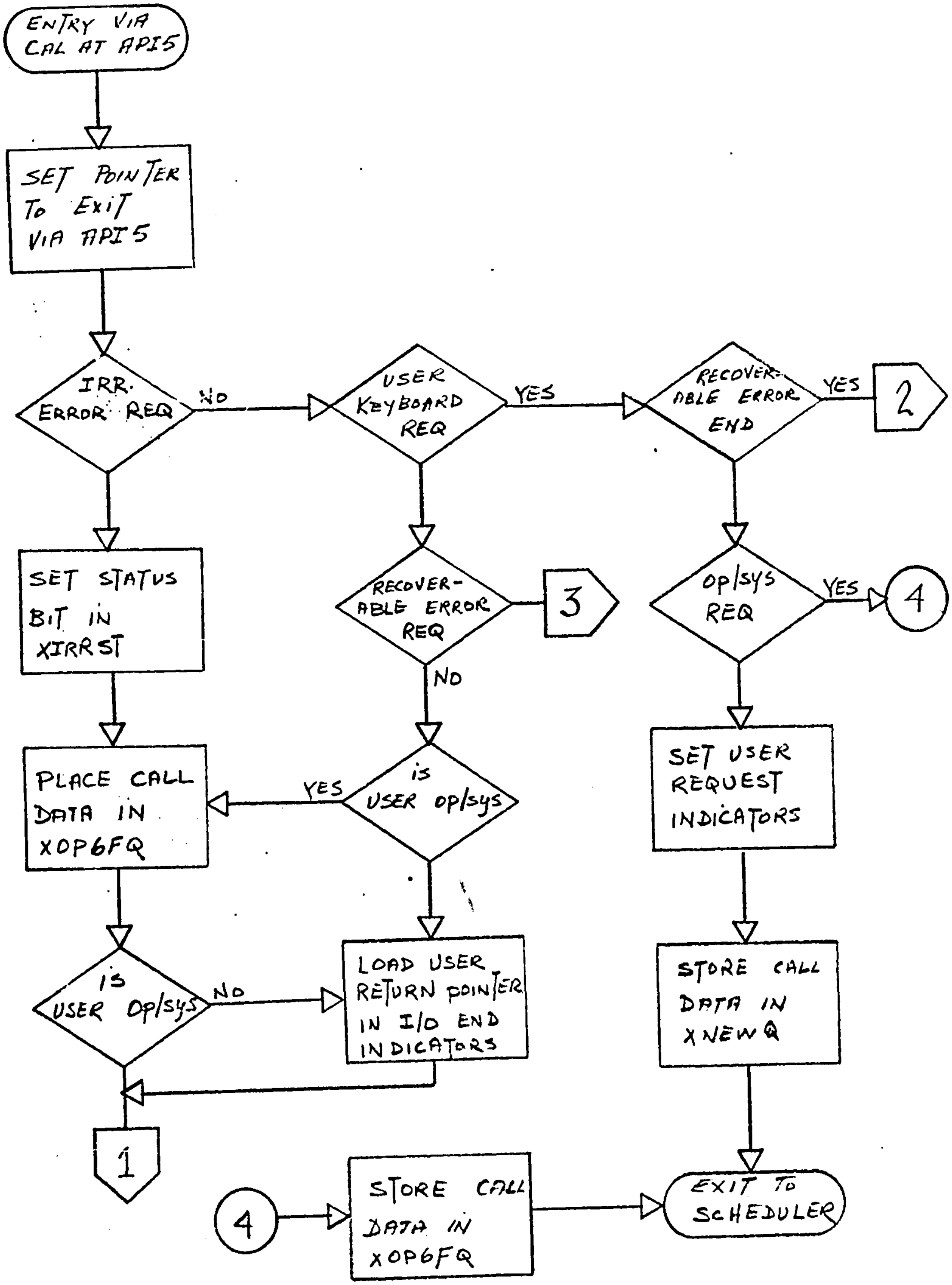
3.1.3 - I/O Termination

When an I/O handler has completed it's processing, or when a user request has been received via the Keyboard Handler, an API level 5 request is made to initiate a Monitor call to register the I/O completion or user request. The I/O handlers which process at API level 0 to 3 cannot directly access the Monitor which process at API level 4, processing at API 5 is used to enter the Monitor.

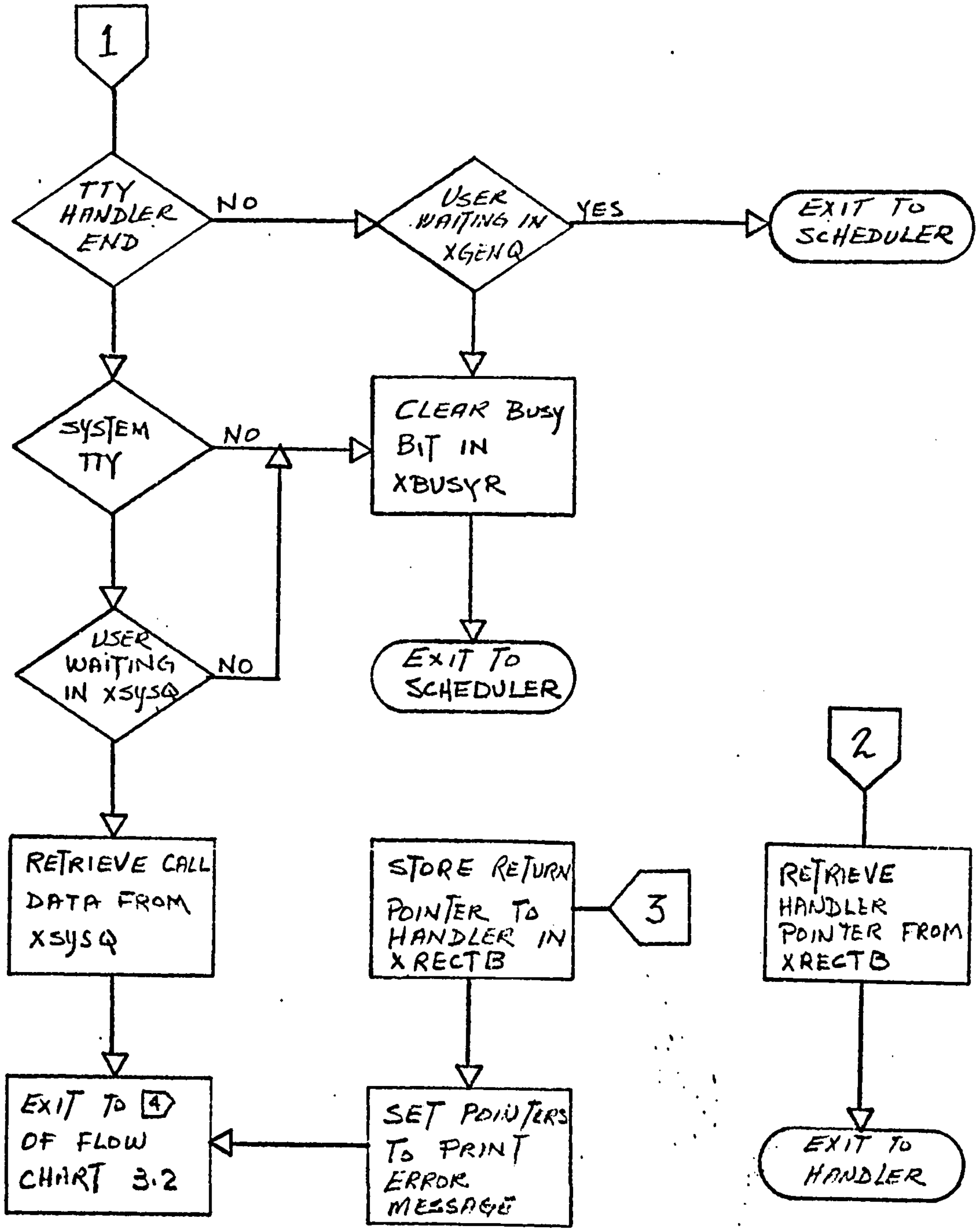
Flow charts 3.3 shows the processing of the call to the Monitor via API level 5. If a user is requesting service of a background program, the call data is stored in the "XNEWBQ" queue and a software indicator is set, for interrogation by the Scheduler. Upon the completion of a background I/O, the return pointer and FACTS station number is stored for user recovery by the Scheduler.

Op/Sys, I/O termination request, or user requests for service call data is stored in "XOP6FQ" queue which is reference by the Scheduler on a FIFO basis.

Irrecoverable and Recoverable error print requests are also processed by the I/O terminator. For recoverable errors, the return Pointer to the I/O handler is stored in "XRECTB" table with reference to the FACTS Station number and the Print Handler is initialized to type the error message. When the user has taken the required corrective action and typed a ↑R, the I/O Terminator returns control to the Handler via the "XRECTB" table. For Irrecoverable errors the call data is stored in the "XOP6FQ" for all System Program users and an irrecoverable



I/O TERMINATOR FLOW CHART 3.3 PAGE 1 of 2


1
TTY HANDLER END
NO
USER WAITING IN XGENQ
YES
EXIT TO SCHEDULER
SYSTEM TTY
NO
CLEAR BUSY BIT IN XBUSYR
USER WAITING IN XSYSQ
NO
EXIT TO SCHEDULER
2
RETRIEVE CALL DATA FROM XSYSQ
STORE RETURN POINTER TO HANDLER IN XRECTB
3
RETRIEVE HANDLER POINTER FROM XRECTB
EXIT TO 4 OF FLOW CHART 3.2
SET POINTERS TO PRINT ERROR MESSAGE
EXIT TO HANDLER

status bit is set in "XIRRST". If the user was processing in the background, the I/O complete indicator is also set for the Scheduler to complete the abort procedure. The Irrecoverable error print out is processed as if the user was a foreground user and at its completion, the Irrecoverable status register is cleared. The bit assignment for the status register "XIRRST" is assigned relative the FACTS Station Number.

3.1.4 - Queues

There are two types of queues used in the FACTS Monitor:: a table where data are referred by the table name and a multiple of the FACTS Station number and a Circular List[8] to store requests for single threaded device handlers or user requests for service.

There are two types of circular lists, one to store two data words, one to store 3 data words.

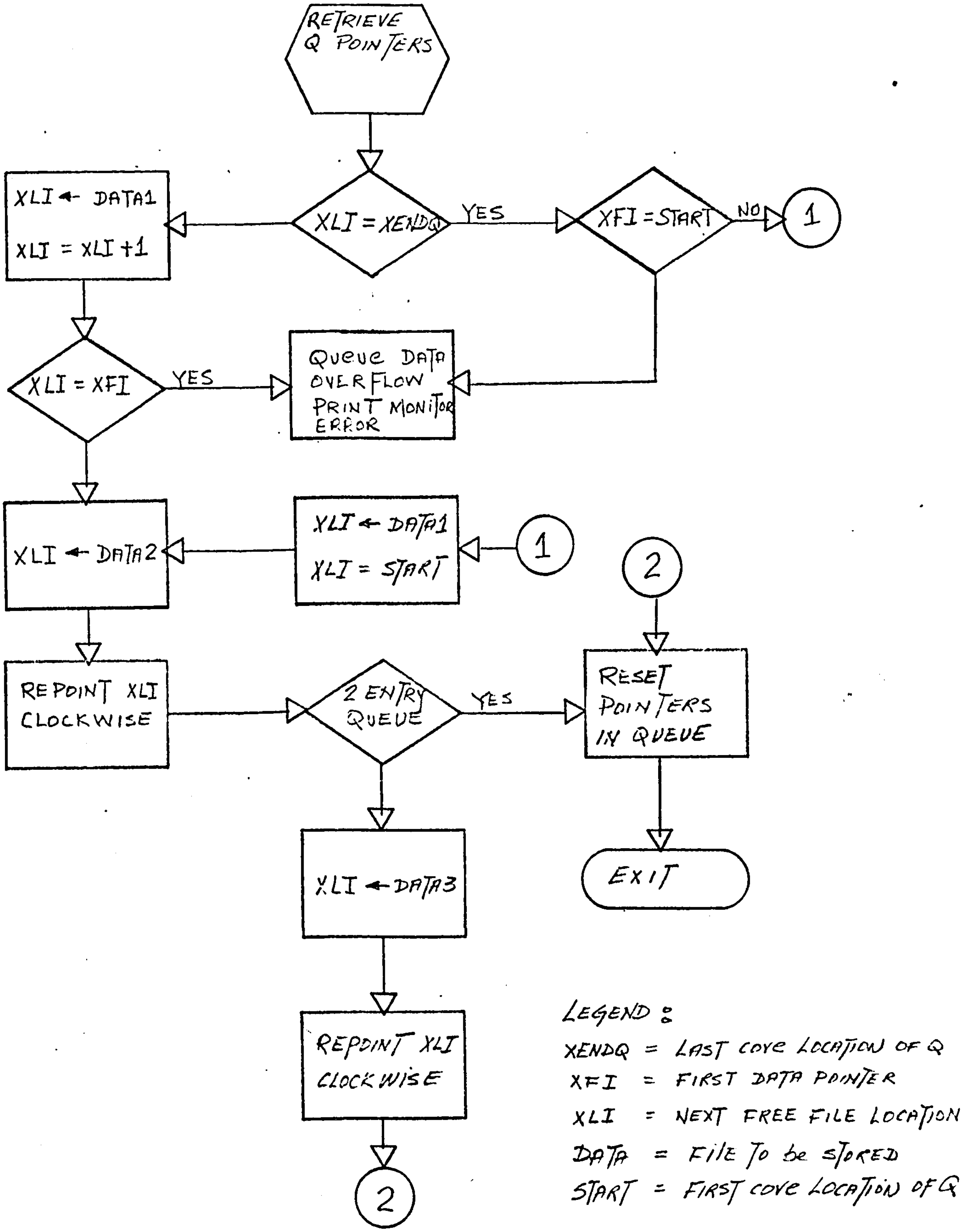
The first three entries of the Circular List specifies the last core location of the list, the first data entry , and the next free data entry. The 3 data entry list, to store eight possible entries, has 4 + (3 X 8) locations and the two data entry lists consists of 4 + (2 X 8) locations, allowing one free entry for overflow checking.

The data is stored by the "XSTORQ" sub routine as shown in Flow Chart 3.4. The call for this sub-routine is followed by the name of the queue, starting core location, prefixed by law[(5)] for a 2 entry queue. The data stored for a 3 entry queue, consists of the CAL information, followed by the return pointer to the user and the buffer address where the data is to be retrieved or stored. For a 2 entry queue the data stored is the CAL information followed by a pointer.

Upon entry in the "XSTORQ" sub-routine the queue parameters are referenced. If the next free data entry is at the last location of the queue, and first user to be accessed from the queue is not in the first data location of the queue, then the data is added to the queue, otherwise an overflow condition will exist. The first data register is accessed and stored. If the next data entry address in the queue is equal to the first data file address that can be accessed from the queue, then an overflow condition will occur.

Before each register of data can be added to the queue, a check is made so that if the next free location is at the end of the queue, the next data entry will be at the beginning of the queue. If a two register data queue is specified in the call, the routine is EXITED after two entries have been made. After the processing is complete, the parameters of the queue are stored in the FIRST three locations.

The data is retrieved from the queues by the "XGETUQ" sub-routine, as shown in Flow Chart 3.5. The call for this sub-routine is followed by the name of the queue prefixed by an XCT^5 to indicate a two entry file. If prefixed by a LAC^5, the scanning of the queue is made for a match fo the System User Program as specified in the "XRQSYS" register; otherwise, it masks for the handler number as specified in "XHARDN" register. The information in the queue could be retrieved from any part of the queue, and if it is interstitial between the first and the last entries in the queue, the queue will be pushed down to fill the void. There are two or three storage registers following the name of the queue in the call up for retrieval of the file data. The following



LEGEND :

XENDQ = LAST CORE LOCATION OF Q

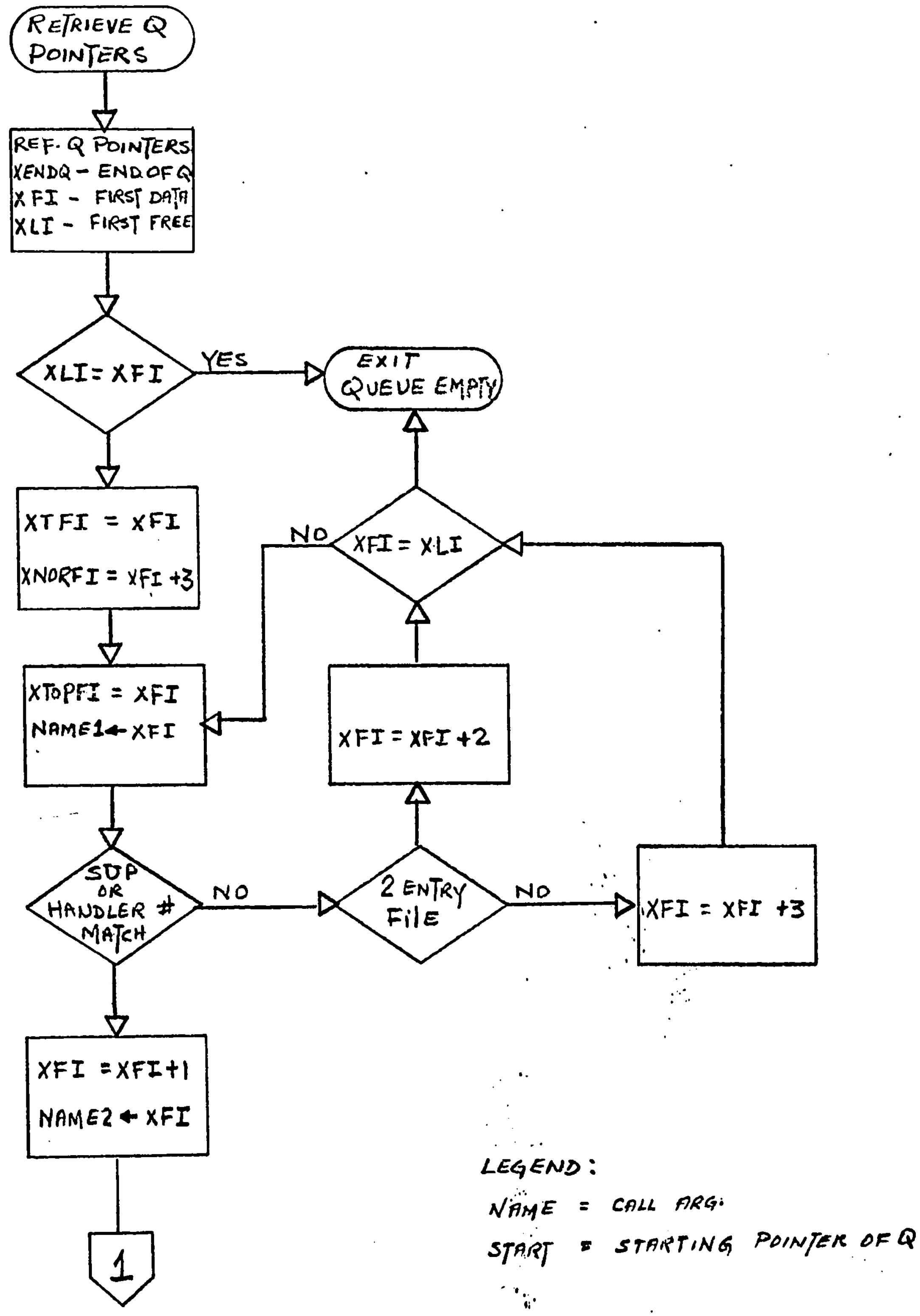
XFI = FIRST DATA POINTER

XLI = NEXT FREE FILE LOCATION

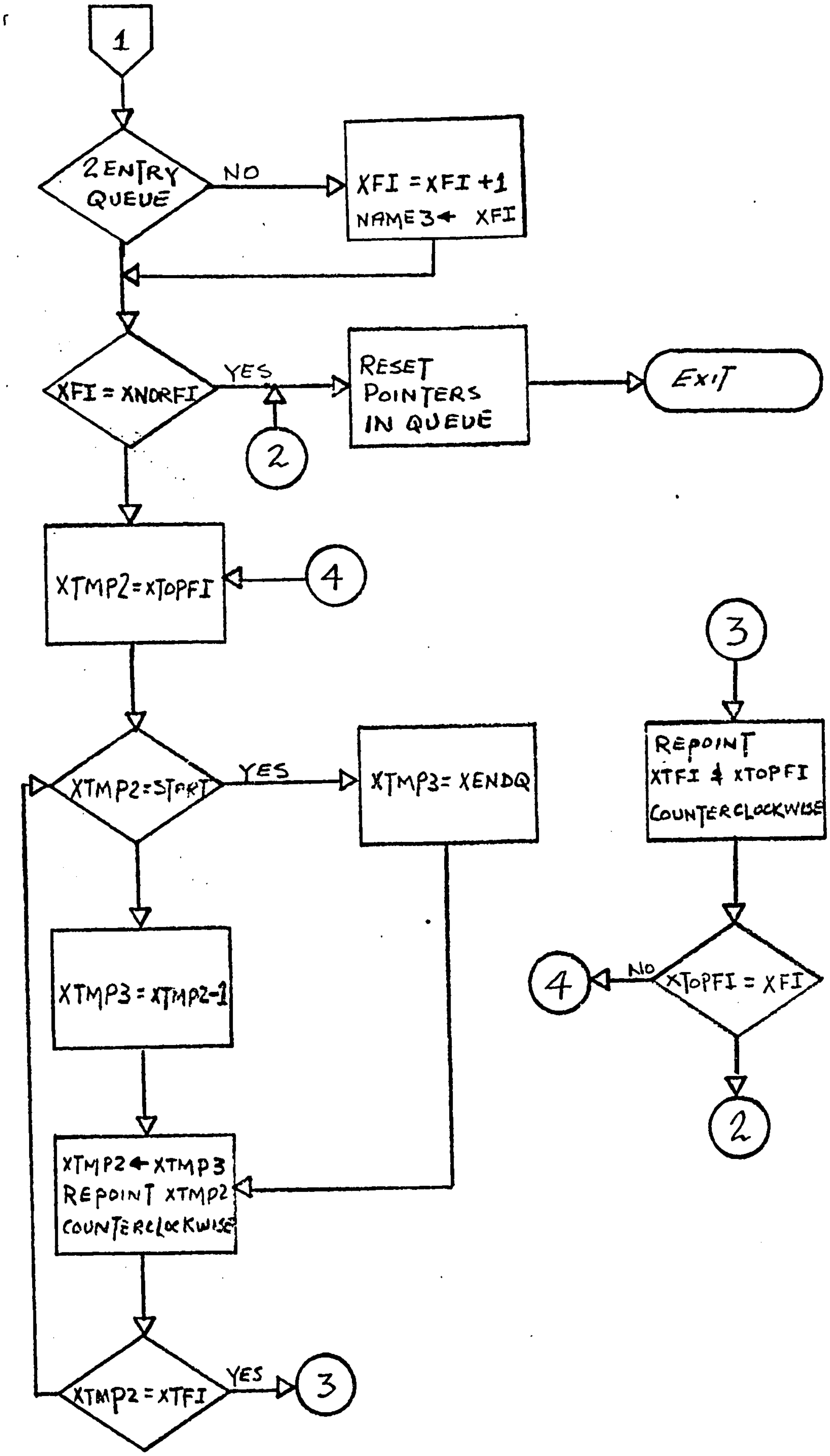
DATA = FILE TO BE STORED

START = FIRST CORE LOCATION OF Q

"XSTORQ" SUBROUTINE FLOW CHART 3.4



"XGETUQ" SUBROUTINE FLOW CHART 3.5 PAGE 1 of 2



"XGETUQ" SUBROUTINE FLOW CHART 3.5 PAGE 2 of 2 location is an error return, file not found in queue. If the file is found in the queue, control returns at the location following the error return location.

Upon entry in the "XGETUQ" sub-routine, indicators are set according to the prefix of the queue name and the queue parameters are accessed. If the queue is empty, the sub-routine exits to the error return in the call up; otherwise, pointers of the first file entry are stored and the CAL entry in the file is accessed and masked either for the handler number, or the System User Program number, as specified in the sub-routine call.

If there is a match, the two or three entries in the file are accessed and stored in the registers specified in the call. If there is no match, the next data file is accessed and the process is repeated. When a match is found, the file is accessed and stored in the call registers. If the match is not found in the first file entry, the remaining files, before accessed file, are moved down to replace the reference file. Each file is moved down one by one until the top of the queue is reached. The pointers are incremented by 1 and the process is repeated two or three times depending on the queue.

3.2 - MONITOR SCHEDULER

The Scheduler initiates service to users processing at API level 6 & 7. If there is processing at API level 6, it is uninterrupted until this user makes an I/O request, except for I/O interrupt servicing. The request is honored by the I/O Initiator which subsequently calls the Scheduler. The Scheduler terminates service at the user API level that is waiting for an I/O operation & control reverts back to the user processing at the other API level, if one is ACTIVE at that time. Control is therefore switched between foreground and background processing whenever an I/O operation is requested. When a foreground user I/O is complete and a background user is processing, the background user will be interrupted to give service to the foreground user.

Each user has a buffer of 256 locations in core for one page of his program to reside, with an additional 256 location buffer for parameters. The normal processing time for a page is normally 30 milliseconds, if there is no I/O calls in the user program. Therefore, the maximum time a user can process is 30 msce. before an I/O request is made to load the next page of the user program. Since the Scheduler assigns processing time on a FIFO basis, under worst conditions (7 users) this user would regain processor control after 6 X 30 msec. plus the time required for the I/O, which was requested, to be completed.

The programs that are processed in the background require a maximum of CPU time by design, whereas the foreground program which controls real time processes is I/O bound for the most part. These I/O operations are normally from 1 to 10 msec.. in duration, and since the foreground has priority over the background, normally 6 foreground users will each process an I/O request before one page of the background userprogram will be completed.

Flow chart 3.6 shows the detailed processing of the Scheduler. A user requesting background processing service is placed in the "XNEWBQ" via the TTY handler through the I/O Terminator. Software indicators are set specifying the number of requests for different background programs. In addition, indicators are previously set specifying what background program is in process, whether a Swap is in process, and what programs have been swapped.

The Scheduler INTERROGATES these software flags giving priority to the Compiler, Editor, and BTP, in that order. If no background program is in process, the user request for background service is interrogated. If one is active, the call information in "XNEWBQ"is accessed, via the "XGETUQ" sub-routine, and control goes to System program loader via a request for service at API level 7. The "XTRANS" Indicator is forced up during the loading process to guarantee non-interruption while loading. When loading is complete, the respective in process indication is set and control goes to the system program, with API 7 being requested via the user entry table in the keyboard handler, so that the file name that the system program will operate on can be accessed.

When the BTP or Editor is active, and a user request for the Editor or Compilation service is registered, the Scheduler will initiate a Swap of the BTP or Editor via the Swap Output program, when an I/O background call is attempted. Upon completion, the requested System Program will be loaded and control will be transferred via a request for processing at API level 7. When this program is complete, and no higher priority program has been requested, the previously swapped program will be reloaded and control will be returned at the location where it had been interrupted. The BTP can only be interrupted when requesting an I/O. The Editor program can be interrupted when requesting an I/O or when a Teletype Print or Keyboard I/O is in process. In both cases peripheral interrupts will be handled but control will revert to the background at their completion. There can be up to 7 Operating System program users being processed at API level 6. Request for service and I/O completion data is stored in "XOP6FQ" queue. The Scheduler interrogates this queue and if a user is waiting control is transferred via a request for processing at API level 6. The Operating System user program is loaded at cold start and is always resident.

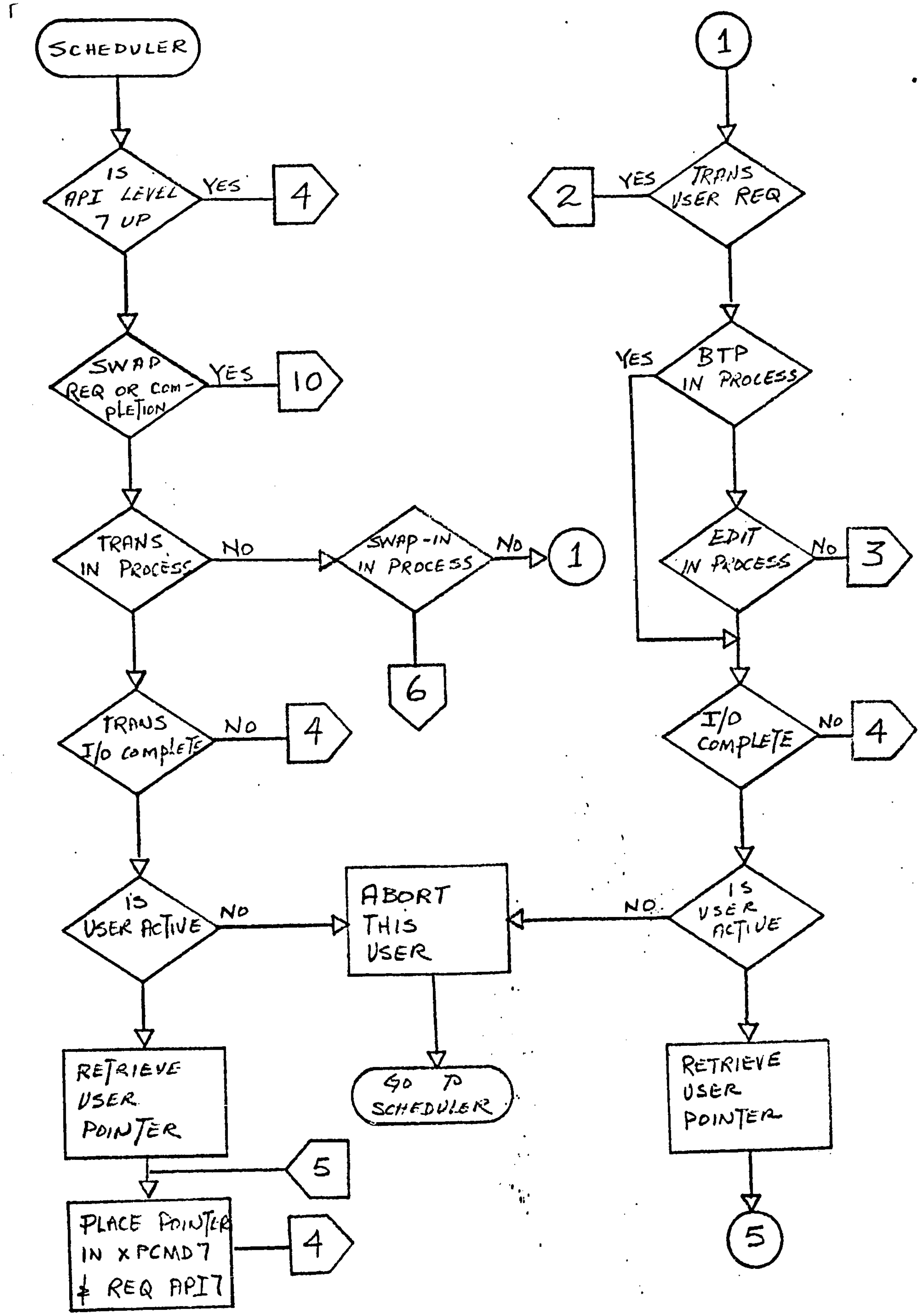
If a ";" is typed at any time, which indicates that the user wishes interaction with the Op/Sys, this fact is stored in table "XSEMIR" with reference to the FACTS Test Set station number. This table is interrogated by the Scheduler and if a ";" has been registered, control is returned to the Op/Sys Semicolon Handler with the normal return being saved in the table "XSEMIR" for recovery by the user, if requested.

There is an Activity Table "XTBL1Ø" that is interrogated with reference to the FACTS number prior to control going to any System User Program. This table indicated if the user is still active or if an abort indication has been received by the user via the Keyboard handler. If active, the user gets service, if an abort request is registered, the user is aborted.

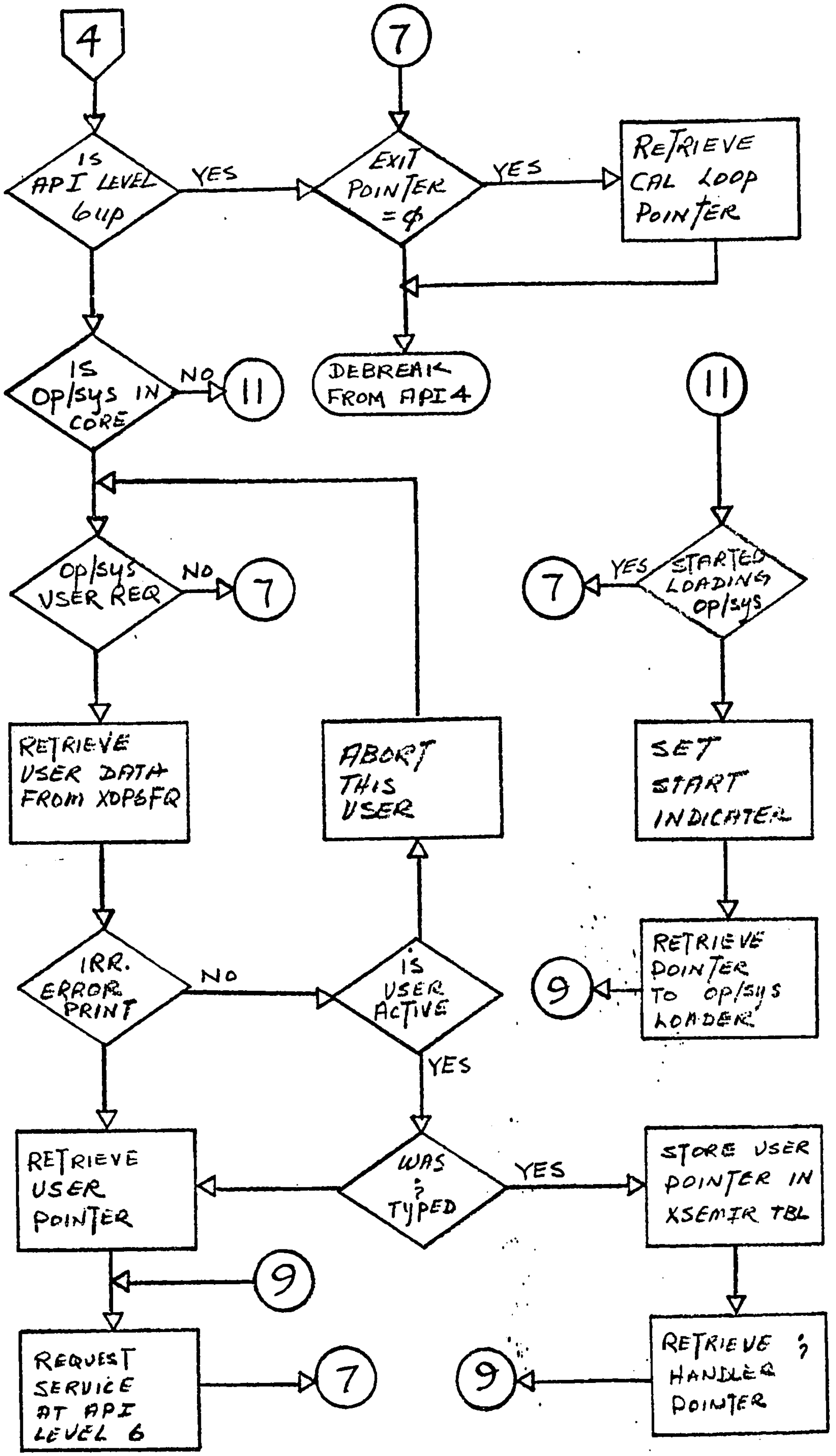
The FACTS station number which is defined by the TTY that is requesting service, defines unequivocally each user. All data in tables **are accessed** by the starting core location of the table and the FACTS station number for each user. When control goes to one of the System User Programs, this number is accessed and all I/O calls or internal processing is made with reference to this number.

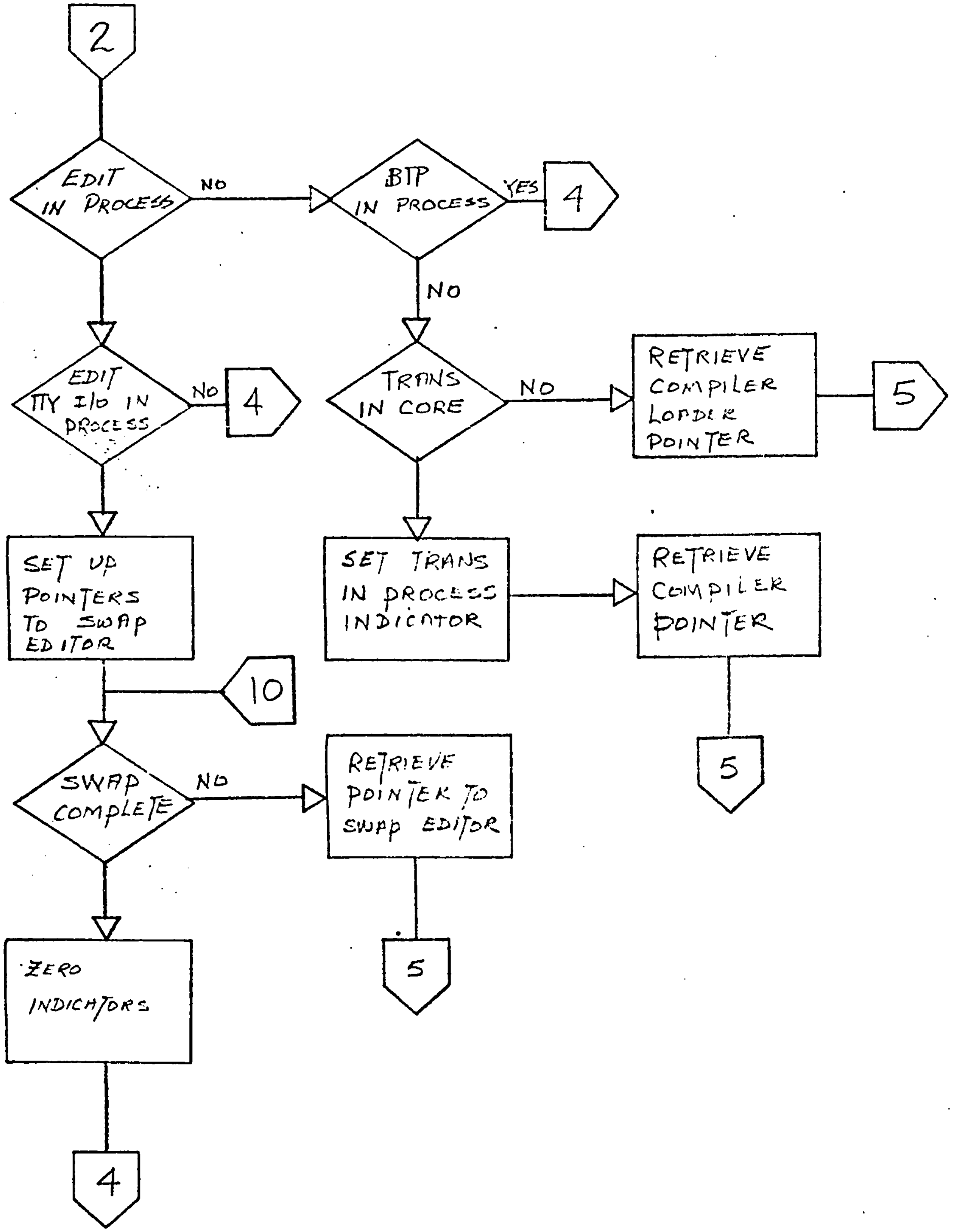
3.3 - SYSTEM USER PROGRAM LOADERS

After the Monitor is loaded, control goes to the cold

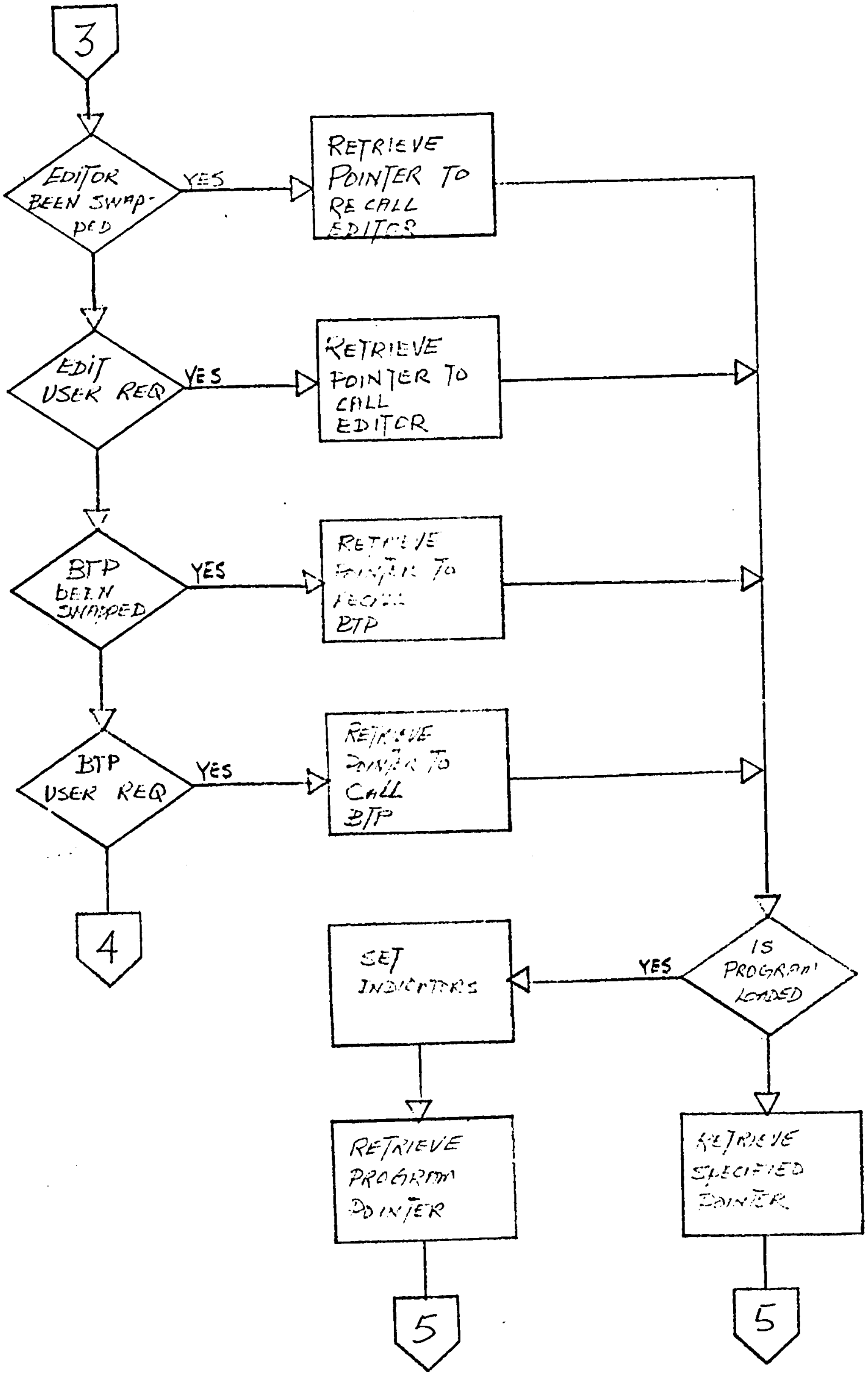


SCHEDULER FLOW CHART 3.6


4
7
IS API LEVEL 6up
YES
EXIT POINTER = φ
YES
RETRIEVE CAL LOOP POINTER
IS OP/SYS IN CORE
NO
11
DEBREAK FROM API4
11
OP/SYS USER REQ
NO
7
7
YES
STARTED LOADING OP/SYS
RETRIEVE USER DATA FROM XOP6FQ
ABORT THIS USER
SET START INDICATER
IRR. ERROR PRINT
NO
IS USER ACTIVE
9
RETRIEVE POINTER TO OP/SYS LOADER
YES
RETRIEVE USER POINTER
WAS ? TYPED
YES
STORE USER POINTER IN XSEMIR TBL
9
REQUEST SERVICE AT API LEVEL 6
7
9
RETRIEVE ; HANDLER POINTER


2
EDIT IN PROCESS
NO
BTP IN PROCESS
YES
4
NO
EDIT TTY I/O IN PROCESS
NO
4
TRANS IN CORE
NO
RETRIEVE COMPILER LOADER POINTER
5
SET UP POINTERS TO SWAP EDITOR
SET TRANS IN PROCESS INDICATOR
RETRIEVE COMPILER POINTER
10
5
SWAP COMPLETE
NO
RETRIEVE POINTER TO SWAP EDITOR
5
ZERO INDICATORS
4

SCHEDULER FLOW CHART 3.6 PAGE 3 of 4


3
EDITOR BEEN SWAPPED
YES
RETRIEVE POINTER TO RECALL EDITOR
EDIT USER REQ
YES
RETRIEVE POINTER TO CALL EDITOR
BTP BEEN SWAPPED
YES
RETRIEVE POINTER TO RECALL BTP
BTP USER REQ
YES
RETRIEVE POINTER TO CALL BTP
4
IS PROGRAM LOADED
YES
SET INDICATORS
RETRIEVE PROGRAM POINTER
RETRIEVE SPECIFIED POINTER
5
5

SCHEDULER FLOW CHART 3.6 PAGE 4 of 4

start program which clears all flags and enables API. A call loop is established, which repeatedly calls the Monitor until a user request is registered. When the first call is made, the Scheduler initiates the loading of the Op/Sys. The Loader consists of two dectape handler calls for opening the Op/Sys file and for reading that file into core. The Op/Sys is then resident in core.

When a request is registered for Op/Sys usage, control is transferred to the Op/Sys program. When a background user request is registered, the Scheduler initiates the loading of the required program. Upon entry into the loader, the compiler in process indicator is set so that the loading will be completed before allowing interruptions. A file is opened requesting the background program. Subsequent I/O calls are made to read the file into core. When complete, indicators are set to specify what program is in process and the loader is terminated via the END CAL. The swap loader writes and reads programson predetermined mass storage locations, one for the BTP and one for the Editor.

3.4 - MONITOR ERRORS

If an error condition arises because of software bugs in the Monitor or I/O Handlers, a Monitor Irrecoverable error printout is generated on the System TTY and processing is Halted.

The format of the messages is "ERRN" when N is a number from 0 to 50. The definition of the error appears in Appendix 5.

When an error is detected, a Jumpstore instruction is executed with the reference number N. The Error Handler disables API, converts the number N into the appropriate ASCII characters, and outputs the message on the System teletype. When the print is complete, a HLT[5] instruction is executed.

CHAPTER 4

I/O HANDLERS

A typical call for an I/O operation is:

LAC[5] Arg 1
XOR[5] Arg 2
XOR Arg 3
DAC[5] .+1
CAL[5]
Arg 4

Arg 1 - specifies the System Program; either the Op/Sys, the Compiler, the Editor or the BTP.

Arg 2 - specifies the Handler number.

Arg 3 - specifies the FACTS station number.

Arg 4 - specifies the Buffer location where data is to be transferred.

Arguments 1, 2, and 3 are combined with the CAL instruction prior to the CAL being executed.

For single threaded handlers, i.e. dectape and paper tape handlers, the CAL data, buffer pointer, and the user return pointer are accessed from the "XGENQ" queue and stored in the handler via the "XGETUQ" sub-routine. In multi-threaded handlers, i.e. the TTY and Real Time Handlers, the data is stored in the handler or referenced directly from the I/O initiator by the handler at the time of the I/O call.

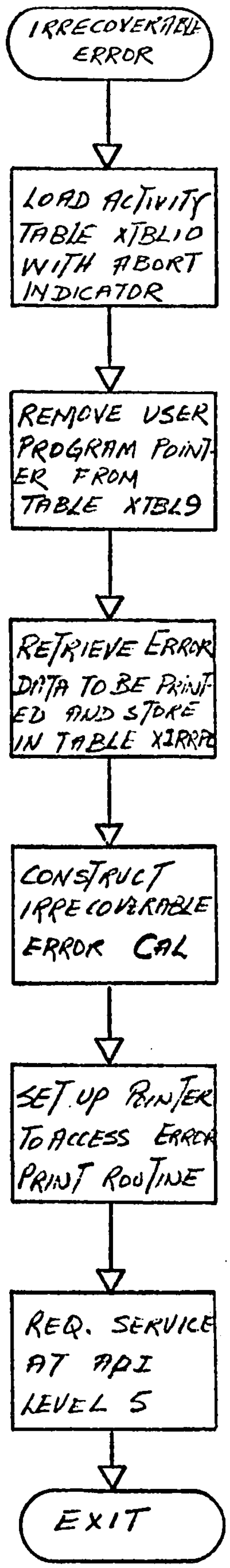
4.1 - ERROR HANDLING

There are two types of Error processing in Hardware Handlers: Recoverable Errors, which allow for operator interaction for recovery, and Irrecoverable Errors which aborts the user. The hardware handler interrogates flags and if a recoverable error is detected, a request for recoverable error print is made via API Level 5 handler. If an irrecoverable error is detected, the user active indicator in "XTBL10" table is changed to an abort request, which is interrogated by the Scheduler, and the user system program identifier in "XTBL9" table is cleared. The error data to be typed is stored in "XIRRPO" table (in Octal) with reference to the FACTS station number. A request for an error print via API level 5 handler is subsequently made. This handler completes the print request via the I/O terminator.

When a recoverable error option is included in a handler, a limited number of recoveries are possible, after which, if not successful, an irrecoverable error condition will result. Refer to Flow Chart 4.1.

4.2 - TTY HANDLERS

The TTY handlers can service eight separate TTY's. The TTY's are interfaced to the PDP-9 computer with separate hardware buffers tied to the I/O bus with all flags ORED to one position of API level 3. When an API interrupt occurs, causing a jump store instruction via trap address 71 to the TTY handlers, a common skip chain interrogates which flag, Keyboard, or printer caused the interrupt. Register "XTTYNO" is set


IRRECOVERABLE ERROR
LOAD ACTIVITY TABLE XTBL10 WITH ABORT INDICATOR
REMOVE USER PROGRAM POINTER FROM TABLE XTBL9
RETRIEVE ERROR DATA TO BE PRINTED AND STORE IN TABLE XIRRPC
CONSTRUCT IRRECOVERABLE ERROR CAL
SET UP PRINTER TO ACCESS ERROR PRINT ROUTINE
REQ. SERVICE AT API LEVEL 5
EXIT

IRRECOVERABLE ERROR HANDLING FLOW CHART 4.1

indicating the FACTS station number, according to the position in the skip chain where the flag was on. If a print flag comes up, the Print Handler is entered, if a Keyboard flag comes up, the TTY buffer is read and the Keyboard Handler is entred. The TTY's are connected in a Full Duplex mode. When a character is typed on the Keyboard, the character is echoed via the Print Handler after processing. Appendix 2 gives the trimmed ASCII character definitions used.

4.2.1 - Keyboard Handler

The Keyboard Handler processes all characters that are typed on the TTY Keyboards. There are three modes under which this Handler accepts typed data. The User Initialization, COMMAND or TEXT MODE.

The User Initialization mode is defined when a user has not been identified to the Keyboard Handler. This mode can be entered only by typing one of the six legal Control Characters.

The COMMAND mode is defined to the Keyboard handler by a Monitor CALL request initiated by a System User Program. This mode permits up to an eight character command to be typed by the operator and at it's completion, the first three characters are transferred to the user program.

The TEXT mode is defined to the Keyboard Handler by a Monitor CALL request initiated by a System User Program. An unlimited string of characters may be typed by the operator.

When a user has not initialized a System Program request , "WHAT?" will be echoed on the printer, if a character other than the legal control characters is typed. If a System Program has been initialized, but the COMMAND or TEXT mode has not been defined to the

Keyboard Handler, any character typed on the Keyboard is ignored.

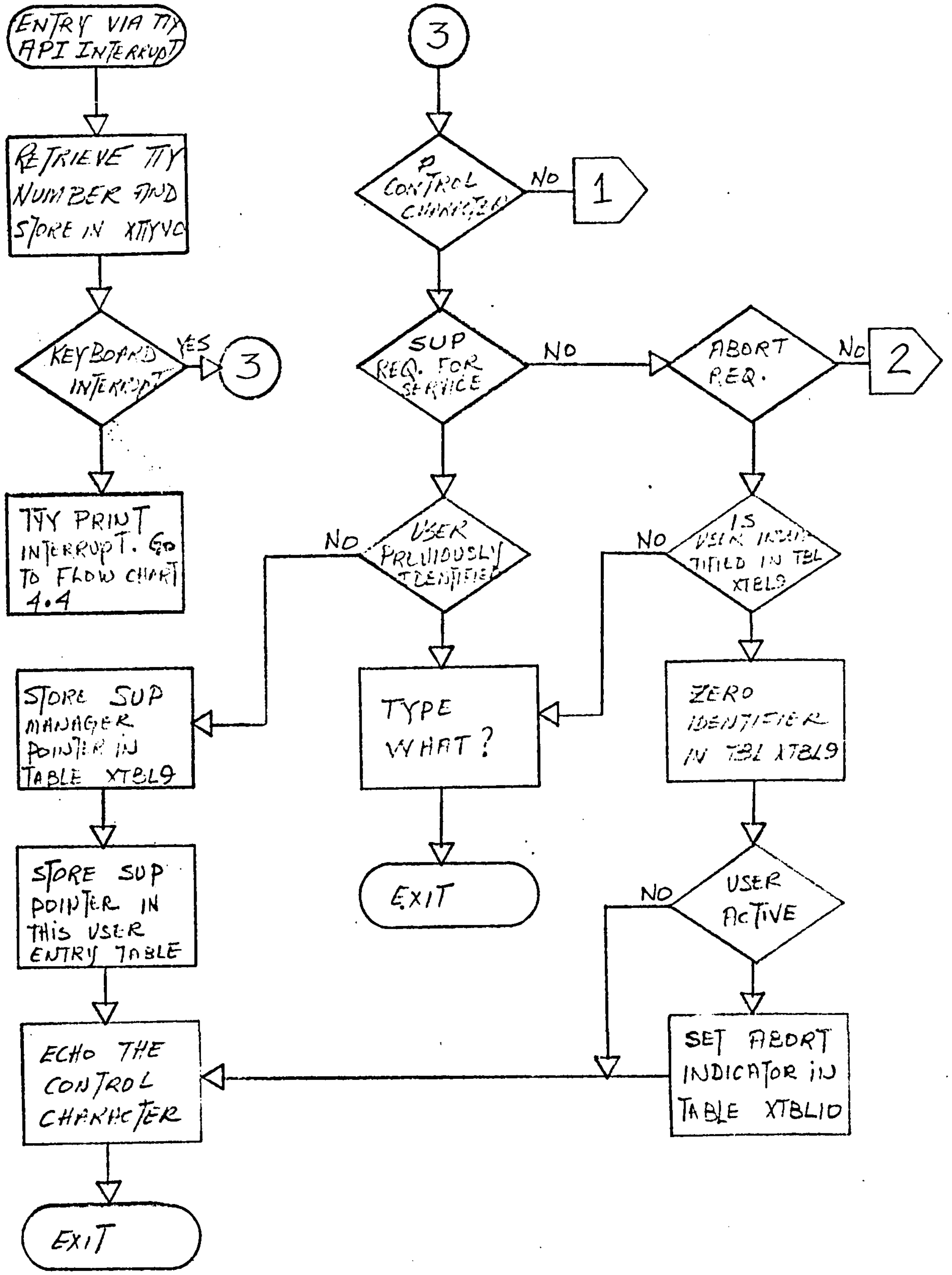
4.2.1.1 - User Initialization Mode

There are 6 Monitor Commands that are defined in the Keyboard Handler. ↑F, ↑T, ↑E, and ↑B are System Program call requests for the Op/Sys, the Compiler, the Editor, and the BTP, respectively. In additon, there is ↑A for aborting the present user and ↑R for giving control back to the I/O Handler subsequential to a recoverable error print and operator interaction. Carriage return (CR) is used for delimiting a command string, and rubout deletes the data being typed with a backslash being echoed. When requesting service, the user types one of the four System Program calls. The Keyboard Handler detects that it is a legal control character; that this FACTS Test Station was previously inactive, and subsequently stores a pointer to the Keyboard service routine for the System Program in "XTBL9" table. The user entry table is loaded with the starting address of the respective System Program. The Echo routine is entered, which sets up the proper links and accesses the Print Handler to echo the control character typed on the Keyboard. The Handler is then primed to receive the File name that the System Program will operate on.

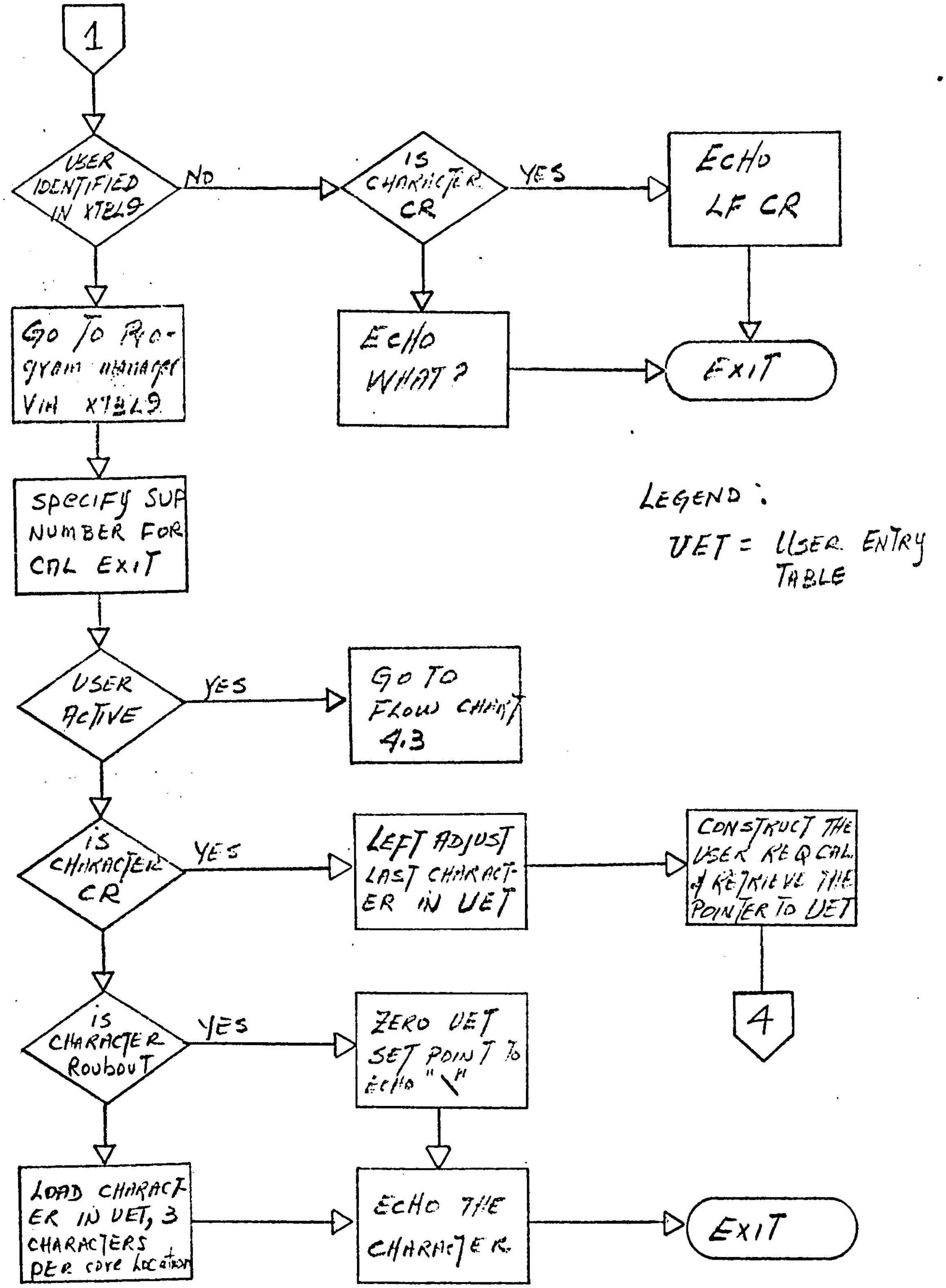
The file name characters are typed and read into the Keyboard Handler. They are checked for validity and are converted if necessary. The characters are packed in 6 bit trimmed ASCII (refer to Appendix 2) and stored in the user entry tables, "XTBL30" to "XTBL37", as a function of the FACTS STATION NUMBER. A maximum of eight characters are accepted. Each character typed is echoed. Subsequently, the user types a carriage return which delimits the character string and initiates a request for service via the API level 5 call to the Monitor. Since a number of requests for service can be made simultaneously, the status register "XAPI5C" is updated to indicate which FACTS STATIONS have requested service and the CALL data is stored in table "XTBL11" as a function of the FACTS number. The CALL DATA contains a CAL^5 with the user request handler number and the FACTS STATION number followed by a pointer to the user entry table. The activity table "XTBL10" is conditioned to indicate that this user is now active. If a character is typed before the legal control character, or if more than eight characters are typed, "WHAT?" is echoed and the user entry table is cleared. Flow Chart 4.2 shows the processing of a Keyboard request for service.

When a user wishes to terminate processing of a System Program a ↑A is typed. The active indication in table "XTBL10" is changed to an abort request, to be completed by the Scheduler, and the System Program indicator in table "XTBL9" is cleared. ↑A is echoed on the printer. If the abort processing is not complete, and a ↑A is typed, "WHAT?" is echoed. If ↑A is typed at an inactive FACTS STATION, "WHAT?" is echoed.

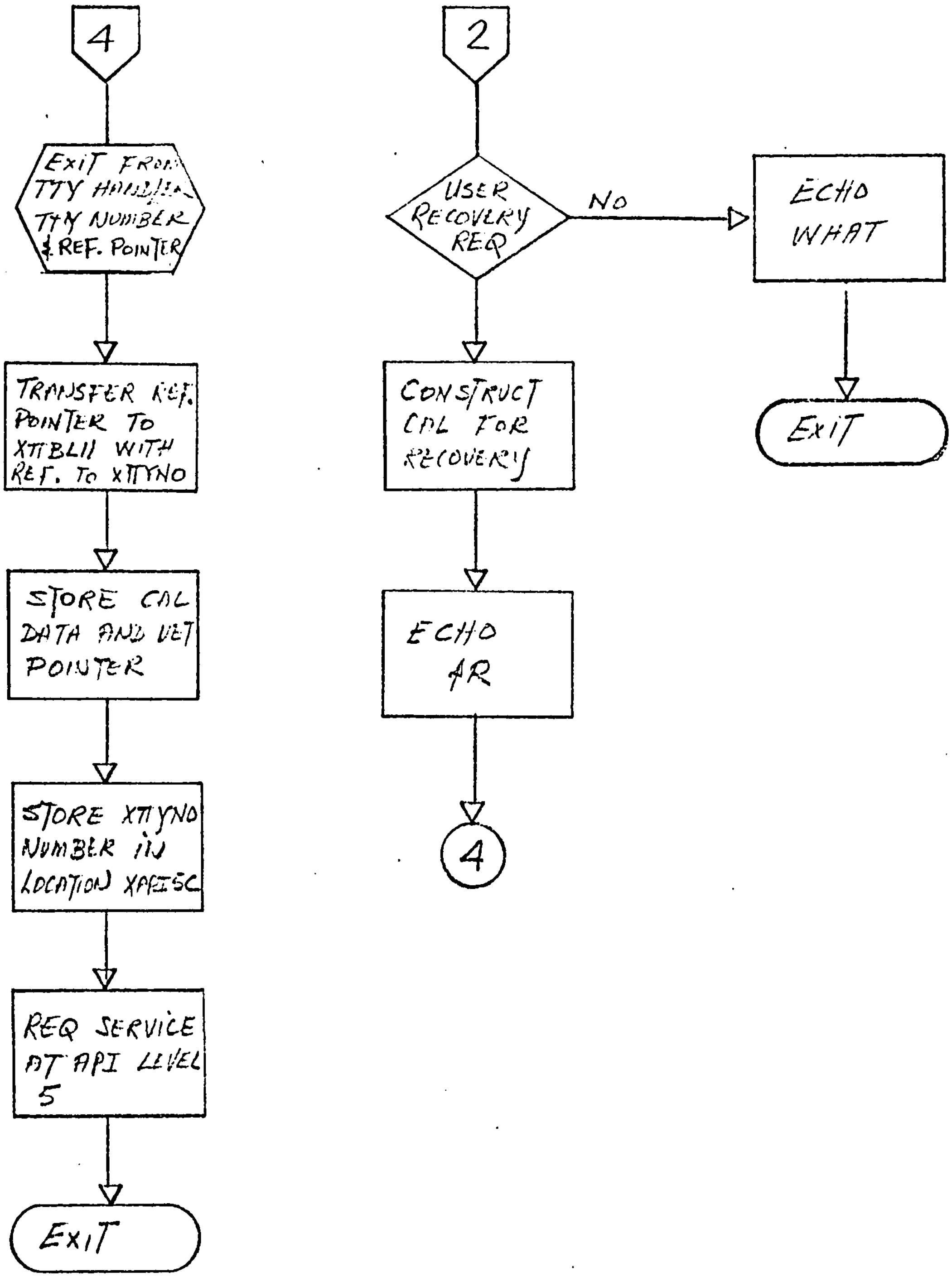
When a recoverable error printout is recorded, the operator performs the required action. At it's completion, an ↑R is typed to transfer control back to the Hardware Handler. A CALL request is constructed with the respective FACTS number and the recoverable request handler number, and this is stored in the "XTBL11" table. The status register "XAPI5C" is updated to indicate which user is requesting a CALL to the Monitor via API level 5. The I/O Terminator will complete the entry back to the handler that originated the recoverable error printout.


ENTRY VIA TTY API INTERRUPT
RETRIEVE TTY NUMBER AND STORE IN XTTYVC
KEYBOARD INTERRUPT
YES
3
TTY PRINT INTERRUPT. GO TO FLOW CHART 4.4
3
P CONTROL CHARACTER
NO
1
SUP REQ. FOR SERVICE
NO
ABORT REQ.
NO
2
USER PREVIOUSLY IDENTIFIED
NO
IS USER IDENTIFIED IN TBL XTBL9
NO
STORE SUP MANAGER POINTER IN TABLE XTBL9
TYPE WHAT?
ZERO IDENTIFIER IN TBL XTBL9
STORE SUP POINTER IN THIS USER ENTRY TABLE
EXIT
USER ACTIVE
NO
ECHO THE CONTROL CHARACTER
SET ABORT INDICATOR IN TABLE XTBL10
EXIT

TTY KEYBOARD HANDLER FLOW CHART 4.2 PAGE 1 of 3


1
USER IDENTIFIED IN XT2L9
NO
IS CHARACTER CR
YES
ECHO LF CR
GO TO PRO-gram manager VIA XT2L9
ECHO WHAT?
EXIT
SPECIFY SUP NUMBER FOR CAL EXIT
LEGEND:
UET = USER ENTRY TABLE
USER ACTIVE
YES
GO TO FLOW CHART 4.3
IS CHARACTER CR
YES
LEFT ADJUST LAST CHARACT-ER IN UET
CONSTRUCT THE USER REQ CAL & RETRIEVE THE POINTER TO UET
4
IS CHARACTER ROUBOUT
YES
ZERO UET SET POINT TO ECHO "\"
LOAD CHARACT-ER IN UET, 3 CHARACTERS PER core location
ECHO THE CHARACTER.
EXIT

TTY KEYBOARD HANDLER　　FLOW CHART 4.2　　PAGE 2 of 3


4
EXIT FROM TTY HANDLER TTY NUMBER & REF. POINTER
TRANSFER REF. POINTER TO XTTBLII WITH REF. TO XTTYNO
STORE CAL DATA AND VET POINTER
STORE XTTYNO NUMBER IN LOCATION XPFISC
REQ SERVICE AT API LEVEL 5
EXIT
2
USER RECOVERY REQ
NO
ECHO WHAT
EXIT
CONSTRUCT CAL FOR RECOVERY
ECHO ↑R
4

TTY KEYBOARD HANDIER FLOW CHART 4.2 PAGE 3 of 3

4.2.1.2 - Command and Text Modes

When a User Program initiates an I/O request for Keyboard input, the I/O initiator loads the user return pointer in table "XTBL2" and the user buffer pointer and mode data in table "XTBL3", as a function of the user FACTS number. A " >" is typed to indicate to the operator that interaction is requested.

There are two modes defined, the Command and Text input data request. If the Keyboard is struck prior to a program user request, the character is ignored. In the Command mode, the Keyboard handler reads and echoes one to eight characters typed and stores them in the user entry table in 6 bit trimmed ASCII. When a carriage return is typed, the first one to three characters of the Command are transferred to the user buffer location as specified in table "XTBL3". The user return pointer is accessed from table "XTBL2", a CAL is constructed and both are stored in "XTBL11." The "XAPI5C" status word is updated and a request is made for API level 5 service to initialize return to user control

In the Text mode, the Keyboard Handler reads and echoes the data being typed, and stores it in the user buffer, as specified in "XTBL3", in 6 bit trimmed ASCII. It is assumed that the buffer size is 255 (decimal) locations. When a carriage return is typed, a calculation is made so that a minimum of 24 (decimal) locations remain in the buffer prior to accepting another line of Text, so that the buffer will have only complete lines of Text. When there are less than 24 locations left, a "*" is typed and control is transferred back to the user via API level 5. After the User Program has processed this buffer, a Text I/O call will be made.

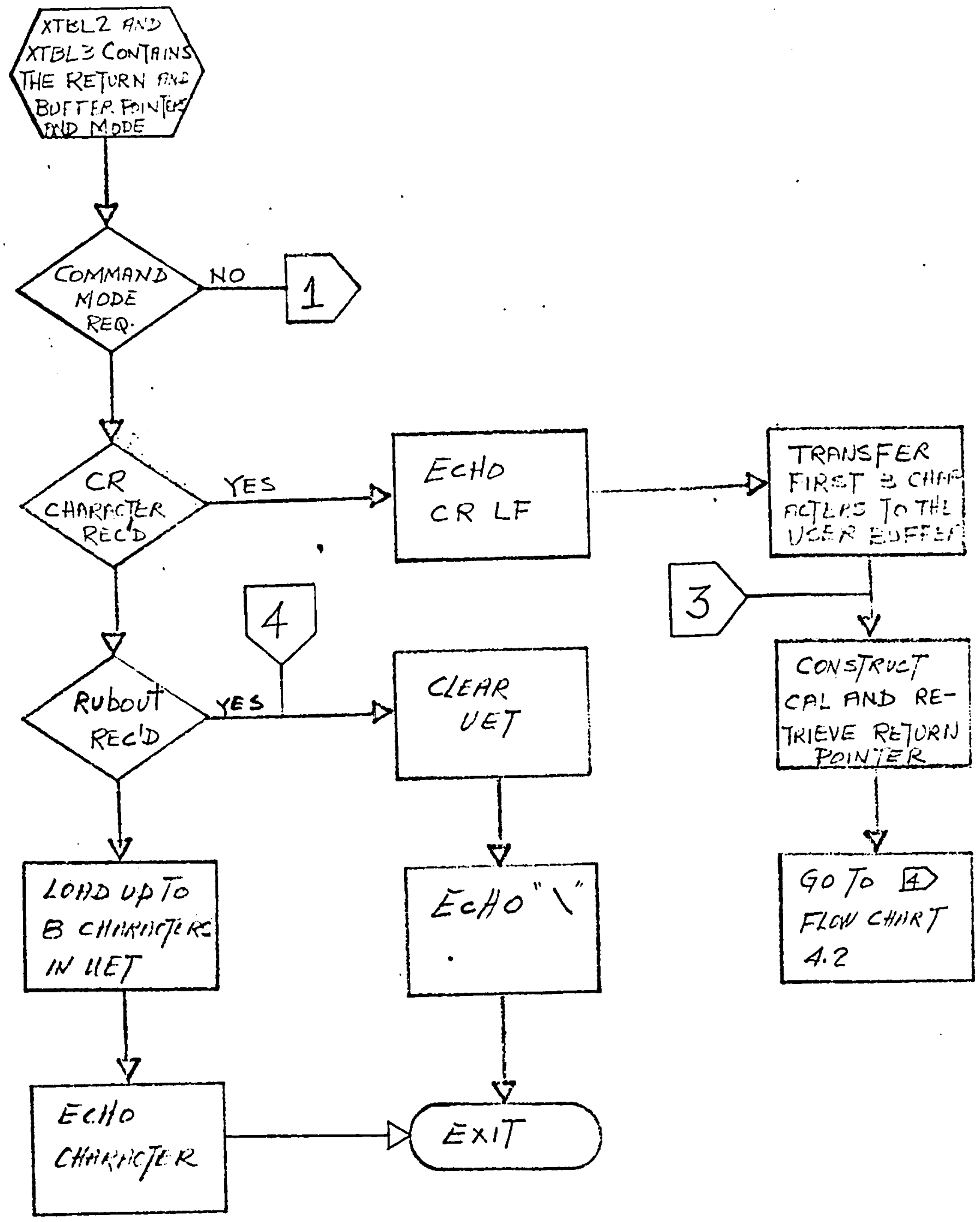
If the user is typing less than a buffer size by typing "*", following the Carriage Return, control will be returned to the User Program. If the user wishes to return to the Command mode, "CR*" is typed following the CR line delimiter. The User Program after receiving control, completes the processing of the Text and returns with the Command mode enabled. Flow chart 4.3 shows the processing of User Program request for operator input, via the Keyboards.

The ECHO sub-routine is entered with a number in the AC indicating what characters in "XTYPBK" table are to be echoed. Table "XTBL6" is loaded with this pointer, and a mode bit indicating an Echo request. Table "XTBL5" is loaded with the return pointer of the user requesting the Echo. The Print Handler is subsequently entered.

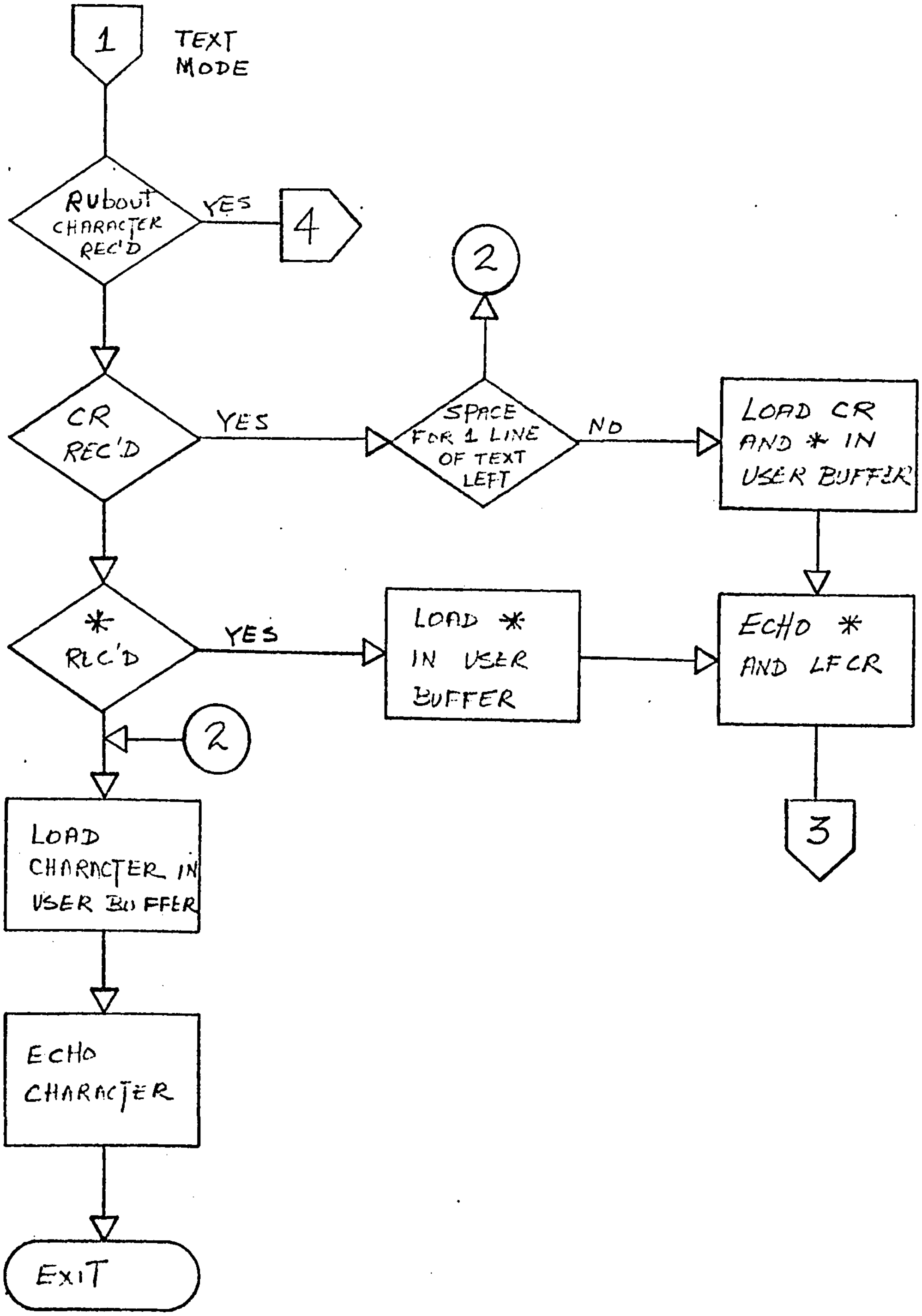
4.2.2 - Print Handler

When a user initiates an I/O request for the Print Handler, the I/O initiator loads the user return pointer in table "XTBL5", and the user buffer location and mode data in table "XTBL6". The I/O initiator conditions the Printer by issuing a non-printing character via the I/O instruction reference from table "XTBL4", as a function the FACTS station number. The printer flag is subsequently set and the printing of the data begins.

There are three modes defined in the Print Handler, 6 bit packed ASCII, unpacked 8 bit ASCII, and Octal print. The delimiter for 6 bit packed ASCII is a zero character, for unpacked 8 bit ASCII and octal printing the delimiter is a zeroed register. Upon entry into the Print Handler, via the print flag, the mode data is referenced


XTBL2 AND XTBL3 CONTAINS THE RETURN AND BUFFER POINTERS AND MODE
COMMAND MODE REQ.
NO
1
CR CHARACTER REC'D
YES
ECHO CR LF
TRANSFER FIRST 8 CHARACTERS TO THE USER BUFFER
3
CONSTRUCT CAL AND RETRIEVE RETURN POINTER
GO TO 4 FLOW CHART 4.2
4
RUBOUT REC'D
YES
CLEAR UET
ECHO "\"
LOAD UP TO 8 CHARACTERS IN UET
ECHO CHARACTER
EXIT

KEYBOARD COMMAND AND TEXT PROCESSING FLOW CHART 4.3 PAGE 1 of 2



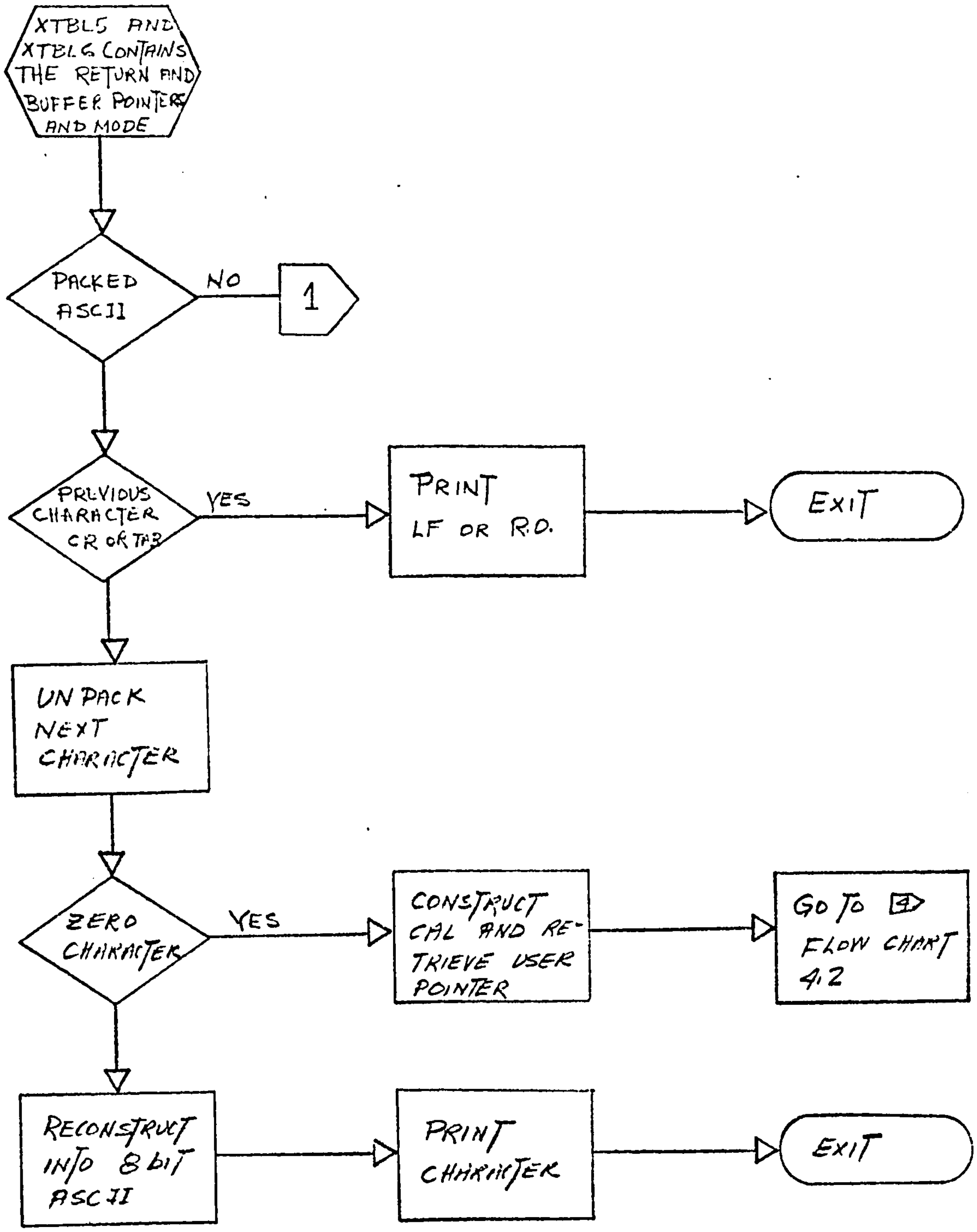
KEYBOARD COMMAND AND TEXT PROCESSING FLOW CHART 4.3 PAGE 2 of 2 and one of the 3 print mode handlers is entered.

For the 6 bit packed ASCII mode, if a "CR" or "TAB" was previously typed, an "LF" or "RO" is typed unconditionally, as referred to via table "XTBL50". If this is not the case, the next character is unpacked with reference to the counters referred to in table "XTBL40", and the user buffer location in table "XTBL6". The character is checked for possible conversion. If the character is a "TAB" or a "CR", table "XTBL50" is updated to subsequently print "RO" or "LF" respectively. The character is reconstructed into an 8 bit format and printed with reference to the I/O instruction in table "XTBL4". The counter in table "XTBL40" is updated as characters are unpacked, and the user buffer location is modified as data is being printed. When a zero character is encountered, control returns back to the user program via an API level 5 request.

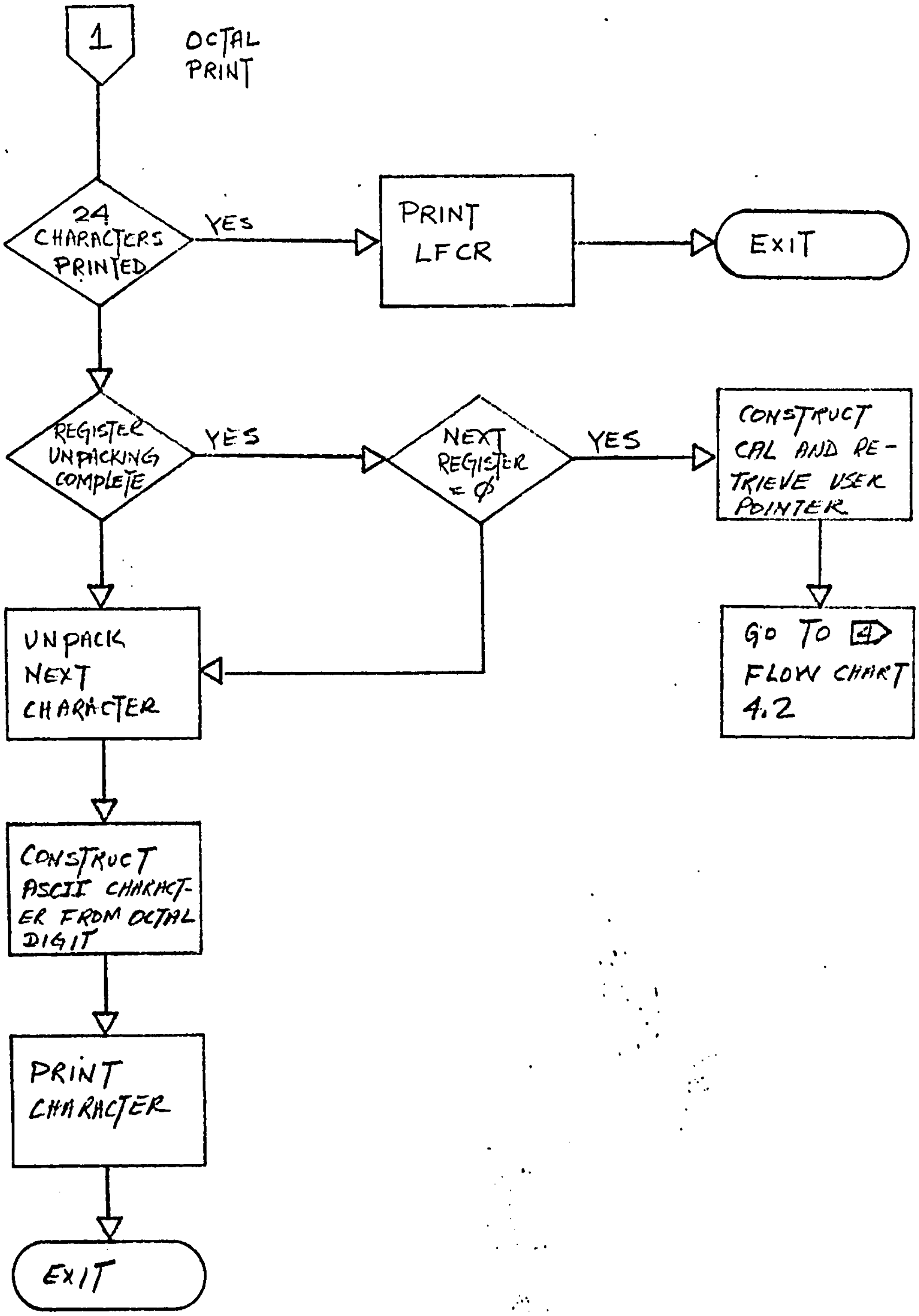
Octal and 8 bit ASCII printing is processed in the same manner. Table "XTBL41" contains the counters to unpack 6 Octal digits per register. When a zero register is encountered, control reverts back to the User Program via an API level 5 request. Refer to Flow Chart 4.4.

4.3 - PAPER TAPE READER HANDLER

This handler is single threaded. The user CALL data is referred to via the "XGENQ" queue and stored in the handler. The input data via paper tape is assumed to be either standard 8 bit ASCII or ASCII with bit 8 used for polarity. The Paper Reader hardware is set in motion, in the alphanumeric mode, and the handler exits to the Scheduler. This processing is done at API level 4.


XTBL5 AND XTBL6 CONTAINS THE RETURN AND BUFFER POINTERS AND MODE
PACKED ASCII
NO
1
PREVIOUS CHARACTER CR OR TAB
YES
PRINT LF OR R.O.
EXIT
UN PACK NEXT CHARACTER
ZERO CHARACTER
YES
CONSTRUCT CAL AND RE-TRIEVE USER POINTER
GO TO 4 FLOW CHART 4.2
RECONSTRUCT INTO 8 BIT ASCII
PRINT CHARACTER
EXIT

PRINT HANDLER FLOW CHART 4.4 PAGE 1 of 2

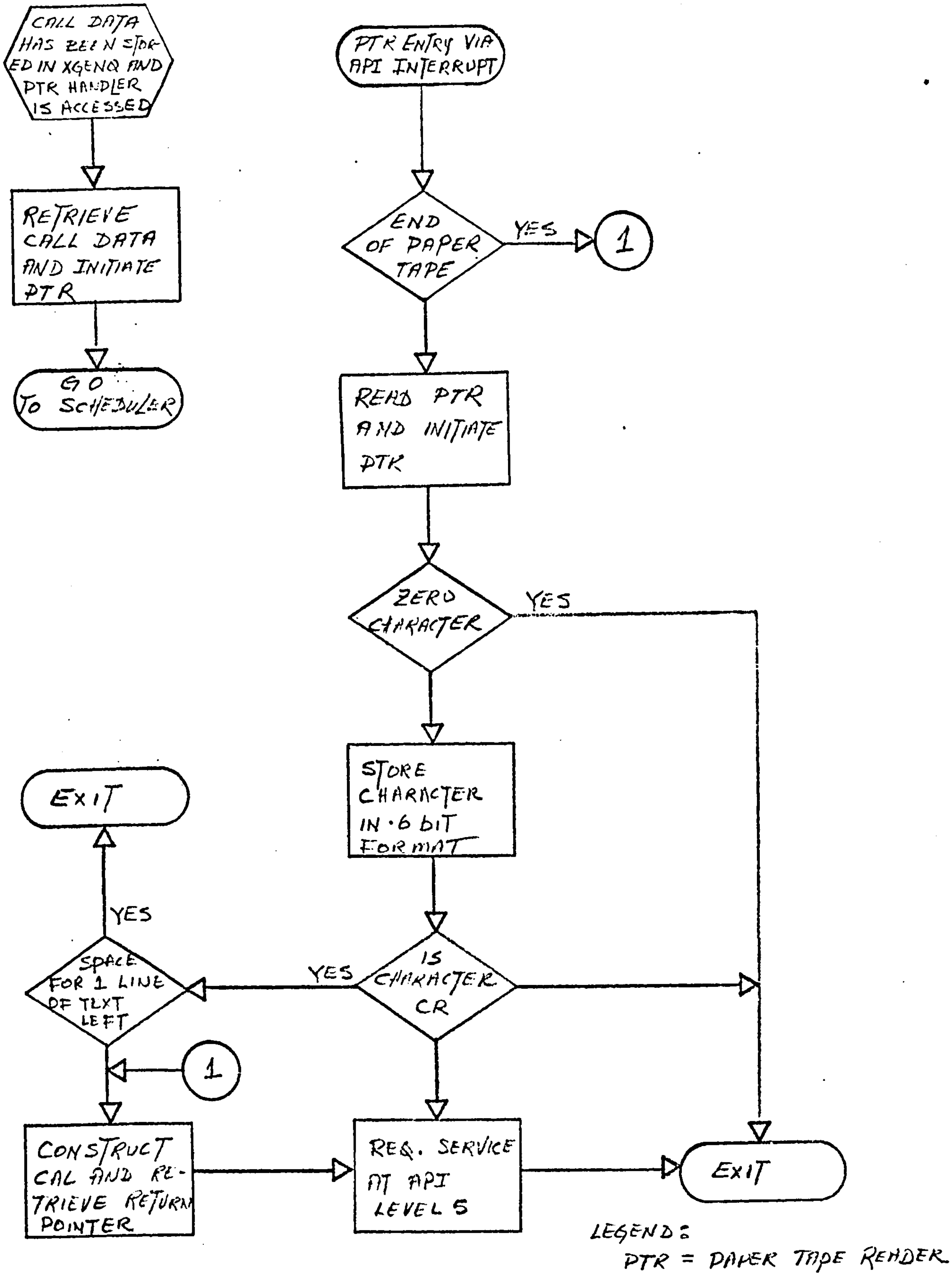

1
OCTAL PRINT
24 CHARACTERS PRINTED
YES
PRINT LFCR
EXIT
REGISTER UNPACKING COMPLETE
YES
NEXT REGISTER = Ø
YES
CONSTRUCT CAL AND RE-TRIEVE USER POINTER
GO TO 4 FLOW CHART 4.2
UNPACK NEXT CHARACTER
CONSTRUCT ASCII CHARACT-ER FROM OCTAL DIGIT
PRINT CHARACTER
EXIT

When a character is detected, the Reader flag is set which causes an API level 2 request. The handler is entered via the forced Jump Store Instruction in the trap location 50. The paper tape end flag is interrogated (this flag is used as the delimiter for the data being read) and if active control is transferred to the user. If this flag is not "up" the character is read substituted if required and subsequently packed and stored in the user buffer. When a "CR" is read, a calculation is performed so that a complete line of data will terminate the buffer, which is assumed to have 255 (decimal) locations. If the buffer gets filled prior to the paper tape completion, control is transferred to the user to process the data, and subsequently the paper tape handler will be recalled to complete the paper tape read. Refer to Flow Chart 4.5.

4.4 - PAPER TAPE PUNCH HANDLER

The Paper Punch Handler is single threaded. The CALL data is referred to from the "XGENQ" queue and stored in the handler. The data buffer is assumed to be 255 locations of 6 bit pack ASCII. The Paper Punch is initialized and exits to the Scheduler. This processing is done at API level 4.

When the Punch gets up to speed, a flag is set which cause an API level 3 interrupt, and forces a Jump Store execution via the trapped address. The status of the paper tape is interrogated, and if insufficient paper is present, a recoverable error request is made via API level 5. If this condition persists after two recoverable error printouts. to the operator, an irrecoverable request is made, and the user is aborted.


CALL DATA HAS BEEN STORED IN XGENQ AND PTR HANDLER IS ACCESSED
RETRIEVE CALL DATA AND INITIATE PTR
GO TO SCHEDULER
PTR ENTRY VIA API INTERRUPT
END OF PAPER TAPE
YES
1
READ PTR AND INITIATE PTR
ZERO CHARACTER
YES
STORE CHARACTER IN 6 BIT FORMAT
EXIT
YES
SPACE FOR 1 LINE OF TEXT LEFT
YES
IS CHARACTER CR
1
CONSTRUCT CAL AND RETRIEVE RETURN POINTER
REQ. SERVICE AT API LEVEL 5
EXIT
LEGEND:
PTR = PAPER TAPE READER

PAPER TAPE READER HANDLER

FLOW CHART 4.5

If there is sufficient paper tape the first character is accessed via the user buffer pointer and it is unpacked, substituted as required and recontructed into 8 bit ASCII and punched. When a zero 6 bit ASCII character is detected, control reverts to the User Program. Refer to Flow Chart 4.6.

4.5 - DECTAPE[9] HANDLER

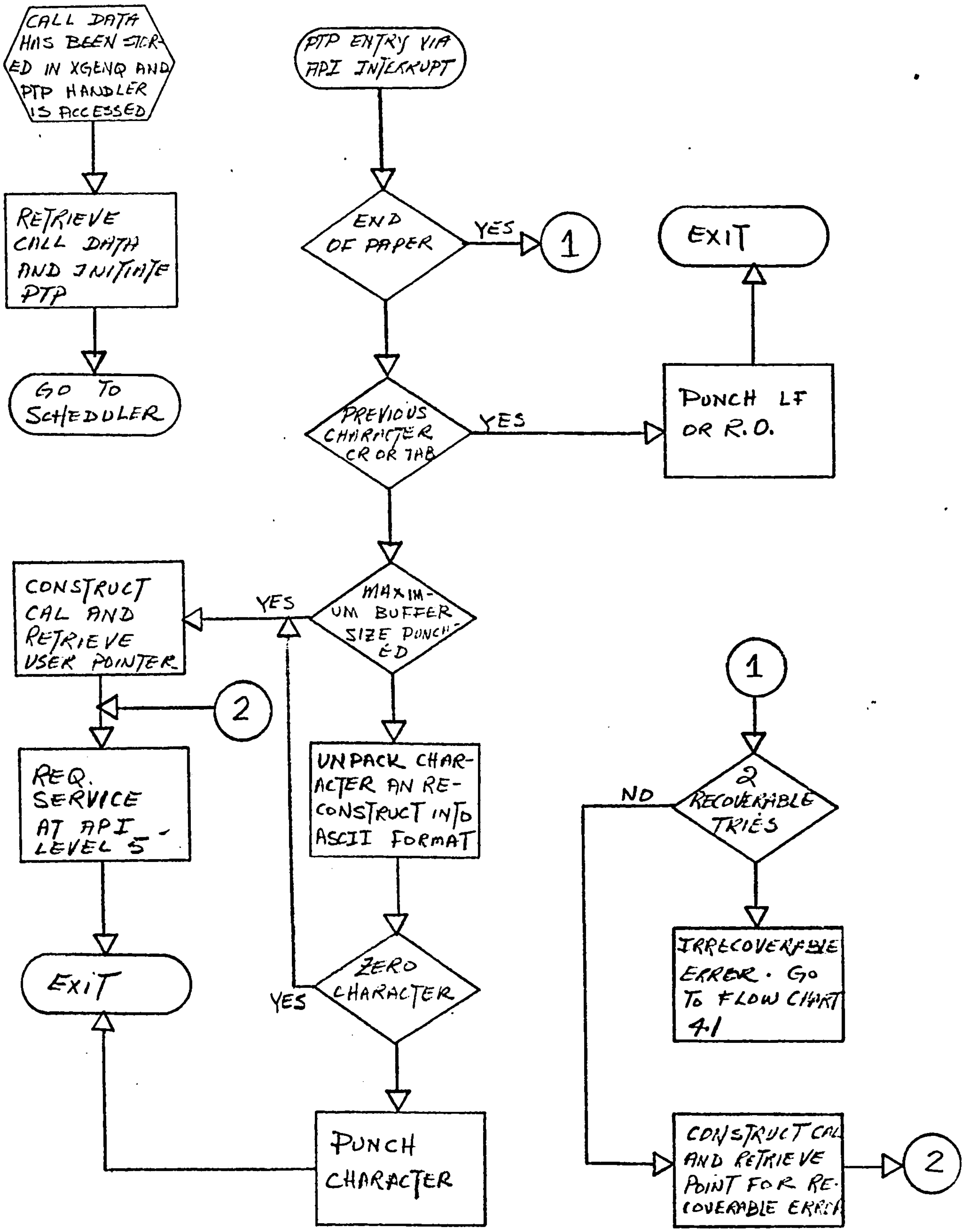
This Handler is single threaded. Data is referred to from the "XGENQ" queue and stored in the handler. The FACTS station number also defines the dectape that will be active. The format of the user buffer for the dectape I/O CALL is:

```
Pointer 1,          NAM          /Name of the file
                    E00          /In 6 bit ASCII
                    000          /8 characters maximum
Pointer 2,                       /# of blocks and starting
                                  block
                                 /Present block
                                 /First core location of user
                                  buffer.
```

There are 6 operations defined in the dectape handler: open a file, close a file, read one block of data, 256 locations, read all the file, write one block of data, and write all the file. Appendix 3 shows the definition of the dectape directory.

4.5.1 - Open a Dectape File

To open a file, whether for reading or writing, the user spe-


CALL DATA HAS BEEN STORED IN XGENQ AND PTP HANDLER IS ACCESSED
RETRIEVE CALL DATA AND INITIATE PTP
GO TO SCHEDULER
PTP ENTRY VIA API INTERRUPT
END OF PAPER
YES
1
EXIT
PREVIOUS CHARACTER CR OR TAB
YES
PUNCH LF OR R.O.
CONSTRUCT CAL AND RETRIEVE USER POINTER
YES
MAXIMUM BUFFER SIZE PUNCHED
1
2
REQ. SERVICE AT API LEVEL 5
UNPACK CHARACTER AN RECONSTRUCT INTO ASCII FORMAT
NO
2 RECOVERABLE TRIES
EXIT
ZERO CHARACTER
YES
IRRECOVERABLE ERROR. GO TO FLOW CHART 41
PUNCH CHARACTER
CONSTRUCT CAL AND RETRIEVE POINT FOR RECOVERABLE ERR
2

LEGEND :
PTP = PAPER TAPE PUNCH

PAPER TAPE PUNCH HANDLER

FLOW CHART 4.6

cifies Pointer 1, which has previously been loaded with the file name. The "find" routine is accessed which in turn accesses the dectape hardware handler to read the first directory block into the user buffer location as specified in location Pointer 2 + 3. The FIND routine masks for a match with the directory file entries and, if not found, recalls the dectape hardware handler for the next directory block. This process is repeated until the directory blocks and the directory file entries are exhausted, or a match is found. If a match is found, the number of blocks and the first block numbers are read from the directory, and stored at Pointer 2 location of the user buffer, and the next location is zeroed (the present block number). If a match is not found, the next free block is loaded in the user buffer location Pointer 2. Control is then transferred to the user via API level 5.

4.5.2 - Close a Dectape File

To close a file the user specifies Pointer 1 for the buffer address in the I/O CALL. If the file closing is by a compiler user, the first character of the file name is changed to a "T" to specify an "object" file. The FIND routine is accessed and the directory is searched. If a file with the same name is found, it is deleted. The FIND routine is accessed again until the next free entry in the directory is found. The file name, the number of blocks, and the first blockage accessed from the user buffer, and added to the directory block. The next free dectape block is added to the header of the directory block, and it is written onto dectape.

4.5.3 - Read from Dectape

To read a block of data from dectape the user specifies Pointer 2 in the I/O CALL. The number of blocks, the dectape number, and the present block number are accessed from the user buffer, and if all blocks of this file have been read, the present block in the user table is cleared, and control returns to the Scheduler. If all blocks have not been read, the hardware handler is entered to read the specified block into the core location specified in the user buffer. When complete, control is returned to the user via API level 5.

To read a complete file, the dectape hardware handler is accessed. The complete file is read into the user buffer table.

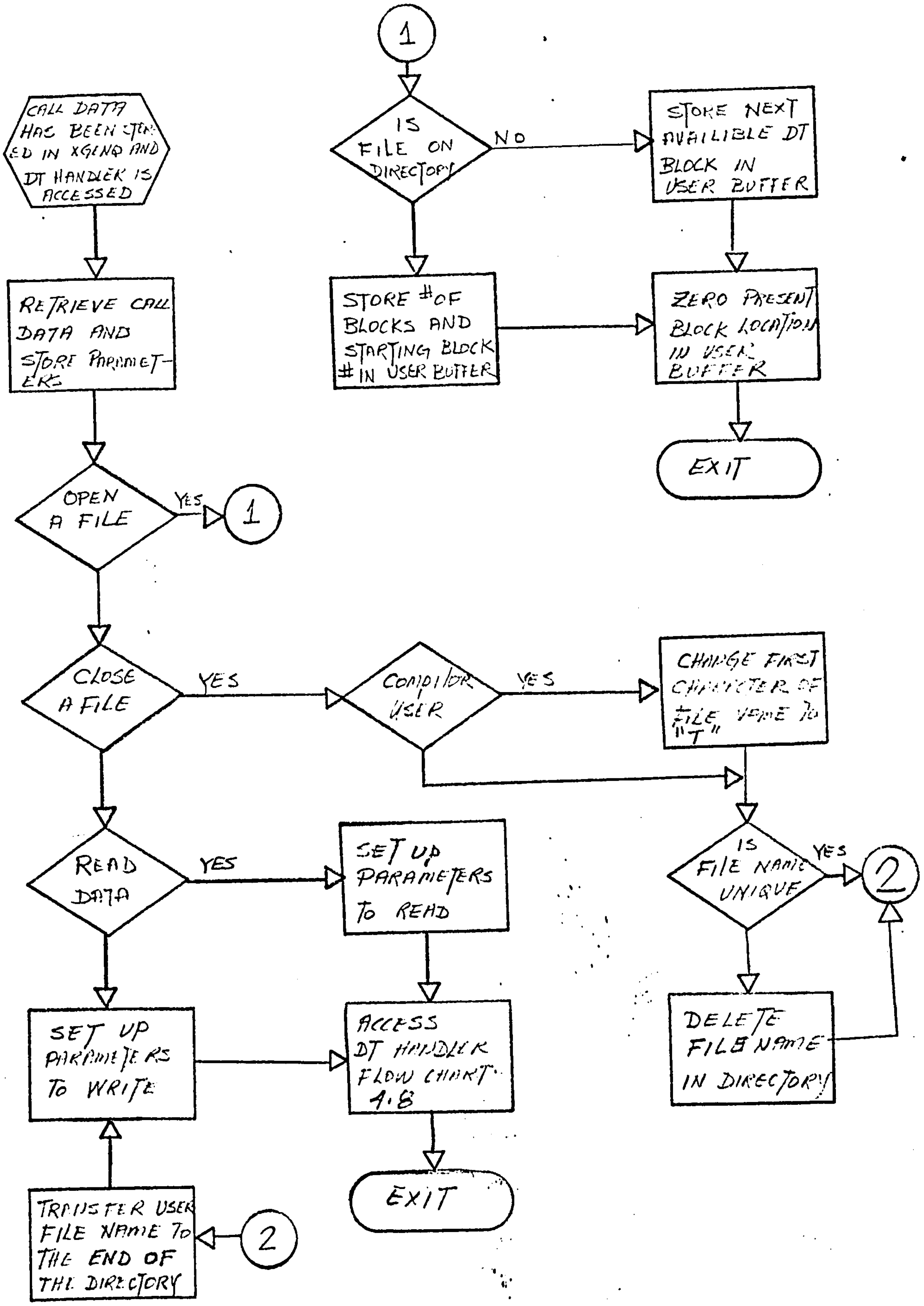
4.5.4 - Write onto Dectape

To write a block of data onto dectape, the first or next block number is accessed, the block number is incremented in the user table and the dectape hardware handler is accessed to write the user data on the specified block.

To write a complete file, the user must specify the number of blocks in bits locations 0 to 7 of Pointer 2 location in the user buffer table. The first block and the number of blocks is accessed from the user table and the complete file is written onto dectape. Refer Flow Chart 4.7.

4.5.5. - Dectape Hardware Handler

Upon entry into the Dectape Hardware Handler, the pointer to the first core location where data is to be transferred, the number of blocks to transfer, the first block number, the dectape number, and


1
CALL DATA HAS BEEN STORED IN XGENQ AND DT HANDLER IS ACCESSED
RETRIEVE CALL DATA AND STORE PARAMETERS
OPEN A FILE
YES
1
IS FILE ON DIRECTORY
NO
STORE NEXT AVAILIBLE DT BLOCK IN USER BUFFER
STORE # OF BLOCKS AND STARTING BLOCK # IN USER BUFFER
ZERO PRESENT BLOCK LOCATION IN USER BUFFER
EXIT
CLOSE A FILE
YES
COMPILOR USER
YES
CHANGE FIRST CHARACTER OF FILE NAME TO "T"
IS FILE NAME UNIQUE
YES
2
DELETE FILE NAME IN DIRECTORY
READ DATA
YES
SET UP PARAMETERS TO READ
SET UP PARAMETERS TO WRITE
ACCESS DT HANDLER FLOW CHART 4.8
EXIT
TRANSFER USER FILE NAME TO THE END OF THE DIRECTORY
2

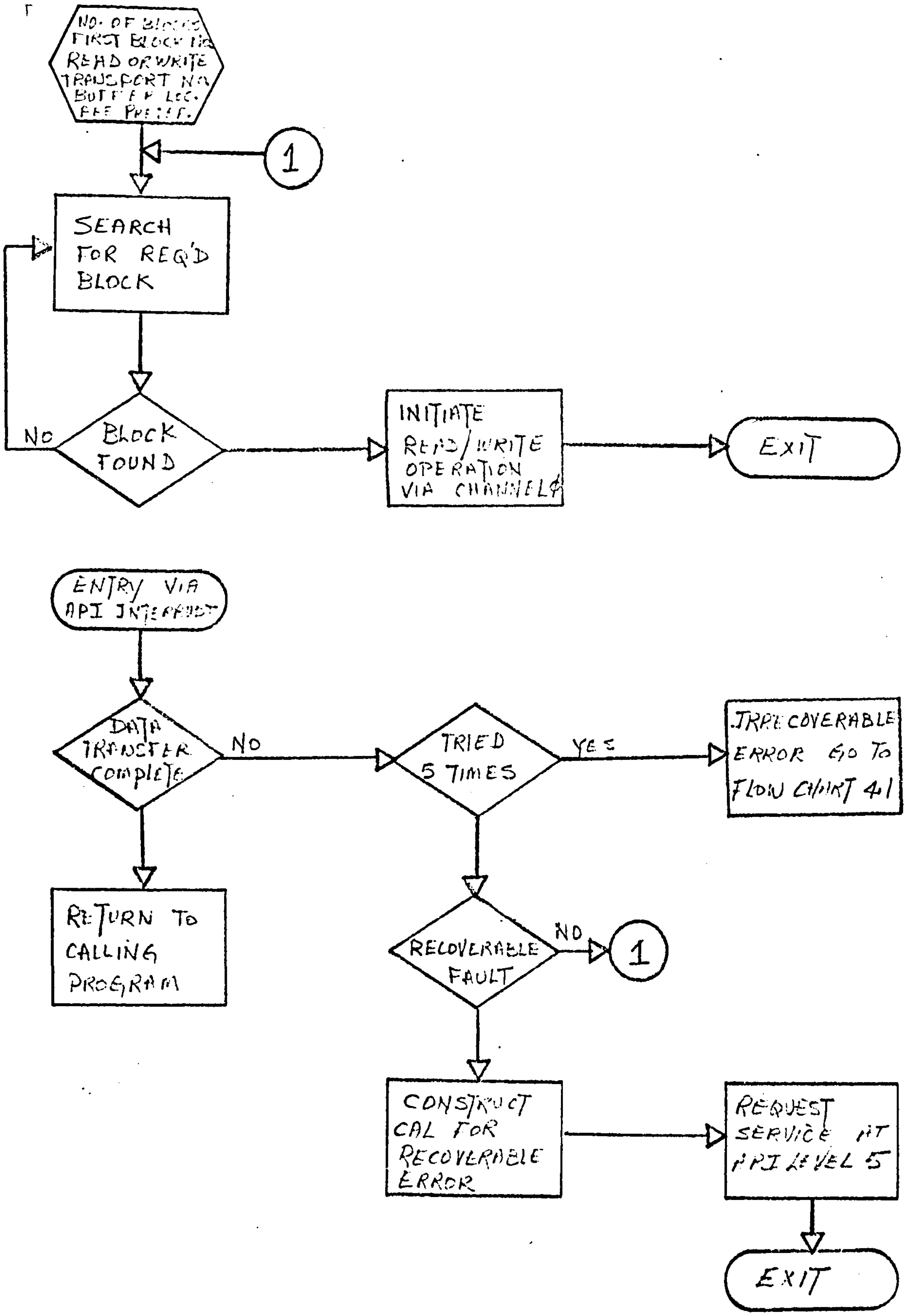
DECTAPE FUNCTION HANDLING

FLOW CHART 4.7

the direction of transfer, are predefined. Status register "A" of the dectape is locaded to set in motion and interrupt when a block number is read. Control then returns to the Scheduler. When an interrupt occurs an API break at level 2 causes the dectape handler to be entered via a Jump Store instruction in the Trap location 43.

If the error flag is active, and it is a selection error, a recoverable error print is requested via API level 5. When the operator has corrected the fault and typed R, the search process is repeated. If the error is End-of-Tape, the direction of movement is reversed and loaded into the "A" register. For all other errors, the process is repeated up to 5 times, and if not successful, an irrecoverable error print is requested via API 5, and the user is aborted. If the data flag is up, a calculation is performed. If the dectape has passed the block required, the dectape direction is reversed, and status register "A" is loaded to move the number of blocks different from the required block. If the dectape has not reached the required blocks, status register A is set to move the dectape the required number of blocks to reach the one required. When the right block is found, the CHANNEL[10] control registers are loaded with the starting core location, and the number of words to read or write the required data. When the transfer is complete status register "A" is loaded to stop the dectape and the handler is exited. Refer to Flow Chart 4.8.

Dectape is interfaced to the PDP-9 computer via CHANNEL 0. Register core location 30 is used to store the block number, when dectape control status register "A" is loaded with a search command. Core location 30 is loaded with the buffer pointer, and core location 31 is loaded with the number of characters to be transferred, with status register "A" indicating a read or write operation.



DECTAPE HANDLER

FLOW CHART 4.8

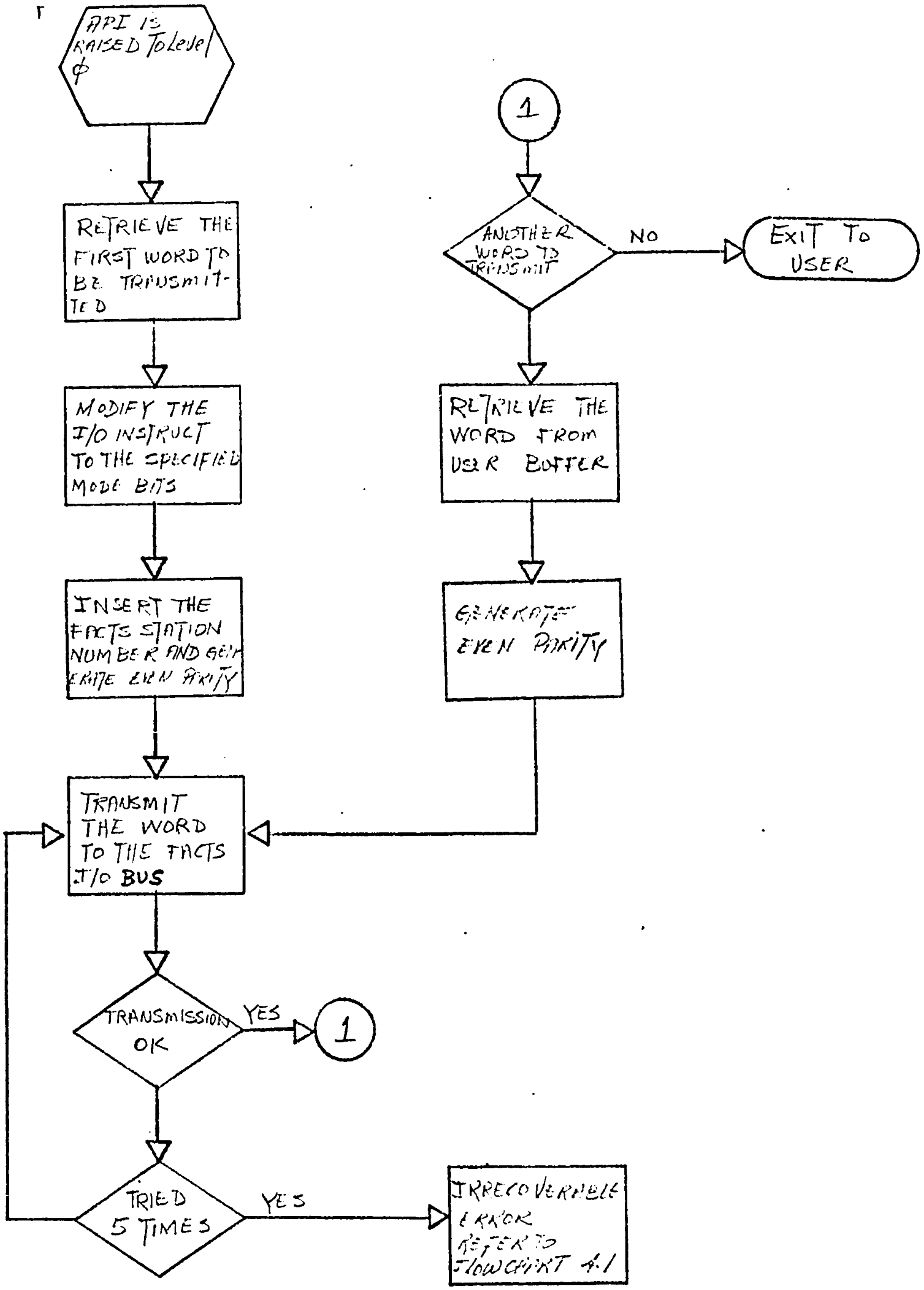
4.6 - FACTS STATION TRANSMISSION HANDLER

When a user requests an I/O transmission to a Facts Station, control is returned to the user directly after transmission. The user buffer points to the word or words to be transmitted. There are two formats of words transmitted to the Facts Station. A complete address word, specified by AC^5 bit 2 on a 1, and a partial address word and wordfill combination, specified by AC^5 bit 2 on a 0, of the respective first word. AC^5 bit 0 and 1 indicate whether $IOP4^5$ and $IOP2^5$ will be transmistted when the I/O instruction is executed to transmit the data.

The handler retrieves AC bit Ø and 1 and modifies the I/O instruction used to transmit the data. The Station address is placed AC bits 0, 1, and 2 of the first word to be transmitted so that only the properly addressed Station will accept the data. A parity bit in location AC 17 is calculated for even parity, and the first word is transmitted. The parity error flag at the FACTS station is interrogated and if active the transmission is repeated up to 5 times. If the parity flag is still up, the user is aborted, and a parity error print is initiated. If the parity flag is not up, the next word is transmitted (if there is one) and control reverts directly back to the User Program. The status of the station is unconditionally read and stored. Refer to Flow Chart 4.9.

4.7 - A/D HANDLER

The A/D Handler is single threaded. It accesses the CALL data via the "XGENQ" queue. The user buffer pointer specifies the expected polarity of the voltage to be measured. The handler connects

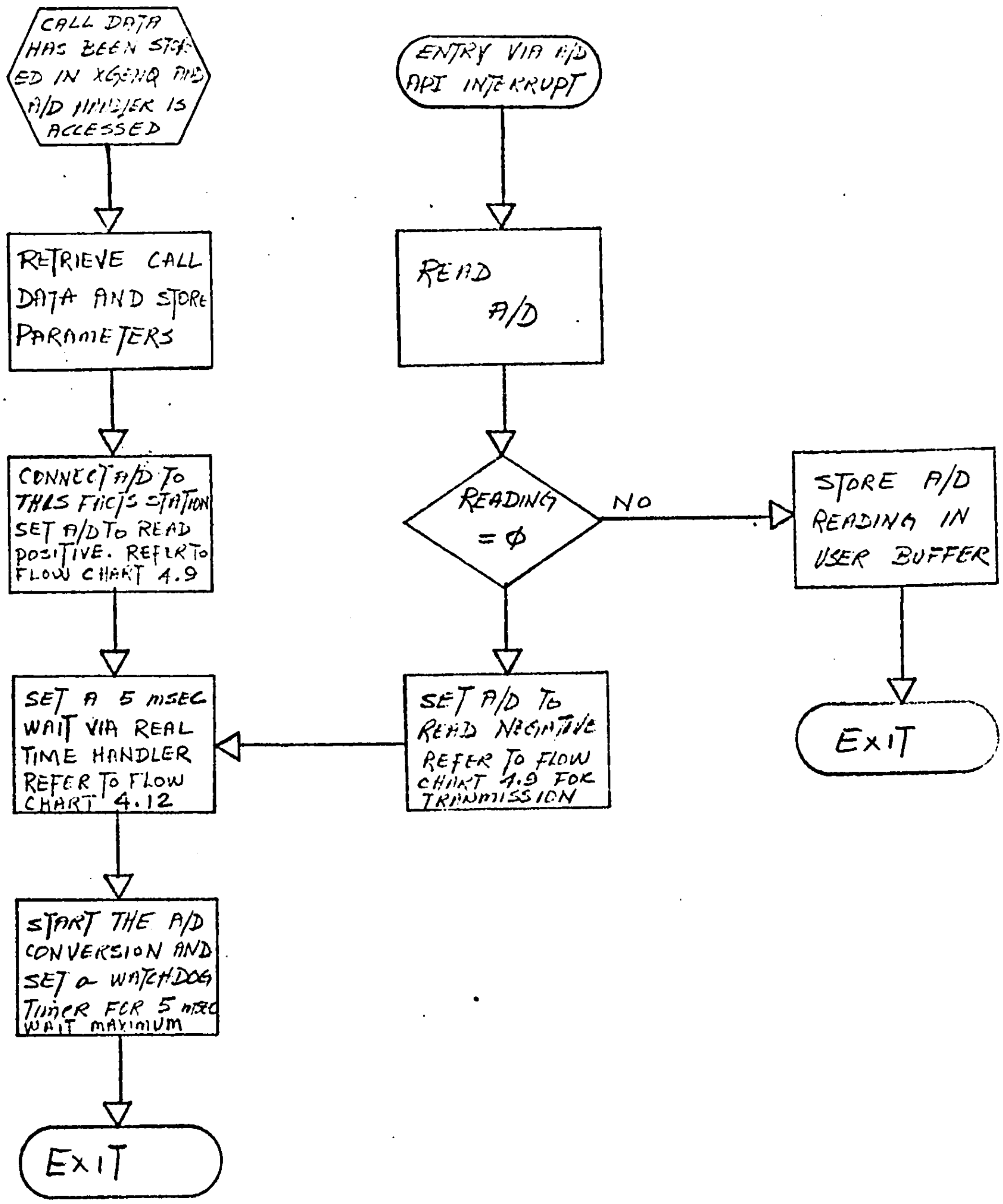

API IS RAISED TO LEVEL φ
RETRIEVE THE FIRST WORD TO BE TRANSMITTED
MODIFY THE I/O INSTRUCT TO THE SPECIFIED MODE BITS
INSERT THE FACTS STATION NUMBER AND GENERATE EVEN PARITY
TRANSMIT THE WORD TO THE FACTS I/O BUS
TRANSMISSION OK
YES
1
TRIED 5 TIMES
YES
IRRECOVERABLE ERROR REFER TO FLOWCHART 4.1
1
ANOTHER WORD TO TRANSMIT
NO
EXIT TO USER
RETRIEVE THE WORD FROM USER BUFFER
GENERATE EVEN PARITY

FACTS STATION TRANSMISSION HANDLER FLOW CHART 4.9

the FACTS STATION ANALOGUE LINE to the A/D and set the A/D to read according to the specified polarity. The Real Time Handler is accessed with a specified wait request of 5 msec. to let the relays stabilize. The I/O initiator real time handler indicator is set so that the Real Time Handler will cause a return to the Scheduler, after setting up the 5 msec. wait.

When the 5 msec. wait is complete, the Real Time Handler returns control to the A/D Handler via "XTTBL2" pointer previously set by the A/D time request. The A/D Start Conversion I/O instruction is executed. The Real Time Handler is accessed for a watchdog time wait of 5 msce. If the A/D conversion is not complete within 5 msec., a Monitor Error is printed indicating the A/D is OFF. If the A/D conversion flag comes up first the wait request is dismissed. If the A/D has a Zero reading, it is set to read the opposite polarity and the user polarity indicator is complemented. The Real Time Handler is accessed for an unconditional wait of 5 msec. for the polarity relays to stabilize.

After the 5 msec. has expired, the Real Time Handler accesses the A/D Handler and the A/D Start Conversion I/O instruction is executed. A watchdog 5 msec. time is set. When the A/D Finished Flag is set, an API level Ø interrupt is requested and the A/D Handler is accessed via the Jump Store in trap address 57. The answer is stored in table "XTTBL1" with reference to the FACTS station number. Refer to Flow Chart 4.10.


CALL DATA HAS BEEN STORED IN XGENQ AND A/D HANDLER IS ACCESSED
RETRIEVE CALL DATA AND STORE PARAMETERS
CONNECT A/D TO THIS FACTS STATION SET A/D TO READ POSITIVE. REFER TO FLOW CHART 4.9
SET A 5 mSEC WAIT VIA REAL TIME HANDLER REFER TO FLOW CHART 4.12
START THE A/D CONVERSION AND SET a WATCHDOG TIMER FOR 5 mSEC WAIT MAXIMUM
EXIT
ENTRY VIA A/D API INTERRUPT
READ A/D
READING = ∅
NO
STORE A/D READING IN USER BUFFER
EXIT
SET A/D TO READ NEGATIVE REFER TO FLOW CHART 4.9 FOR TRANSMISSION

A/D HANDIER

FLOW CHART 4.10

4.8 - REAL TIME HANDLER

The Real Time Handler is a multithreaded Handler. There are three operators defined in this handler: an unconditional wait, a timing of an external operation synchronized to the application of stimuli, and the time between the operation of two external events. There are two clocks defined: a hardware time that increments location 7 in core every 100 usec. when enabled, and a software 102.4 msec. timer derived from the 100 usec. clock.

Upon entry into the Real Time Handler, the call data is stored, and the return pointer and the user buffer pointer are stored in Tables "XTTBL1" and "XTTBL2" respectively, with reference to the FACTS number of the user.

If the user requests an unconditional wait, the time interval is accessed via the user buffer pointer and the time interval routine is entered. For external timings, the user buffer address points to the elemental commands to be transmitted to the FACTS station, followed by the maximum wait allowed for the external timing.

If the user requests an external timing, the elemental command is referred to via the user buffer address, and the FACTS station transmission handler is entered. After the transmission is complete, if the active station is FACTS II, the hardware timing CHANNEL A is enabled; if the active station is FACTS I, and the specified timing is synchronized to the stimuli, the return pointer set in Table "XTTBL4" will initiate the timing of the external event when the FACT station time flag becomes active. If the active station is FACTS I and the external timing is between two external events, the pointer set in Table "XTTBL4" causes the first time flag to be ignored, and the second to initiate the external timing. A watchdog time wait is initialized via the Time Interval Routine. If this time elapses before the FACTS station time flag comes up, the user will regain control with an overflow indication in the answer table "XTTBL1". Refer to Flow Chart 4.11.

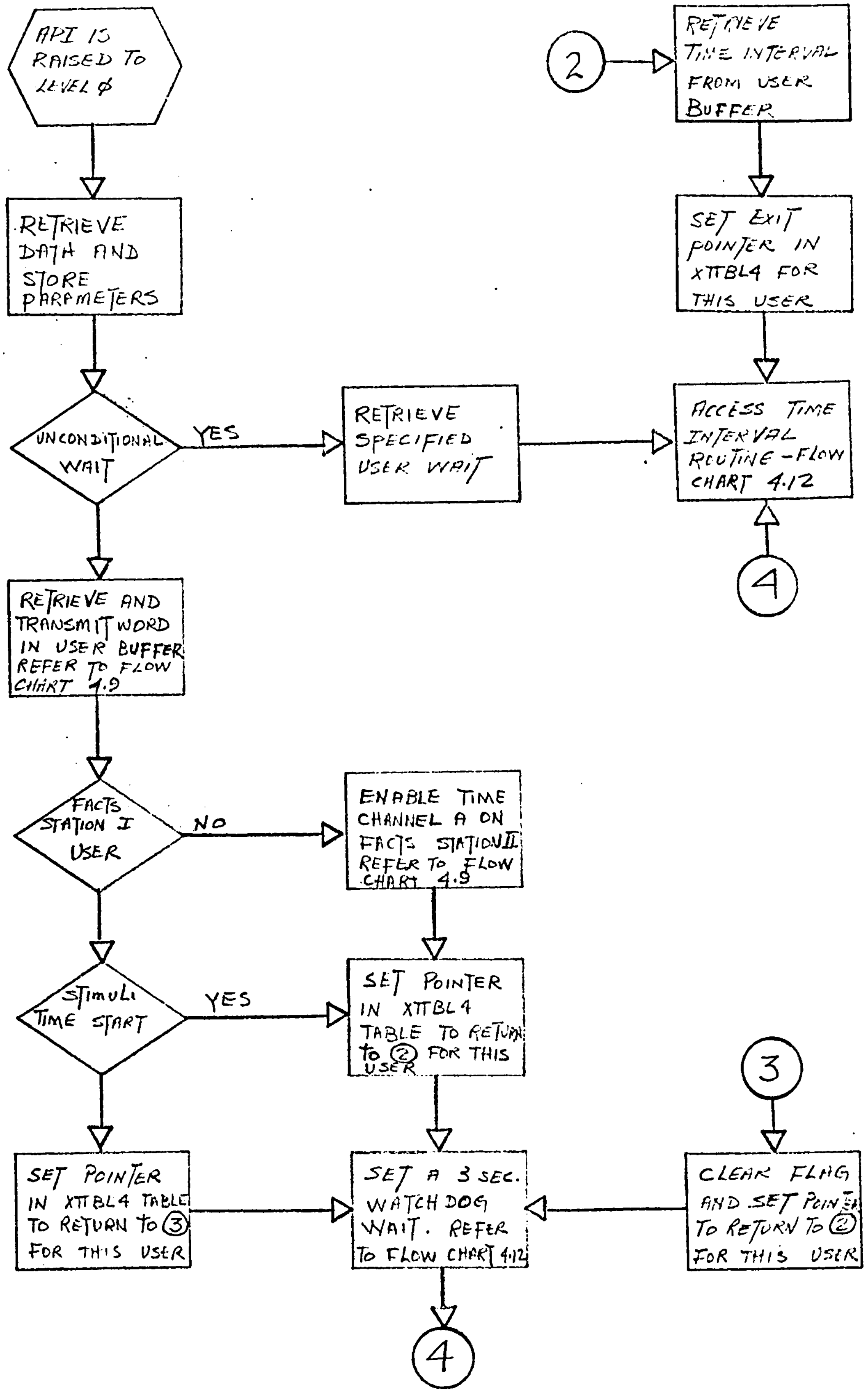
4.8.1 - Time Interval Routine

The first user of the Time Interval Routine has the specified wait stored in table "XTTBL5" in the clock start indicator "XTIMST" and the 2's complement of the wait in core location 7. The Real Time Clock is enabled. It will cause an interrupt when location 7 goes to zero. If the specified timed wait is greater than 128K*, using the 102.4 msec. counter, 128K is stored in location 7 and "XTIMST", and the clock is enabled.

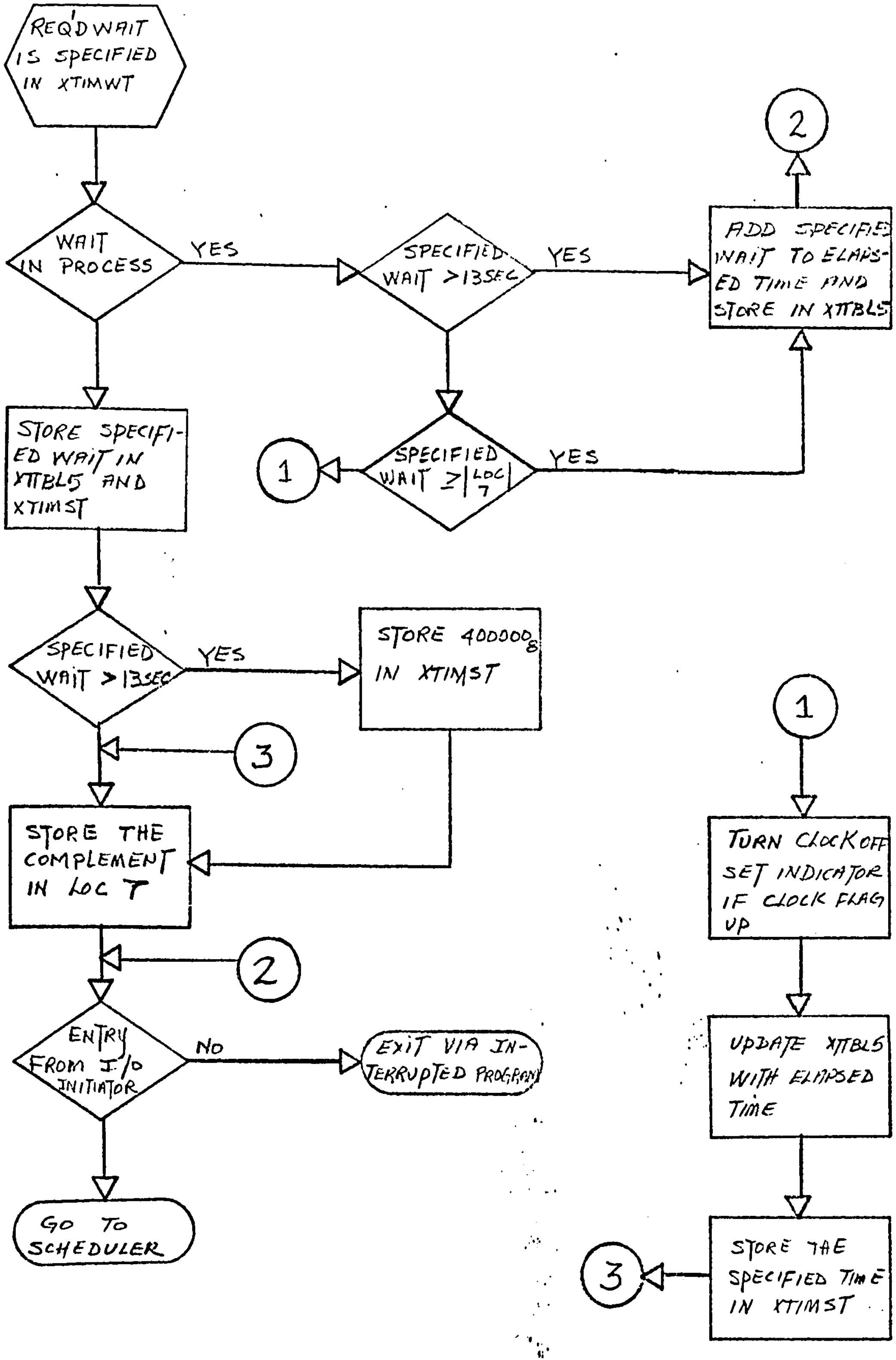
If the Time Interval Routine is in process, and the specified wait is greater than the time interval remaining to be counted in location 7, the contents of "XTIMST" plus location 7 is added to the specified wait. It is subsequently stored in table "XTTBL5" and the routine is exited. If the specified time is greater than 128K, AC bit 0 on a 1, the calculation is done in units of 102.4 msec. When the specified wait is less than the time left to elapse, the smaller time is set in location 7 and "XTIMST" and table "XTTBL5" is updated to reflect the elapsed time prior to the new setting of time to be expanded. Refer to Flow Chart 4.12.

Table "XTTBL5" is updated by calculating the elapsed time in .1 msec. and 102.4 msec. increments and by subtracting these times from each active user in the table. If the entry is specified in 128K increments the 102.4 msec. interval is used otherwise the .1 msec.

(*) K indicates 1024 X 100 usec. time units.


API IS RAISED TO LEVEL Ø
RETRIEVE DATA AND STORE PARAMETERS
UNCONDITIONAL WAIT
YES
RETRIEVE SPECIFIED USER WAIT
RETRIEVE AND TRANSMIT WORD IN USER BUFFER REFER TO FLOW CHART 4.9
FACTS STATION I USER
NO
ENABLE TIME CHANNEL A ON FACTS STATION II REFER TO FLOW CHART 4.9
STIMULI TIME START
YES
SET POINTER IN XTTBL4 TABLE TO RETURN TO ② FOR THIS USER
SET POINTER IN XTTBL4 TABLE TO RETURN TO ③ FOR THIS USER
SET A 3 SEC. WATCH DOG WAIT. REFER TO FLOW CHART 4.12
4
2
RETRIEVE TIME INTERVAL FROM USER BUFFER
SET EXIT POINTER IN XTTBL4 FOR THIS USER
ACCESS TIME INTERVAL ROUTINE - FLOW CHART 4.12
4
3
CLEAR FLAG AND SET POINTER TO RETURN TO ② FOR THIS USER

REAL TIME HANDLER FLOW CHART 4.11


REQ'D WAIT IS SPECIFIED IN XTIMWT
WAIT IN PROCESS
YES
SPECIFIED WAIT >13SEC
YES
ADD SPECIFIED WAIT TO ELAPSED TIME AND STORE IN XTTBL5
2
STORE SPECIFIED WAIT IN XTTBL5 AND XTIMST
1
SPECIFIED WAIT ≥ |LOC 7|
YES
SPECIFIED WAIT >13SEC
YES
STORE 400000₈ IN XTIMST
1
3
STORE THE COMPLEMENT IN LOC T
TURN CLOCK OFF SET INDICATOR IF CLOCK FLAG UP
2
ENTRY FROM I/O INITIATOR
NO
EXIT VIA INTERRUPTED PROGRAM
UPDATE XTTBL5 WITH ELAPSED TIME
GO TO SCHEDULER
3
STORE THE SPECIFIED TIME IN XTIMST

TIME INTERVAL ROUTINE

FLOW CHART 4.12

interval is used in the calculation.

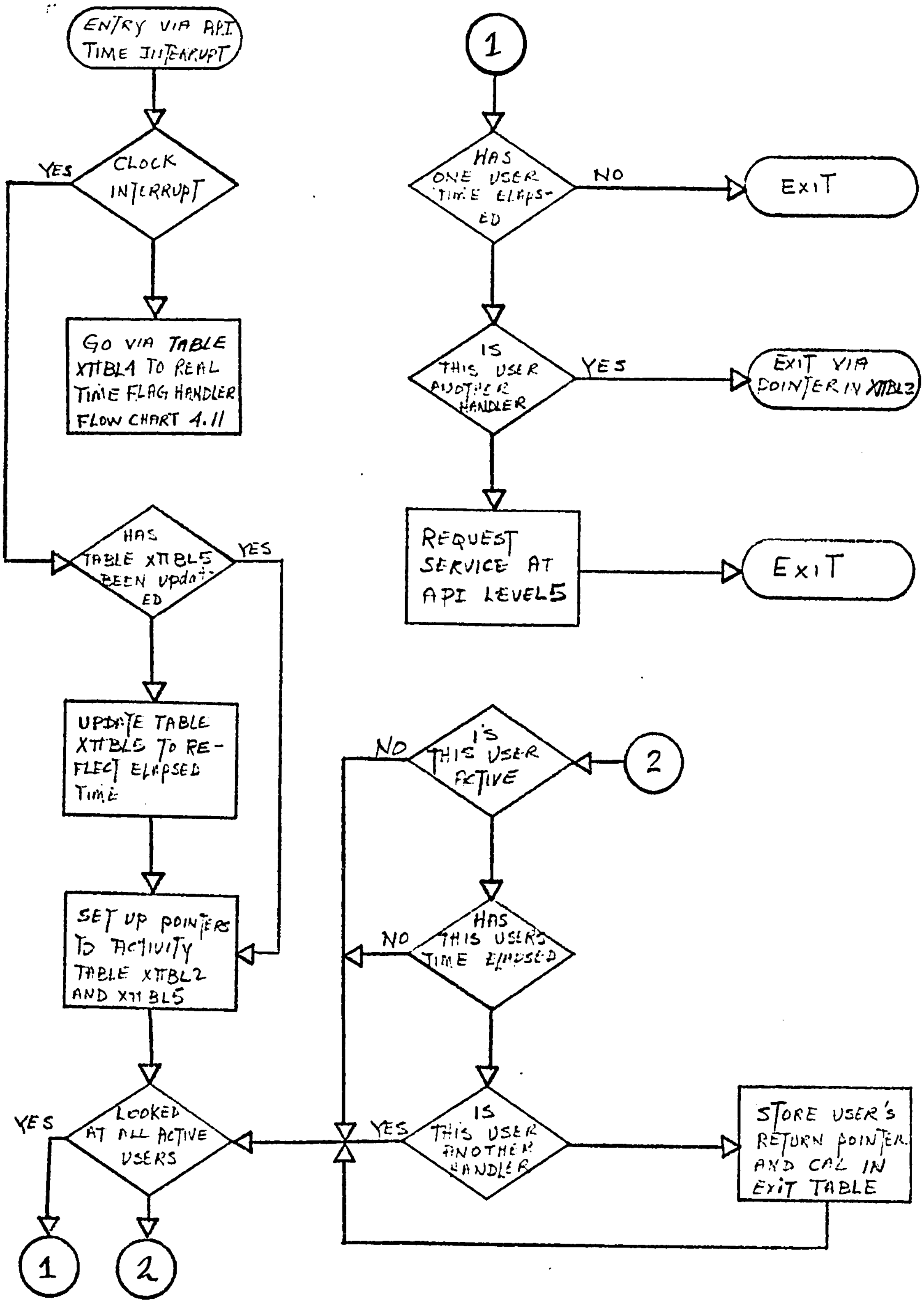
4.8.2 - Real Time Flag Handling

When a time flag becomes active, it causes an API level Ø interrupt which initiates a Jump Return from the trap address 70 to the Real Time Handler skip chain. There are three legal flags defined: the internal clock flag which is raised when location 7 in core is incremented to zero, and the time flags from the FACTS STATIONS 1 AND 2.

When the clock flag is up, the clock service routine is entered. Flow Chart 4.13 shows the detail processing of the clock flag. If a FACTS station time flag is raised, the time flag routine is entered.

4.8.2.1 - Clock Service Processing

The clock service routine checks each entry in table "XTTBL5" that is active as specified in table "XTTBL2". Once it has found one user which has completed the specified wait, it initiates a user recovery by loading the return CAL and user pointer in table "XTBL11". Table "XTTBL1" is loaded with Ø if flag 1 is indicated in table "XTTBL2", and -Ø to indicate overflow of flag 2. These indicators are useful only if the user has specified an external timing. If there are other entries in table "XTTBL5" that are not zero, the smallest of these will be used to set up the new internal counter in core location 7. If more than one user has completed his specified timed interval, an additional time wait of 100 usec. will be specified because only one user can exit from the Real Time Handler at any one time. When all eight entries in table "XTTBL5" have been examined, the routine will be exited either to the


ENTRY VIA API TIME INTERRUPT
CLOCK INTERRUPT
YES
GO VIA TABLE XTTBL4 TO REAL TIME FLAG HANDLER FLOW CHART 4.11
HAS TABLE XTTBL5 BEEN UPDATED
YES
UPDATE TABLE XTTBL5 TO REFLECT ELAPSED TIME
SET UP POINTERS TO ACTIVITY TABLE XTTBL2 AND XTTBL5
LOOKED AT ALL ACTIVE USERS
YES
1
2
1
HAS ONE USER TIME ELAPSED
NO
EXIT
IS THIS USER ANOTHER HANDLER
YES
EXIT VIA POINTER IN XTTBL2
REQUEST SERVICE AT API LEVEL5
EXIT
IS THIS USER ACTIVE
NO
2
HAS THIS USERS TIME ELAPSED
NO
IS THIS USER ANOTHER HANDLER
YES
STORE USER'S RETURN POINTER AND CAL IN EXIT TABLE

CLOCK SERVICE ROUTINE

FLOW CHART 4.13

calling handler or to the user whose timing is complete.

4.8.2.2 - Facts Station Flag Processing

If the API interrupt was caused by a FACTS station time flag, the return pointer to the service routine is made via table "XTTBL4". When the synchronizing time flag is received, the user specified maximum time wait is accessed and stored in "XTIMWT" for processing by the time interval handler. The return pointer to the time completion routine is loaded in table "XTIBL4",. and the time interval routine is accessed. If the FACTS 2 status is active, timing CHANNEL B is enabled.

When the next time flag is received from the FACTS station, the real time completion routine is entered via table "XTTBL4". The elapsed time is calculated by adding the contents of "XTIMST" to location 7, subtracting this from the time specified in table "XTTBL5" and adding the complement to the time specified by the user referenced via table "XTTBL2". This is stored in table "XTIBL1" for user access. The time handler is exited via an API level 5 request.

CHAPTER 5

DEFINITION OF SYSTEM USER PROGRAMS

The SUP, SYSTEM USER PROGRAMS, which consist of foreground Op/Sys, the background testran compiler, the Editor, and the BTP, are initially entered via the Monitor, by a Jump Store instruction in the user entry tables**, with the reference address pointing to the User Program file name. This file name is transferred by the SUP to the user buffer area for future reference to access the file on mass storage. The user is uniquely identified by FACTS station number** which is defined by the TTY Keyboard being used by the operator. When the Scheduler transfers control to the user, this number can be referred to by the SUP for I/O calls, data retrieval from the handlers, and internal processing.

The Test Programs used to control the FACTS STATION are coded in a pseudo english language called TESTRAN[12]. It consists of commands such as: Connect, Release, PS1 Set, (Power Supply #1 Set), Read Instruction - Scan, Adread, Operator Messages, etc. These mnemonics are followed by a stimulus field, a measurement value field, and an access point field, which defines how the unit under test is interfaced to the FACTS test consoles. The Testran Compiler translates these instructions in to ELEMENTAL COMMANDS and stores them on dectape. The format of EC's are shown in Figure 5.1. EC's which appear as DATA are loaded 1 dectape block at a time, and are operated on by Op/Sys to control the test consoles. Each user has a total of 512 (decimal) locations in core where variable data are stored.

(**) Refer to Keyboard Handler.

5.1 - OP/SYS

The Op/Sys is a re-entrant program which operates on the user test programs in ELEMENTAL COMMAND format. When a user requests service via the Monitor, enter into the Op/Sys at absolute location 10000 (octal) the specified file name is accessed from the user entry table, and is stored in the user buffer 512 (decimal) locations.

The file is subsequently opened on dectape with reference to the "XFATNO" register which contains the FACTS station number. The Op/Sys initiates the typing of "FACTS N" where N is the FACTS station number, by an I/O CALL to the Print Handler. Upon completion, the Monitor will transfer control back to the Op/Sys. A Keyboard I/O call will then be generated to permit the user to type in a command or processing mode instruction.

The basic commands defined in the Op/Sys are: "Start" for the user to initialize the testing process and "Continue" to allow the operator to force continuance of the Test Program after a manual test operation or measurement error print out.

There are three test modes that can be initialized by the operator which will control the measurement printout: all data, reject data only, or no data printout.

When the "Start" command is typed, the first block (256 locations) of the test program is read from dectape into the user buffer. This contains the parameters required to program the specific test console that is to be controlled. These parameters which are resident throughout the processing of the user program will be accessed by the Op/Sys via the FACTS station number while processing EC's.

The next user program page is read into the user buffer. The first EC is accessed, and the function bits are decoded in order to reference the routine that will process the data bits.

There are 4 kinds of functional routines in the Op/Sys for processing elemental commands:

a) Connect or relase control points.

b) Connect stimuli to one or more access points.

c) Set up measurements for voltage, resistance, capacitance, the presence of a ground or an open circuit, interval timing and unconditional waits.

d) Operator communications to indicate the details for a manual operation.

When the handling of the first EC has been completed, the next is accessed and the process is repeated until the user test program page is exhausted. At it's completion, the Op/Sys initiates an I/O read request from dectape for the next test program page, and these EC's are subsequently processed. When the "END" EC in the user program is encountered, the command mode is re-enabled via a call to the Keyboard Handler, after the FACTS N has been typed, and the test program is ready to be rerun.

5.2 - TESTRAN COMPILER

There are up to seven Testran Compiler used within the FACTS time shrare system. Each Compiler contains the parameters that are peculiar to every FACTS test station programming. The Compiler translates mnemonic language statements written in Testran into ELEMENTAL COMMANDS which permits the Op/Sys to control widely different test consoles in a universal manner.

When a user requests service via the Monitor, the Scheduler loads the Testran Compiler as a function of the FACTS station number. When the loading is complete, the Compiler is accessed via the user entry table by a Jump Store instruction, with the reference address pointing to the test source program file name. The Compiler transfers the file name to the user buffer, and initiates an I/O to open the file via the dectape handler. The first page of the source program is subsequently read, and the compilation process begins.

The first object block generated, consists of the program identification, and the parameters specific to the particular FACTS station to be programmed. Following this, the source program statements are decoded and stored as elemental commands. When one block of data has been generated (255 decimal locations) a dectape write I/O request is generated. When the transfer is complete, the process is continued until an END statement is read in the source program. The last program block is written onto dectape and an END call is made to the Monitor. This terminates the compilation.

The Compilers verify Syntax and logic combinations of commands with the data in the other fields of a source program statement. If an error is detected, the source line is printed with an error code. The compilation process resumes in a diagnostic mode which identifies the remaining error in the source program without transferring object data to mass storage. The hardware interconnections, by which the unit under test is interfaced with the test console, are specified in the source test program.

The Compiler verifies that the test program's specified combinations of stimuli, connected through the hardware cross connecting matrix to the unit under test, are within the limitations of the FACTS test station.

5.3 - EDITOR

The Testran Editor performs normal edit functions such as inserting, changing,appending, and deleting source program statements in a Testran laguage program. In addition, it formats the source program in terms of 60 lines per page, and references source program data by line and page number.

When a user requests Editor processing via the Monitor, the Scheduler initiates the loading of the Edit SUP. Subsequent to loading, the Editor is entered by a Jump store instruction in the Monitor via the user entry table. The reference address points to the user source program file name. The file name is transferred to the user buffer, and the file is OPENED via an I/O call to the dectape handler by the Editor.

When control returns to the Editor, a Command** I/O request is made via the Keyboard Handler for user interaction. Commands typed by the user are decoded and the source file is manipulated. If Text has to be entered by the user, the Editor will initiate an I/O Text** request via the Keyboard Handler.

When the editing process is complete, the operator is permitted to rename the source file if the previous one is to be kept. The file

(**) - Refer to Keyboard Handler.

is subsequently closed via a user command, and the updated file is copied from the scratch file to the user dectape.

5.4 - BACKGROUND TASK PROGRAM

The BTP permits a user to transfer files from one peripheral device to another. When the user requests a BTP function via the Monitor, the Scheduler will load the BTP. A Jump Store instruction, executed in the Monitor via the user entry table, accesses the BTP with the reference address pointing to the file name if applicable. Transfers of data between non-file oriented peripherals are also legal. If a file name is present it is transferred to the user buffer for future reference. A Text I/O request is made via the Keyboard Handler. The user types the pertinent instruction string defining the peripherals, the direction of transfer, the format of the input and output, image, and packed ASCII or unpacked ASCII. This data is decoded and the transfer is initiated. When complete, an I/O END request is initiated which terminates the BTP processing.

CHAPTER 6

FACTS MONITOR FEATURE TESTING

The Hardware configuration (refer to Figure 1.1) that was used for comprehensive testing of the FACTS Monitor consisted of:

1) PDP-9 Computer with 8K memory.

2) API Option KF09A.

3) Memory Protection Option KX09A.

4) Two Dectape Transports and Controller.

5) FACTS Interface equipped to handle two FACTS Stations.

6) An A/D Convertor.

7) Two FACTS Stations number 1 and 2.

8) 4 Teletypes.

The Monitor verification consisted of processing User Programs that exercised the Monitor features such as the Priority Structure, Swapping of Programs, I/O handling the paper tape, Dectape, and TTY's, as well as the special feature required by the FACTS Test Sets such as Timing, A/D handling, etc.

The following jobs were run:

1) Processing Test programs on the FACTS 1 and FACTS 2 Test Stations, while a source program was being listed from the paper tape reader, on the FACTS 3 TTY by a BTP. The testing programs included control functions on the Test consoles, the measurement of voltage and time intervals on the units under test.

2) The Op/Sys on the FACTS 1 Test Set was aborted and an Edit function was requested. This caused the BTP, which was processing

on FACTS 3, to be swapped out of core and the Editor to be subsequently loaded. Suspention of typing occurred on the FACTS 3 TTY. The Editor was requested to output a User Program on the FACTS 1 TTY from Dectape.

3) The Editor program if not complete, was aborted which caused the BTP to be reloaded into core. The program listing on the FACTS 3 TTY resumed when the loading was complete.

For the Monitor Testing the dectapes used were generated OFF line using the FACTS Dectrive[11] program. An OFF line compilation of two Testran Programs was loaded onto the dectape including serveral source User Programs.

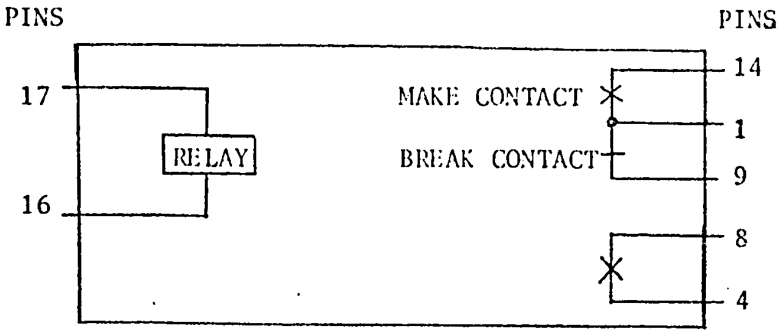
6.1 - TEST PROGRAM FOR FACTS 1 STATION

There were two distinct modules selected to test the features of the FACTS Monitor. The circuit pack that was tested on the FACTS 1 Test Station is shown schematically on Figure 6.1.

The tests consisted of:

1) Measuring that the make contacts are open, using the A/D converter.
2) Measuring that the break contacts are made using the A/D converter.
3) Measuring that the operate time of the relay is less than 50 milliseconds.

The detail Testran[12] program to do the tests are shown in Figure 6.2



TEST CIRCUIT 1 FIGURE 6.1

```
STATEMENT
 NUMBER             FACTS 1 DUMMY TSI ISSUE 1

    1               PS1SET    -24

    2               CON       PS1      A17

    3               CON       GND      A1

    4               SCAN-T    OPEN     A4       A14

    5               SCAN-C    GND      A9

    6               TIME-T    50MS     MAX.     GND

    7                         A16      A14

    8               END
```

TESTRAN TEST PROGRAM FOR FACTS 1 FIGURE 6.2

```
WORD NUMBER

     1           055070

     2           075200

     3           360202

     4           055240
```

PARTIAL COMPILATION OUTPUT FOR TEST CIRCUIT 1 FIGURE 6.3

The statements of the Testran program in Figure 6.2 have the following interpretation:

1) Preset Power supply number 1 to -24 volts.
2) Connect Power supply number 1 to pin 17 of UUT.
3) Connect Ground potential to pin 1 of UUT.
4) Scan for an open circuit condition at pin 4 and 14 of UUT.
5) Scan for continuity to a Ground potential at pin 9 of UUT.
6) Measure the operate time of the relay by synchronizing to the application of a Ground potential at pin 16 and monitoring when pin 14 measures a Ground potential.

6.1.1 - Testran Compiler

In order to get this source program into the format required for the operating system to process, it must be compiled. The Compiler for the FACTS 1 Test Set is programmed with all the hardware features of the FACTS 1 Test System. The Testran program was compiled and stored on dectape OFF line. An example of the compiler output is shown in Figure

The octal words in Figure 6.3, which are called Elemental Commands, represent the Compiler output for the statement numbers 1 and 2 of the test program as shown in Figure 6.2. Word 1 of Figure 6.3, which is a C.A.W.***, causes the power supply 1 programming bits at the FACTS Station 1 to be cleared. Words 2 and 3, which are a P.A.W.*** W.F.*** combination,programs power supply 1 to -24 volts. Word 4, C.A.W., enables power supply 1.

In addition to the test program, the Compiler stored the name of the program and required parameters for the operating system to program the FACTS 1 Test System on dectape.

(***) - Refer to Figure 7.3 for definitions.

6.2 - TEST PROGRAM FOR FACTS 2

The circuit pack that was tested on the FACTS 2 Test Station is shown schematically in Figure 6.4.

The Tests consisted of:

1) Reading a voltage using the A/D converter.
2) Measuring the operate time of the reed relay synchronized to the application of a ground potential at pin 22.

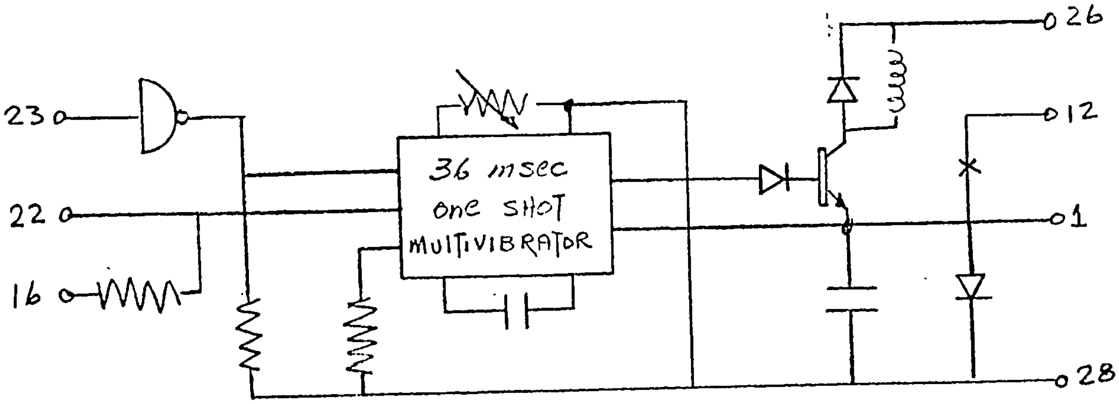
The detail Testran program to do these tests are shown in Figure 6.5. The interpretation of these Testran statements are similar to those of Figure 6.2. Statement 7 means: read the A/D for a voltage between 13.2 and 10.8 volts at pins 17, 26, and 27.

6.3 - OP/SYS PROCESSING

After the Monitor was loaded, the Op/Sys was loaded by the Monitor. In order to access the Op/Sys the operator typed ↑F followed by the file name of the test program to be processed.

The Monitor transferred control to the Op/Sys which printed "DK or DT" (Disk or Dectape file) followed by a command request which printed ">" on the user TTY. The operator responds with DK if the file in on DISK or DT if the file is on DECTAPE. After typing "DT" the FIRST Block of the user test program was read from dectape and the test program title was printed "FACTS 1 DUMMY TS1" followed by a command request ">".

If the title was incorrect, the user typed ↑A to abort the job and the process was repeated with the correct file name. If the



TEST CIRCUIT 2 FIGURE 6.4

```
1 FACTS 2 DUMMY TSI ISSUE 1

2 PS1SET +12V

3 PS2SET +5.12V

4 CON     PS1    A17     A26      A27

5 CON     PS2    A16     A28

6 CON     GND    A1      A23

7 AD READ       10.8V   13.2V    A17      A26      A27

8 TIME-T        30MS    42MS     GND

                 A22     A12

9 END
```

TESTRAN TEST PROGRAM FOR FACTS 2 FIGURE 6.5

```
↑F TPF 11

DK OR DT    DT↲

FACTS 1 DUMMY TS1 ISSUE 1

  Y↲

  AD↲

  S ↲
```

OP/SYS OPERATOR COMMUNICATION FIGURE 6.6

Title was correct, the operator typed "Y" for Yes and the Op/Sys then requested a command.

Several commands are legal at this time:

"AD" for requesting an all data printout.

"ST" for initializing the step mode.

"S" for starting the testing process.

After typing "AD" the Op/Sys responded with ">" to request the next command. By typing "S" the Op/Sys loaded the second dectape block of the test program and overlayed the FIRST Block. After loading was completed, the Op/Sys started processing the test program.

Figure 6.6 summarizes the communication of the operator with the FACTS Monitor at the FACTS 1 TTY. Underlined characters are typed by the operator.

6.4 - BACKGROUND USER PROGRAM

The BTP that was designed for testing the Monitor features had three transfer modes defined. Transfer of data between: Punched paper tape to a TTY, Punched paper tape to a dectape, and dectape to a TTY (refer to Figure 6.7). The format for requesting a transfer at a FACTS Test Station was: The operator typed ↑B delimited by a carriage return (⮐). This initiated a request on the Monitor to load the BTP. When the BTP was loaded, the Monitor initiated the BTP via the Scheduler. The BTP requested a TEXT input and printed "<" to indicate this to the operator. The operator typed one of three legal transfer commands as shown in Figure 6.7.

The BTP interpreted the command string and set up the proper I/O call links to perform the required transfer. Refer to Flow Chart 6.1 for the detailed process of the BTP.

If the format type or an undefined transfer was requested, "WHAT?" was printed and the Text buffer was cleared.

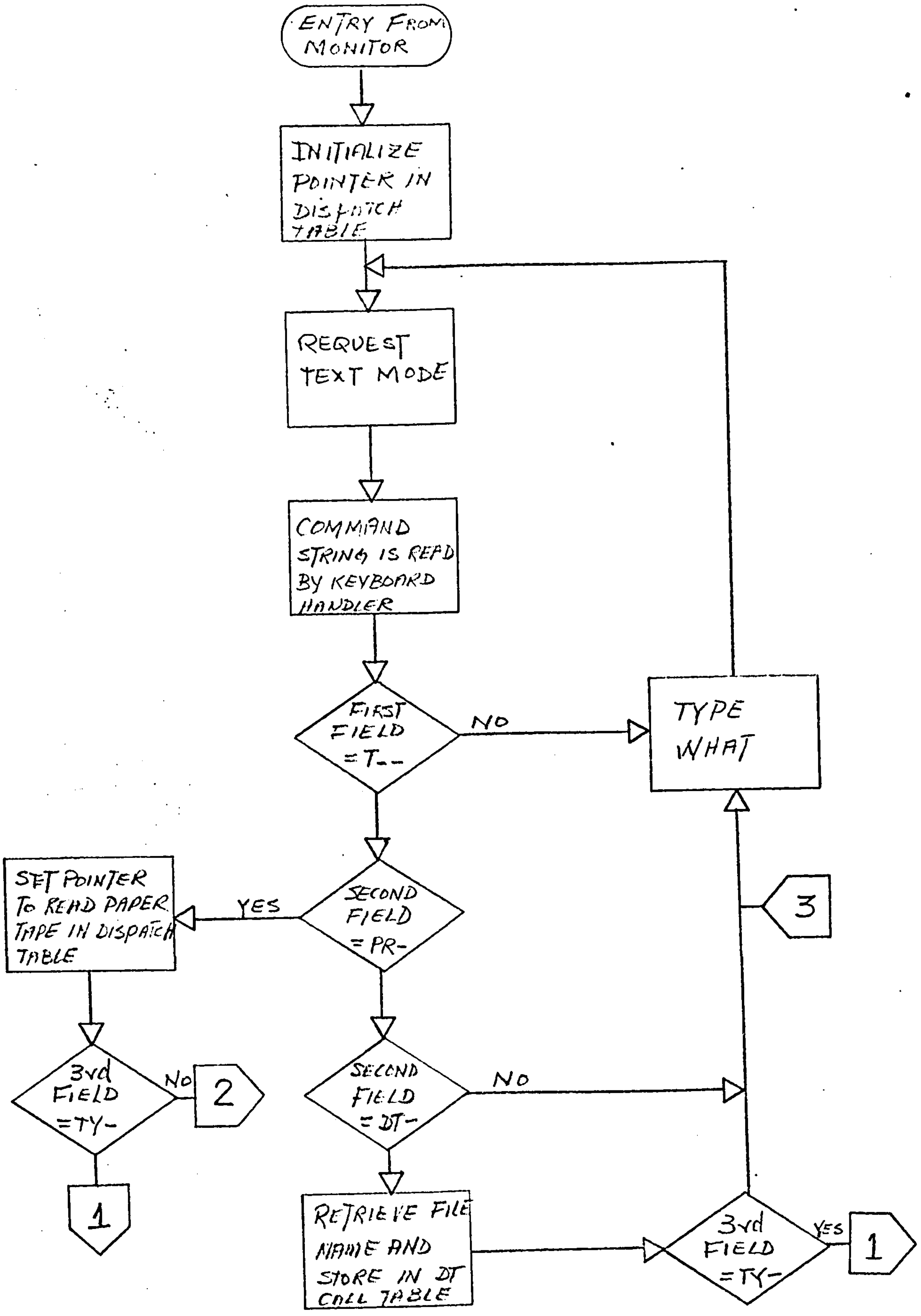
An example of a request for printing an ASCII paper tape on a TTY is shown in Figure 6.8. Underlined characters are typed by the operator.

```
T -- PR - TY - *

T -- PR - DT - FILENAME - *

T -- DT - FILENAME - TY - *

The minus indicate a space

* is the delimiter for TEXT MODE
```

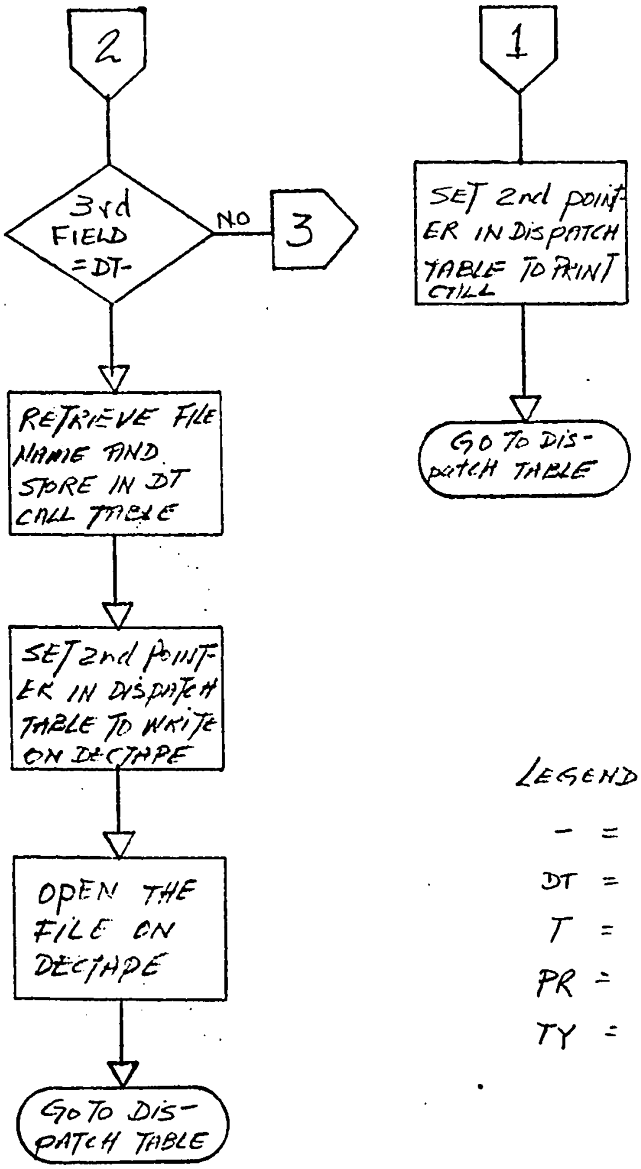
BTP COMMAND STRINGS FIGURE 6.7

```
↑B ↲

< T -- PR - TY - *

PRINTING WILL COMMENCE
```

BTP OPERATOR COMMUNICATION FIGURE 6.8


ENTRY FROM MONITOR
INITIALIZE POINTER IN DISPATCH TABLE
REQUEST TEXT MODE
COMMAND STRING IS READ BY KEYBOARD HANDLER
FIRST FIELD =T--
NO
TYPE WHAT
SECOND FIELD =PR-
YES
SET POINTER TO READ PAPER TAPE IN DISPATCH TABLE
3
3rd FIELD =TY-
NO
2
1
SECOND FIELD =DT-
NO
RETRIEVE FILE NAME AND STORE IN DT CALL TABLE
3rd FIELD =TY-
YES
1

BACKGROUND TASK PROGRAM FLOW CHART 6.1 PAGE 1 of 2


2
3rd FIELD =DT-
NO
3
RETRIEVE FILE NAME AND STORE IN DT CALL TABLE
SET 2nd POINTER IN DISPATCH TABLE TO WRITE ON DECTAPE
OPEN THE FILE ON DECTAPE
GO TO DISPATCH TABLE
1
SET 2nd POINTER IN DISPATCH TABLE TO PRINT CALL
GO TO DISPATCH TABLE

LEGEND:

-	=	APACE
DT	=	DECTAPE
T	=	TRANSFER
PR	=	PAPER TAPE READER
TY	=	TELETYPE

CHAPTER 7

SUMMARY OF CALL PROCEDURES FOR I/O HANDLER USAGE

In order to access a handler, from a User Program via the Monitor, a CAL[5] instruction must be executed with the format as shown in Figure 7.1.

Mnemonics have been defined to permit the user to construct the particular CAL instruction required. Appendix 4 defines all the mnemonics with their respective bit assignments, that can be used to perform specific I/O operations. The CAL[5] is constructed by a string of instructions as shown in CHAPTER 4.

Arg. 1 - There are four User Programs defined with the following Mnemonics.

XOPSYS - for the Operating System User

XTRSYS - for the Compiler User

XEDSYS - for the Editor User

XBKSYS - for the Background Task Program User

Arg. 2 and Arg. 3 - will be defined in the specific I/O Handler CALLS that follow.

Arg. 4 - There are three FACTS station numbers depending on the User Program being executed.

XFATNO - for the Operating System User

XFACTR - for the Compiler User

XFACBK - for the Editor of BTP User

Immediately proceeding the CAL instruction, there is a pointer to the user buffer where data will be transferred. This will be the case in all I/O requests via a CAL instruction. For reference, this pointer is called Arg. 5.

The User Program looses control after the CAL instruction is executed. Therefore, all parameters previously calculated must be saved in the user buffer prior to the CALL. Once the I/O operation is completed, control returns to the user at the core location, following the user buffer pointer. The arguments Arg. 1, Arg. 4, and Arg. 5 as defined, are used in the contruction of the I/O CALLS for specific handlers that follow.

7.1 - KEYBOARD HANDLER

There are two modes for accessing data from a TTY Keyboard, the COMMAND mode and TEXT mode. The command mode is used to access one to three characters from a Keyboard. In this mode, the handler permits up to 8 characters to be typed. When the delimiter CR is received the first 3 characters are transferrred to the user buffer location, as specified after the CAL[5] instruction. If more than 8 characters are typed, "WHAT?" is·printed and the command buffer is cleared.

The format for requesting the Keyboard command mode is:

```
LAC - Arg. 1            /System User Program
XOR - XRDCOM            /Read Keyboard Command Mode
XOR - Arg. 4            /Facts Station Number
DAC - .+1
CAL                     /Modified
Arg. 5                  /User Buffer Location
```

When the CAL instruction is executed, the Monitor will print ">" to indicate to the operator that a command is requested.

A rubout typed will clear the command buffer and a "\" will be echoed.

The format for requesting the Keyboard Text Mode is:

```
LAC - Arg. 1                 /System User Program
XOR - XRPTTY                 /Read Keyboard Text Mode
XOR - Arg. 4                 /Facts Station Number
DAC - .+1
CAL                          /Modified
Arg. 5                       /User Buffer Location
```

When the CAL instruction is executed, the Monitor will print "<" to indicate to the operator that the Text mode is initialized.

A rubout typed will clear the Text buffer and a "\" will be echoed.

Up to 750 decimal characters can be typed. The handler will ensure that a complete line of text will terminate the buffer.

A "*" is used as a delimiter. When the operator types "*", control returns to the User Program. When the buffer is full, the handler will force a "*" to be echoed and control will revert to the User Program. By typing "CR CR*" the User Program will recognize this as a request for the command to be initialized after the Text buffer is processed.

7.2 - PRINT HANDLER

There are two modes for printing data on a TTY. Printing from a 6 bit packed ASCII buffer or Octal printing.

The format for requesting the printing of a buffer in 6 bit ASCII is:

```
LAC - Arg. 1                  /System User Program
XOR - XPPTTY                  /Print for 6 Bit Packed ASCII
XOR - Arg. 4                  /Facts Station Number
DAC - .+1
CAL                           /Modified
Arg. 5                        /User Buffer Location
```

The buffer must be delimited by a zero character (00 octal) with the characters left adjusted. A "LF" and "Rubout" will be inserted after a "CR" and "TAB" is specified, respectively. The detail character assignment for 6 bit ASCII is shown in Appendix 2 .

The format for requesting an Octal printing of a user buffer is:

```
LAC - Arg. 1                  /System User Program
XOR - XPOTTY                  /Print Octal
XOK - Arg. 4                  /Facts Station Number
DAC - .+1
CAL                           /Modified
Arg. 5                        /User Buffer Location
```

The buffer must be delimited by a zero register. There are 24 Octal characters printed per line.

7.3 - PAPER TAPE READER HANDLER

The paper tape reader handler will accept standard ASCII paper tape and it will pack 3 characters per register in 6 bit format as defined in Appendix 2.

The format for requesting a paper tape to be read is:

```
LAC - Arg. 1                  /System User Program
XOR - XRDPT                   /Read Paper Tape & Pack
XOR - Arg. 4                  /Facts Station Number
DAC - .+1
CAL                           /Modified
Arg. 5                        /User Buffer Location
```

The handler will ensure that a complete line of Text terminates the user buffer. When the buffer is full, control is transferred to the user. When the end of the tape is detected Octal 77 is inserted as the last character in the buffer. This character can be used as a delimiter to the paper tape reading operation.

7.4 - PAPER TAPE PUNCH HANDLER

The paper tape punch handler will perforate paper tape in standard ASCII code from a 6 bit pack ASCII user buffer.

The format for requesting paper tape punching is:

```
LAC - Arg. 1                  /System User Program
XOR - XWTPT                   /Punch Paper Tape
XOR - Arg. 4                  /Facts Station Number
DAC - .+1
CAL                           /Modified
```

Arg. 5 /First Location of User Buffer

The buffer must be delimited by a ØØ octal character. If insufficient paper is available, a recoverable message "RER6" will be printed on the user TTY. After adding paper tape to the punch hardware, the operator must type ↑R for the punching operation to resume.

There is provision for recovering twice from an out of paper tape condition during one I/O punching operation. An irrecoverable error will occur if a third out of paper tape condition occurs.

7.5 - DECTAPE HANDLER

The magnetic tape mass storage is a directory oriented Dectape[5] whose format is 512 blocks of 256 18 bit words. The directory consists of 5 blocks having a format as shown in Appendix 3.

The first word of the partially filled directory block contains the next free block number. A directory entry consists of a file name in 6 bit ASCII occupying 3, 18 bit words have up to 8 characters with the unused characters being cleared. The following 18 bit word contains the number of blocks occupied by the file in bits 0 to 7 with the starting block in bits 8 to 17.

There are six functions defined in the Dectape Handler for accessing and manipulating files on dectape. They include opening or closing a file, read or writing a file in block or complete file mode.

The dectape transport number must be dialed to correspond to the FACTS station. The transport must be switched to "REMOTE" mode in order to be accessed. The WRITE mode switch must be enabled for WRITING OR CLOSING files. If any of these conditions are violated, a recoverable error printout "RER3" will occur on the user TTY. After the conditions are corrected, the operator types ↑R for processing to resume. There is provision for recovering 5 times from selections errors. An irrecoverable error will occur if a sixth selection error occurs.

7.5.1.- To Open a File

In order to read or write a file, it is first required to find where it is located on the dectape or obtain the first free block respectively.

The format for opening a file is:

```
LAC - Arg. 1                  /System User Program
XOR - XOPDT                   /Open File on Dectape
XOR - Arg. 4                  /Facts Station Number
DAC - .+1
CAL                           /Modified
Arg. 6                        /Location of File Name
```

In this case "Arg. 6" is the pointer to the file name and as FIRST user buffer location, as shown in Figure 7.2.

After the handler is accessed, via the "CAL" instruction, the FIRST directory block is loaded into core starting at the FIRST location of the user buffer "Arg. 5". The directory block is searched for a match and if one is found, the number of blocks and the starting block number is loaded in the "Arg. 7" core location. If no match is found, the next directory block is read and the search is repeated.

If no match is found in the directory, then the next free directory block number is loaded into core location "Arg. 7".

7.5.2 - To Read a File

After the file has been opened, bits 0 to 7 of Arg. 7 , refer to Figure 7.2, must be non zero. If these bits are zero, the file was not found and therefore an error message must be originated.

There are two modes for reading a file from dectape, the Block mode and the Continuous mode.

In the block mode, one block of dectape is read into the user buffer, starting at location Arg. 5, refer to Figure 7.2, and control is returned to the User Program.

The format for reading dectape in the Block mode is:

```
LAC - Arg. 1            /System User Program
XOR - XRDDTB            /Read Dectape in Block Mode
XOR - Arg. 4            /Facts Station Number
DAC - .+1
CAL                     /Modified
Arg. 7                  /Pointer to No. of Blocks
                        /And the First Block No.
```

Arg. 7, refer to Figure 7.2, points to the number of blocks and the FIRST block. Arg. 8 points to the present block read, which is zeroed at the start. As blocks are read from the dectape, the contents of Arg. 8 are incremented; when the difference between the contents of Arg. 8 and the starting block number is equal to the number of blocks as stored in Arg. 7, the file has been read and Arg. 8 is cleared.

The format for reading a complete file, starting at location Arg. 5, refer to Figure 7.2, is:

```
LAC - Arg. 1                 /System User Program
XOR - XRDDTA                 /Read all File from Dectape
XOR - Arg. 4                 /Facts Station Number
DAC - .+1
CAL                          /Modified
Arg. 7                       /Pointer to No. of Blocks
                             /And Starting Block Number
```

7.5.3 - To Write a File

After the File has been opened, bits 0 to 7 of Arg. 7, refer to Figure 7.3 must be zero. If these bits are non-zero, this file name is on the directory. These bits must be zeroed before a request for writing a file is initiated. When the file is closed, the previous file will be deleted.

A message to the operator could be initiated by the User Program to give him the option of defining a different name.

There are two modes for writing a file onto a dectape, the Block mode and the Continuous mode. In the Block mode, one block of dectape is written from the user buffer starting at location Arg. 5, refer to Figure 7.3, and control is returned to the User Program.

The format for writing on dectape in the Block mode is:

```
LAC - Arg. 1                 /System User Program
XOR - XWTDTB                 /Read Directory in Block Mode
XOR - Arg. 4                 /Facts Station Number
DAC - .+1
CAL                          /Modified
```

```
Arg. 7                          /Pointer to No. of Blocks
                                And the First Block No.
```

Arg. 7, refer to Figure 7.2, points to the FIRST Block where the file is to be stored. As Blocks of DATA are written, the number of blocks used are stored in bits 0 to 7 of Arg. 7 and the present block written is stored in Arg. 8.

To write in the Continuous mode, bits 0 to 7 of Arg. 7 must be updated with the number of blocks of data required to transfer. The format for writing in the Continuous mode is:

```
XOR - Arg. 1                    /System User Program
XOR - XWTDTA                    /Write on Dectape Continuous
                                 Mode
XOR - Arg. 4                    /Facts Station Number
DAC - .+1
CAL                             /Modified
Arg. 7                          /Pointer to No. of Blocks
                                /And the First Block No.
```

7.5.4 - To Close a File

After a file is WRITTEN it must be closed in order to update th- directory with the new file name. The format for closing a file is:

```
LAC - Arg. 1                    /System User Program
XOR - SCLDT                     /Close Dectape File
XOR - Arg. 4                    /Facts Station Number
DAC - +1
CAL                             /Modified
Arg. 6                          /Pointer to file name
```

The directory is searched and if the file name, as specified in Pointer Arg. 6, refer to Figure 7.2, is found, the previous file directory entry is deleted. The file name in Arg. 6 and the number of blocks and the first block of the file is stored at the end of the directory and the next free block is loaded in the first entry of the directory block.

If the System User Program is the Compiler, the first character of the file name is changed to a "T" prior to closing the file. This will identify the compiled programs in the directory.

7.6 - FACTS STATION TRANSMISSION HANDLER

There are two transmission formats defined in the FACTS Station Transmission Handler, the complete address word (CAW) and the partial address word (PAW), wordfill (WF) combination as shown in Figure 7.3.

The handler masks bits 0 and 1 of the CAW and the PAW requesting IOP4 and IOP2 and inserts them into the input/output instruction. The FACTS station number is inserted into bits 0, 1 and 2 of the CAW and PAW. Even parity is generated and placed into bit 17. The CAW or PAW is then transmitted to the FACTS BUSS via the modified I/O instruction.

If a CAW is transmitted, control returns directly to the User Program. If a PAW is transmitted, the WF is accessed. Parity is generated, followed by the transmission of the WF to the FACTS BUS . Control is then directly transferred to the User Program.

The format for transmitting a CAW or PAW, WF combination is:

```
XOR - Arg. 1                  /Operating System User Program
XOR - XTRFST                  /Transmit to Facts Station
XOR - Arg. 4                  /Facts Station Number
DAC - .+1
CAL                           /Modified
Arg. 5
```

Arg. 5 points to either the CAW or PAW, WF combination.

7.7 - A/D HANDLER

The A/D Handler programs the A/D to read the DC voltage that appears at it's input. It assumes a positive polarity and if a zero buffer is read, the polarity is switched to read a negative voltage.

The format for reading the A/D is:

```
LAC - Arg. 1                  /System User Program
XOR - XRDAD                   /Read A/D
XOR - Arg. 4                  /Facts Station Number
DAC - .+1
CAL                           /Modified
Arg. 5                        /Answer will be stored in
                              /Location Arg. 5
```

The answer is stored in location "Arg. 5". Bit 17 of Arg. 5 is on a Ø or a 1 to indicate if the voltage measured was positive or negative respectively. The 12 most significant bits represents the answer with a weighted value of 2.44 milivolts per increment.

7.8 - REAL TIME HANDLER

The Real Time Handler can process 8 users simultaneously. There are two clocks defined in the handler. One operating in increments of 100 microseconds, the other in 102.4 milliseconds.

The range of measurement is 1 msec. to 13 second increments, with a resolution of 100 microseconds and 13 seconds to 3 hours with a resolution of 102.4 milliseconds.

There are three modes defined for the measurement of Real Time. They are the Unconditional wait, the Measurement of the time between the application of the stimulus and the external event, and between two external events.

The format for requesting an Unconditional wait is:

```
LAC - Arg. 1                    /System User Program
XOR - XWAIT                     /Unconditional Wait
XOR - Arg. 4                    /Facts Station Number
DAC - .+1
CAL                             /Modified
Arg. 5                          /Pointer to time parameter
```

Arg. 5 will contain the time parameter in units of 100 microseconds, indicated by bit Ø on a Ø, or in units of 102.4 milliseconds, indicated by bit Ø on a 1.

After the time has elapsed, the user will exit from the Real Time Handler, but could stay up to an additional 30 milliseconds in a queue before regaining control, if the system is fully loaded.

In the measurement of time intervals of external events, the accuracy requirement dictates that the Real Time Handler initiate the timing. This is required to synchronize the start of the external timing with the 100 microsecond clock.

The format for requesting the timing of an external event synchronized to the application of a stimulus is:

```
LAC - Arg. 1          /System User Program
XOR - XTTOC           /Timing Synchronized to Stimulus
XOR - Arg. 4          /Facts Station Number
DAC - .+1
CAL                   /Modified
Arg. 5                /Pointer to Parameters
```

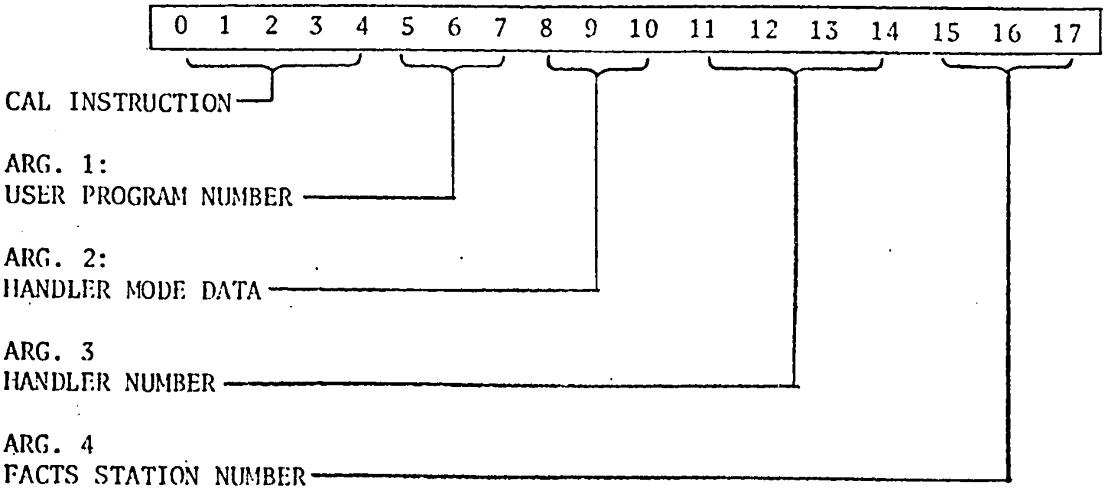
The first parameter, referred via pointer Arg. 5, is the word that is transmitted by the handler to the Facts station to apply the stimulus. When the stimulus is applied, a flag is set at the FACTS station which causes an interrupt at which time the handler starts the clock.

The second parameter specifies the maximum time wait before causing a time overflow. The answer is stored in table "XTTBL1" with reference to the FACTS station number. If the timing start flag does not occur at the FACTS station, before a 3 second interval default wait, than zero (Ø) would be stored in "XTTBL1". If the stop flag does not occur, before the specified maximum wait expires, than all ones (1's) would be stored in "XTTBL1". If the start and stop flag do occur, before the specified times are exhausted, the answer that is stored in "XTTBL1" is the difference between the elapsed or measured time and the maximum specified time wait.

The processing of the measurement of time between two external events is indentical to the processing of a time measurement between the application of a stimulus and an external event from the user point of view. There are several differences in the handler due to the hardware.

The format for requesting the timing of the internal between two external events is:

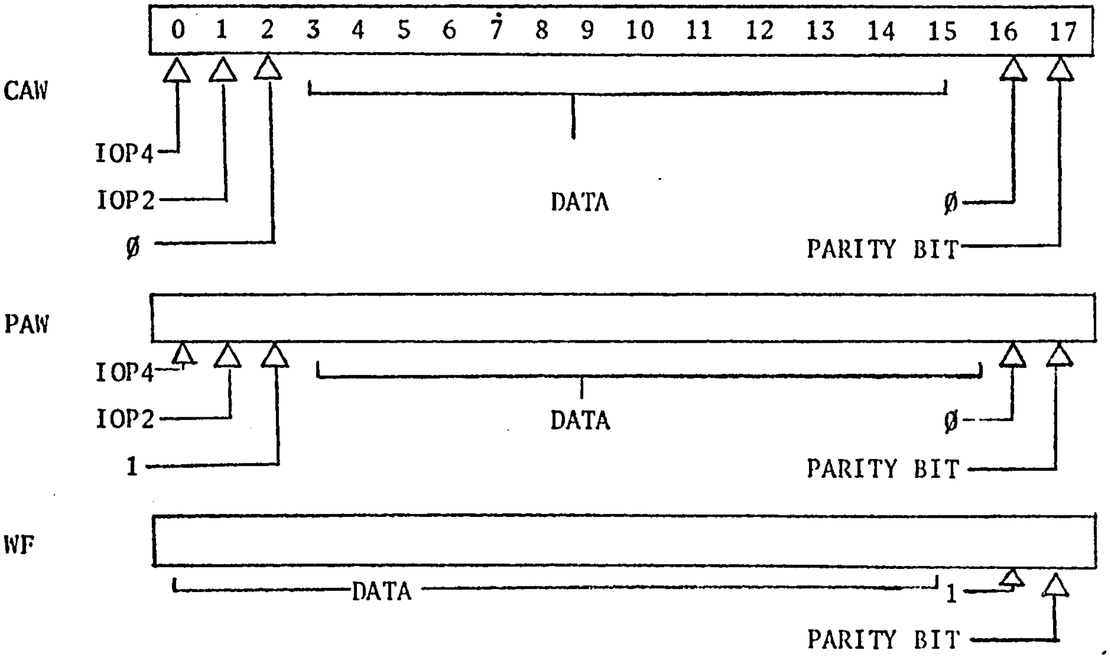
```
LAC - Arg. 1          /System User Program

XOR - XTCTC           /Timing Synchronized to First
                       Event

XOR - Arg. 4          /Facts Station Number

DAC - /+1

CAL                   /Modified

Arg. 5                /Pointer to parameters
```



I/O CALL FORMAT FIGURE 7.1

```
Arg. 6 - F I L          /File Name in 6 Bit
         E N A          /ASCII
         M E
Arg. 7 - Ø
Arg. 8 - Ø
         Arg. 5         /First Location of User Buffer
```

DECTAPE FILE NAME BUFFER FORMAT FIGURE 7.2



FACTS STATION TRANSMISSION FORMATS FIGURE 7.3

CHAPTER 8

CONCLUSIONS

The comprehensive Testing Program as defined in Chapter 6 was performed by the Author in August 1972. The results showed that the Monitor features such as proper queuing of user requests and I/O requests, priority of the Compiler over the Editor and in turn over the BTP, swapping of the lower priority program, priority of the foreground over the background, I/O handling of - the paper tape reader - Dectapes - TTY's - timing measurement - A/D measurement - FACTS Test Set Control - etc., functioned according to the requirements that the Author designed into the Time sharing software.

In addition, the features of aborting a job, the recoverable error printout initiated by a device handler and subsequent recovery by typing ↑R, and the irrecoverable error printout initiated by a device handler were verified.

In order to service more than two FACTS Test Stations an addition 8K of core memory would be required and either one additional Dectape Transport per test station or a Disk would be required to store the Test Programs.

APPENDIX 1

FACTS TEST STATION

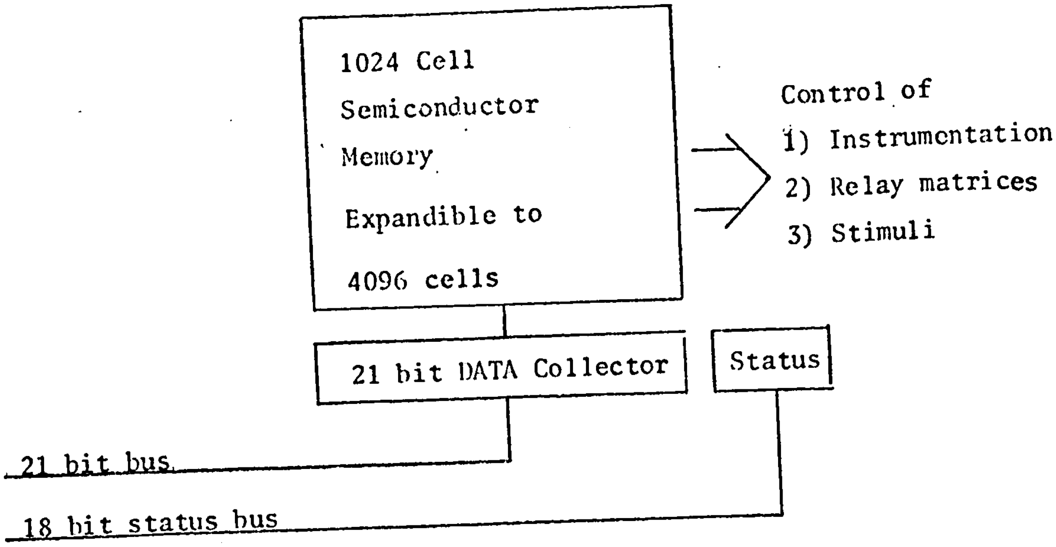
A FACTS Test Station is shown below.

It consists of a 21 bit data collector which accepts data that are generated on the 21 bit bus . If the address bits correspond to the address of the Test Station the other 18 bits are processed, as is the case with a CAW*. Parity is verified and if correct, one of the cells in the memory is switched. If a PAW*, WF* combination is accepted up to 16 cells are programmed.

The STATUS data reflects the conditions present at the FACTS station. These may include the status of instrumentation, power supplies, etc.

The generality of the FACTS station stems from the facts that whatever can be controlled by the semiconductor memory can be programmed. In the case of the FACTS 1 and 2 Test Stations, groups of memory cells were assigned to control instruments, relay matrices , and stimuli. These control bits were used as parameters in the Testran compilers. By following the rules defined in using the Testran language, the compiler inserts the corresponding parameters required to reflect what the Testran Test program specifies.

(*) Refer to Figure 7.3


1024 Cell
Semiconductor
Memory
Expandible to
4096 cells
Control of
1) Instrumentation
2) Relay matrices
3) Stimuli
21 bit DATA Collector
Status
21 bit bus
18 bit status bus

APPENDIX 2

8-BIT AND TRIMMED 6-BIT ASCII CODES

The 8-bit code and resulting 6-bit code are shown with their corresponding symbols. Certain symbol definitions had to be changed in the case of the 6-bit code to include all required symbols within the 64-symbol set.

8-BIT CODE		6-BIT CODE	
(TELETYPE INPUT/OUTPUT)		(CHARACTER STORAGE)	
SYMBOL	CODE (OCTAL)	CODE (OCTAL)	SYMBOL
A	301	01	A
B	302	02	B
C	303	03	C
D	304	04	D
E	305	05	E
F	306	06	F
G	307	07	G
H	310	10	H
I	311	11	I
J	312	12	J
K	313	13	K
L	314	14	L
M	315	15	M
N	316	16	N
o	317	17	O
P	320	20	P
Q	321	21	Q

8-BIT CODE		6-BIT CODE	
(TELETYPE INPUT/OUTPUT)		(CHARACTER STORAGE)	
SYMBOL	CODE (OCTAL)	CODE (OCTAL)	SYMBOL
R	322	22	R
S	323	23	S
T	324	24	T
U	325	25	U
V	326	26	V
W	327	27	W
X	330	30	X
Y	331	31	Y
Z	332	32	Z
0	260	60	0
1	261	61	1
2	262	62	2
3	263	63	3
4	264	64	4
5	265	65	5
6	266	66	6
7	267	67	7
8	270	70	8
9	271	71	9
!	241	41	!
"	242	42	"
#	243	43	#
$	244	44	$
%	245	45	%
&	246	46	LINE FEED

8-BIT CODE		6-BIT CODE	
(TELETYPE INPUT/OUTPUT)		(CHARACTER STORAGE)	
SYMBOL	CODE (OCTAL)	CODE (OCTAL)	SYMBOL
'	247	47	CARRIAGE RETURN
(	250	50	(
)	251	51	)
*	252	52	*
+	253	53	+
,	254	54	,
-	255	55	-
.	256	56	.
/	257	57	/
:	272	72	:
;	273	73	;
<	274	74	<
=	275	75	=
>	276	76	>
?	277	77	RUB-OUT
@	300	00	END-OF-FIELD
[	333	33	TAB
\	334	34	\
]	335	35	FORM FEED
↑	336	36	↑
←	337	37	←
SPACE	240	40	SPACE

APPENDIX 3

DECTAPE DIRECTORY ORGANIZATION

The Dectape Directory Organization is identical to that used for the SG-1 Dectrive program. The first 5 blocks of dectape are reserved for the Directory. The format of the Directory appears below:

Word Ø contains the next free block number on the dectape. A file directory entry word 1, 2, and 3, consists of a file name in 6-bit ASCII with eight characters maximum followed by at least one zero character and Word 4, which contains the number of blocks occupied by the file with the starting block number.

There are 315 decimal directory entries per dectape. If a directory block is full word Ø will contain 777777 Octal.

BITS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Word Number			
Ø		NEXT FREE BLOCK NUMBER	
1	CHAR. 1	CHAR. 2	CHAR. 3
2	CHAR. 4	CHAR. 5	CHAR. 6
3	CHAR. 7	CHAR. 8	ZERO CHAR.
4	NO. OF BLOCKS	STARTING BLOCK NUMBER	
5	CHAR. 1	CHAR. 2	CHAR. 3
6	CHAR. 4	CHAR. 5	CHAR. 6
7	CHAR. 7	CHAR. 8	ZERO CHAR.
8	NO. OF BLOCKS	STARTING BLOCK NUMBER	
9	CHAR. 1	CHAR. 2	CHAR. 3

A total of 63 decimal directory entries per block.

APPENDIX 4

MNEMONICS FOR I/O CALLS

The following list defines the mnemonics that are used for I/O Calls.

XCALLP	10	/CAL LOOP
XOPDK1	20	/OPEN FILE DISK DIRECTORY 1
XCLDK1	1020	/CLOSE FILE DISK DIRECTORY 1
XOPDK2	220	/OPEN FILE DISK DIRECTORY 2
XCLDK2	1220	/CLOSE FILE DISK DIRECTORY 2
XRRDKB	420	/READ FROM DISK 1 BLOCK
XWTDKB	1420	/WRITE ON DISK 1 BLOCK
XRRDDKA	620	/READ FROM DISK ALL PROGRAM
XWTDKA	1620	/WRITE ON DISK ALL PROGRAM
XOPDT	30	/OPEN FILE DECTAPE REF. XFACTN
XCLDT	1030	/CLOSE FILE DECTAPE REF. XFACTN
XRDDTB	230	/READ FROM DECTAPE REF. XFACTN 1 BLOCK
XRDDTA	430	/READ FROM DECTAPE ALL PROGRAM
XWTDTB	1230	/WRITE ON DECTAPE REF. XFACTN 1 BLOCK
XWTDTA	1430	/WRITE FROM DECTAPE ALL PROGRAM
XRDCRD	40	/READ CARD READER 1 CARD
XRDPT	50	/READ FROM PTR UNTIL DEL OR FULL BUFFER
XWTPT	60	/WRITE ON, PTR UNTIL BUFF EMPTY
XADVRD	70	/READ A TO D VOLTAGE
XADSRD	270	/READ A TO D SCAN VOLTAGE FACTS 1
XRPTTY	320	/READ AND PACK FROM TTY (6 BIT ASCII)
XRDCOM	520	/READ COMMAND

```
            720     /SPARE READ TTY
XPPTTY      1120    /PRINT FROM PACKED 6 BIT ASCII ON TTY
XPDTTY      1320    /PRINT FROM UNPACKED BUFFER ON TTY
XPOTTY      1520    /PRINT OCTAL
            1720    /SPARE PRINT TTY
XWAIT       130     /TIME 1 UNCONDITIONAL WAIT
XTSTOC      330     /TIME 2 TERMINATION TO CONTACT
XTCTOC      1330    /TIME 3 CONTACT TO CONTACT
XTRFST      140     /TRANSMIT TO FACTS STATION
XEND        150     /END OF PROGRAM - RELEASE API
            160     /SPARE
XUSRET      170     /USER KEYBOARD REQUEST FOR SERVICE
XUSREC      1170    /USER RECOVERY FROM HARDWARE FAULT
XOPSYS      0       /OPERATION SYSTEM USER
XTRSYS      2000    /TRANSLATOR SYSTEM USER
XEDSYS      4000    /EDITOR USER
XBKSYS      6000    /BACK TASK PROGRAM USER
XLDSYS      16000   /LOAD OR WRITE BACK PROGRAM
```

The format of the Handler numbers as specified above can be divided into two parts. The mode data represented by bits 8, 9, and 10, and the hardware number represented by bits 11, 12, 13, and 14. There can therefore be 16 devices addressed with 8 functions per device.

APPENDIX 5

MONITOR ERRORS

These error diagnostics are required to fully debug the Monitor. Once the system is fully operational, the error messages will be replaced by irrecoverable user messages.

The following lists the Monitor error messages as presently defined.

ERROR NUMBER PRINTED ON SYSTEM TTY	SOFTWARE ROUTINE WHERE ERROR OCCURED	DESCRIPTION OF ERROR
ERR Ø	XCALHD	UNDEFINED SYSTEM USER PROGRAM
ERR 1	XCALHD	API LEVEL 4 NOT ACTIVE
ERR 2	SCALHD	OP/SYS CAL BUT API 6 NOT ACTIVE
ERR 3	XCALHD	CAL BUT NO ACTIVE SOFTWARE API LEVELS
ERR 4	XSEVCL	TRANS CAL'D BUT NOT IN PROGRESS
ERR 5	XEDPRO	EDIT CAL'D BUT NOT IN PROGRESS
ERR 6	XSEVCL	API 7 ACTIVE BUT NO SYSTEM USER
ERR 7	XTTYIN	UNDEFINED TTY
ERR 10	XABUSE	CAL FROM BACKGROUND
ERR 11	XGOHND	UNDEFINED HANDLER
ERR 12	XUTTY	ILLEGAL BUSY TTY
ERR 13	XSHIFT	MORE THAN 17 SHIFTS - ILLEGAL
ERR 14	XSTORQ	OVERFLOW
ERR 15	XTIMU	ILLEGAL HANDLER
ERR 16	XCKSTX	UNIDENTIFIED ACTIVITY
ERR 17	XUSERE	IND NOT SET BUT RECOVERY REQUESTED
ERR 20	XGETSP	EDIT IN PROCESS BUT NOT I/O OF EDITOR.

ERR 21	XGETSP	BACK NOT IN PROCESS YET REQ SWAP
ERR 22	XGETSP	BACK IN PROCESS BUT NOT I/O OF BACK
ERR 23	XBKFIL	FILE NOT FOUND QUEUE
ERR 24	XABORT	ACTIVE USER, BUT ZERO SYSTEM PR POINTER
ERR 25	XPINH	COUNT ERROR, MODE BITS
ERR 26	XCALHD	ZERO CAL
ERR 27	XPTEND	SYSTEM PROGRAM NOT IDENTIFIED IN XTBL9
ERR 30	XDECOD	UNDEFINED DECTAPE FUNCTION
ERR 31	XAPIRT	DT API INTERRUPT BUT NO FLAG
ERR 32	XREADA	# OF BLOCKS NOT SPECIFIED DT
ERR 33	XNOTI	UNDEFINED TIME FUNCTION
ERR 34	XNOT1	UNDEFINED FACTS NUMBER
ERR 35	XRETIM	UNDEFINED FACTS TIME FLAG
ERR 36	XTFLG	ACTIVE BUT NO RETURN ADDRESS
ERR 37	XSETF1	A/D IS OFF
ERR 40	XDTHA2	WRONG BUFFER LOC. DECTAPE

APPENDIX 6

The Peripheral assignments to the API hardware are as follows:

CHANNEL NUMBER OCTAL	TRAP ADDRESS	DEVICE	PRIORITY LEVEL
0	40	Software channel Ø	4
1	41	Software channel 1	5
2	42	Software channel 2	6
3	43	Software channel 3	7
4	44	Dectape (TCD2)	1
7	47	Disk	1
10	50	Paper tape Reader	2
11	51	Clock Overflow	Ø
12	52	Power Fail	Ø
17	57	A/D	Ø
30	70	FACTS.time flag	Ø
31	71	FACTS TTY's	3
32	72	FACTS push buttons	3

BIBLIOGRAPHY

1. Rosen, F., Robert, "Supervisory and Monitor Systems", Computer Surveys, Volume 1, #1, March 1969.

2. Gruenberger, Fred, "The Transition to On-Line Computing", Thomas Book Company, London.

3. "I.B.M. 1800 Data Acquisition and Control System Functional Characteristics", File #1800-01, Form A26-5918-4.

4. Digital Equipment Corporation "PDP15 RSX", #15-GRQA-D.

5. Digital Equipment Corporation "PDP-9 User Handbook", #75, Chapter 7.

6. Digital Equipment Corporation "Automatic Priority Interrupt" Option KF09A, #DEC-09-15AA.

7. Digital Equipment Corporation "PDP-9 User Handbook", #75, Chapter 1.

8. Maurer, W.D., "Programming - An Introduction to Computer Languages and Techniques", Holder Day Inc., 1968.

9. Digital Equipment Corporation "PDP-9 User Handbook", #75, Chapter 5.

10. Digital Equipment Corporation "PDP-9 User Handbook", #75, Chapter 14.

11. Northern Electric Company "FACTS Dectape User Manual", #K412-2, September 1971.

12. Northern Electric Company "Testran Users Manual", #K412-3, December 1971.

13. Martin, James, "Programming Real Time Computer Systems", Prentice-Hall, 1965.

14. Martin, James, "Design of Real Time Computer Systems", Prentice-Hall, 1967.

15. Scherr, Lee, Allen, "An Analysis of Time Shared Computer Systems", Research Monograph #36, The M.I.T. Press, 1967.

16. Wilbes, M.V., "Times Sharing Computer Systems", American Elsevier Publishing Company 1968.

17. Wegner, Peter, "Programming Languages, Information Structures and Machine Organization", McGraw-Hill.

18. Laner, George, "A Real Time Multiuser Foreground, Singleuser Background System for the PDP-9 Computer", Science Centre, North American Rockwell Corporation, Paper given at spring DECUS Meeting 1970.

19. Digital Equipment Corporation, "Advanced Software System Monitors", #DEC-9A-MADO-D, Chapter 7.

20. Digital Equipment Corporation, "PDP-10 Time Sharing Handbook", Code AKW, 1970.