TIME SHARING SYSTEM FOR THE PDP-1 • YATES • REPORT ESL-R-140 • JUNE 1962 • M.I.T.

AFCRL-62-519
June 1962

SCIENTIFIC REPORT NO. 5
REPORT ESL-R-140
M. I. T. PROJECT DSR 8494
Contract AF-19(604)-6654

A TIME SHARING SYSTEM *for the* PDP-1 COMPUTER

John E. Yates

Electronic Systems Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE 39, MASSACHUSETTS

Department of Electrical Engineering

AFCRL-62-519 M. I. T. Report ESL-R-140 Copy ________

A TIME SHARING SYSTEM FOR THE PDP-1 COMPUTER

by

John E. Yates

June 1962

Prepared for

ELECTRONICS RESEARCH DIRECTORATE
AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

Project 4641
Task 464102

Scientific Report No. 5
Contract AF-19(604)-6654
M. I. T. Project DSR 8494

Prepared by

Electronic Systems Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge 39, Massachusetts

NOTICES

Requests for additional copies by Agencies of the Department of Defense, their contractors, and other Government agencies should be directed to the:

ARMED SERVICES TECHNICAL INFORMATION AGENCY
ARLINGTON HALL STATION
ARLINGTON 12, VIRGINIA

Department of Defense contractors must be established for ASTIA services or have their "need-to-know" certified by the cognizant military agency of their project or contract.

All other persons and organizations should apply to the:

U. S. DEPARTMENT OF COMMERCE
OFFICE OF TECHNICAL SERVICES
WASHINGTON 25, D. C.

ABSTRACT

A system for time sharing a PDP-1 digital computer with seven typewriters, two paper tape punches, two paper tape readers and two CRT displays is described. The additional hardware required for the system and the modifications required to a basic PDP-1 are described and a program is presented to handle the monitor or "executive" functions of the system. A system using two typewriters, one punch, one reader and one display based on this design is currently being installed at M.I.T.

ACKNOWLEDGMENT

This report is based upon a thesis submitted May 21, 1962 by John E. Yates in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering at the Massachusetts Institute of Technology.

The author wishes to express his appreciation to Professor Jack B. Dennis for his counsel and guidance in the definition and completion of this research.

This work was sponsored by the Air Force Cambridge Research Laboratories, Office of Aerospace Research under Contract No. AF-19(604)-6654.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Recently a PDP-1 digital computer has been donated to the Electrical Engineering Department at M. I. T. by the Digital Equipment Corporation for educational purposes. In an effort to make the computer available to more students, and at the same time more efficient, it is desired to install a time sharing system on this computer. A time sharing system, as used here, is one where several users may simultaneously use the computer with only a small degradation in performance (speed) such that each user thinks he has a complete computer. As such, a system has been designed for a maximum of seven users, and is being built at present for two users.

A. REASONS FOR TIME SHARING

There are several reasons for having a time sharing system on such a computer; (1) better man-machine communication; (2) more efficient use of input-output equipment, and (3) effectively more computers are available for use. All of these serve the purpose of providing better service to the user, which is the underlying motive of such a system.

1. Man-Machine Communication

A recent report[1] to the Computation Center at M. I. T. recommended that a time shared computer would provide the best service to users. One reason for this would be what is known as better man-machine communications. The program debugging process can be made much more efficient if the programmer can be present at the computer console and correct any mistakes as they are pointed out by the error indications. Waiting for results to come back, as on a large computer, or just waiting for another try at a computer can waste a lot of a programmer's time. A small computer, such as the PDP-1, is usually cheap enough that they can be run with the programmer present, in fact they are usually run open shop - i. e. by the programmer himself, and the problem of turn around time are not so critical. A large computer, as the IBM 7090, is usually operated closed shop and the turn around time may be a day or more.

[1] Teager, H. (chairman), "Report of the Long Range Computation Study Group," MIT Computation Center, April 1961.

Any time sharing system, therefore, should strive to maintain this close relationship found in the small computer or to introduce this into a large system. A closed shop operation usually results in less time actually spent on the computer by each program and therefore smaller costs than an open shop operation, but the time sharing system can offset this difference in costs of the two types of operation by allowing many users to simultaneously share the computer.

2. Efficient Use of I/O Equipment

A problem common to all computers is that input-output equipment usually has a rate of transfer much lower than the cycle time of the computer. Thus the computer is severely limited during input-output operations. The very large computers have what is effectively a smaller computer just to handle this feature alone so the main computer may go on computing. Smaller computers usually have to wait for the in-out transfer to take place greatly decreasing their effective speed of operation.

A time sharing system will make the in-out process more efficient by allowing the computer to keep several IO devices active at once. For example, if a computer, thru some system, could keep seven programs active in memory, and seven sets of output devices, it could switch control to one of the other programs while waiting for a completion of some one in-out transfer.

Obviously there is some "overhead" or bookkeeping required for such a system. This may be great enough that for the faster in-out devices, such as CRT, mag tape and possibly paper tape readers, the overhead is a significant portion of the time required to service the device, defined as overhead plus device time. But for slower devices, such as paper tape punches and typewriters, the device time becomes the dominate factor in the service time. The following table assumes an overhead time of 500 microseconds.

	Transfer Time		Overhead %
CRT	50	us	1000
mag tape	66		900
paper tape reader	3300		16.0
paper tape punch	16,000		2.5
typewriter	100,000		0.5

The slower devices can be very advantageously time shared with a minimum of lost time.

3. More Computer Use Available

In a time shared system assuming the slower in-out device one effectively has a number of computers equal to the number of time shared consoles, minus the time spent on "overhead." Considering that an additional console is much cheaper than an additional computer, a time shared system gives both more efficient operation and increased computing capacity.

B. PHILOSOPHY OF TIME SHARING

Several items must be kept in mind when designing a time-sharing system. One might call these the different philosophies of time sharing.

1. Methods of Storage

As was mentioned earlier, in a time sharing system several programs are kept active in the computer at once. There are two basic different ways of accomplishing this. One is applicable to computers having a large memory and programs which usually occupy a small portion of the memory, and the other is applicable to all machines regardless of memory or program size.

In the first method, several programs are stored simultaneously in different sections of memory. A small section of memory is set aside permanently containing the necessary program to accomplish the "overhead" work of coordinating the in-out transfers, and determining which program to serve next. This routine is called the Executive Routine. Here the number of users is limited by the size of memory and the size of program. Larger programs require larger memory, or dictate fewer users.

The second method requires some sort of auxiliary storage for programs, such as a drum or disk file. This type is necessary with small memories or large programs. In it we still have the Executive Routine, ER, but now the program occupies the rest of memory. When it has been determined to transfer control to another program, the program in memory must be first stored on the drum or file, and a new program brought in. This method of course takes longer to transfer control-the drum swap may take 33 milliseconds - and necessitates the executive routine being longer to handle the swap mechanism as well. The effective size of core under the time sharing system is therefore cut down still further.

On the PDP-1, core size is only 4096-18 bit words. This is too small to share among several programs, so the second system using auxiliary storage is used. The swap time represent about 8 % of a users quanta -- the time allotted for him to remain active in memory. Thus it is felt that this method does not work under too great a disadvantage. The ER takes about 400 registers in the two console system and may be around 700 for the full 7 consoles.

2. Operation Under Time Share System

One would like to design the system such that the machine, as the programmer and operator see it, is exactly the same under time shared and non time-shared operation. This would allow programs written on similar non time-shared machines at other installations to be run on the time shared computer and vice versa. The programmer would not have to learn any new or non-standard conventions beyond the manufacturer's programming manual. In fact he need not even know it is time shared.

This of course is the ideal and can not fully be met in practice without an exceptional amount of extra equipment and programming. The overriding philosophy throughout the design of this system is to make it appear as close to a non time-shared machine as is practical. In some cases this has necessitated a few compromises in the basic design between a desired better operation and the desire for compatibility with other machines. In a very few cases instructions have been added to the order code of the computer which are available to the user of the time shared system.

Another idea is to keep the computer running at all times. Instructions which cause the machine to stop are outlawed, and it is made impossible for the user to attempt manual control thru the console switches.

3. Hardware or Program

Several of the operations necessary to accomplish the "overhead" work may be accomplished either by program or by hardware. Where these could be done with hardware fairly conveniently and not too expensively, hardware was used. If the hardware becomes complicated just to save an instruction or two in the executive routine, the programming method was resorted to. Of course, one tries to keep the executive routine as small as possible since the area required by it is taken from each program used on the system.

CHAPTER II

METHODS OF OPERATION

This section describes the time sharing system from a functional or programming standpoint. But first a little must be known about the operation of the PDP-1 in normal non time-shared operation.

A. PROGRAMMING THE PDP-1

The PDP-1 is a 18 bit per word, 4,096 word digital computer. The first five bits, 0-4, make up the instruction code; bit 5 is the indirect address or defer bit for instructions that can have deferred addressing; and bits 6-17 make up the address for memory reference.

The machine operates using fixed point binary arithmetic. Negative numbers are represented by the ONES compliment of the positive numbers. Bit 0 is the sign bit which is ZERO for positive numbers. Bits 1-17 are magnitude bits.

1. Instruction Order Code

The five instruction code bits allow for 32 possible codes, numbered 00 thru 76 octal, using the even numbers. Appendix A contains a list of these. These fall into two classes, memory reference instructions and augmented instructions. The memory reference instructions require two memory cycles, except for the jump instruction, one to get the instruction, and a second for the data specified by the address section of the instruction. If bit 5 is a ONE, the data is taken not from the address specified by the instruction, but from the address specified by the word which is addressed in the instruction -- thus deferred addressing. There is no limit to the number of levels of deferred addressing allowed. The augmented instructions use bits 5-17 to specify variations in the basic instruction. No indirect addressing is allowed with these instructions.

The order codes can be divided in another fashion depending on the time required for operation. The basic cycle of the PDP is 5 microseconds with 12 unevenly spaced time pulses TP_0-TP_{10}. All of the memory reference instructions except JMP and JSP require two cycles for execution and are called two cycle commands. The augmented instruction plus JMP and JSP are one cycle commands. These latter have instruction codes in the 60's and 70's making bits 0 and 1 ONES.

2. Machine Registers

The machine registers in the PDP include the accumulator (AC), in-out register (IO), program counter (PC), instruction register (IR), memory buffer register (MB), memory address register (MA), and six program flags (PF). These are indicated with their control lines in Fig. 2. 1. The AC and IO perform the arithmetic operations. The PC and MA perform address bookkeeping and modification. The MB holds information going to and from memory. The program flags serve as independent switches, synchronizes, or program storage, which can be tested by a complete set of instructions.

3. Console Operations

The console of the PDP-1 contains several sets of switches, lights, and control buttons. The lights indicate the contents of the IO, AC, MB, IR, MA, PC, PF, sense switches and various control states of the machine. Two sets of 18 switches are the TEST WORD and FIELD-ADDRESS switches. The Test Word may be sampled by the instruction LAT in a program, or may be manually deposited at an address set on the Address Switches. Another set of 6 Sense Switches exist for program use which can be tested by a complete set of skip instructions.

The control buttons include:

START
Transfer control to the address specified in the Address switches and continue.

STOP
Stop operation at the end of the current memory cycle. The registers are not changed.

CONTINUE
Continue operation starting at the present location of the PC.

EXAMINE
Display the contents of the location specified by the Address switches in the AC. The previous contents of the AC are lost.

DEPOSIT
Deposit the Test Word in the location specified by the Address Switches.

READ IN
The paper tape reader will read binary paper tape and place in memory according to a Read-In format.

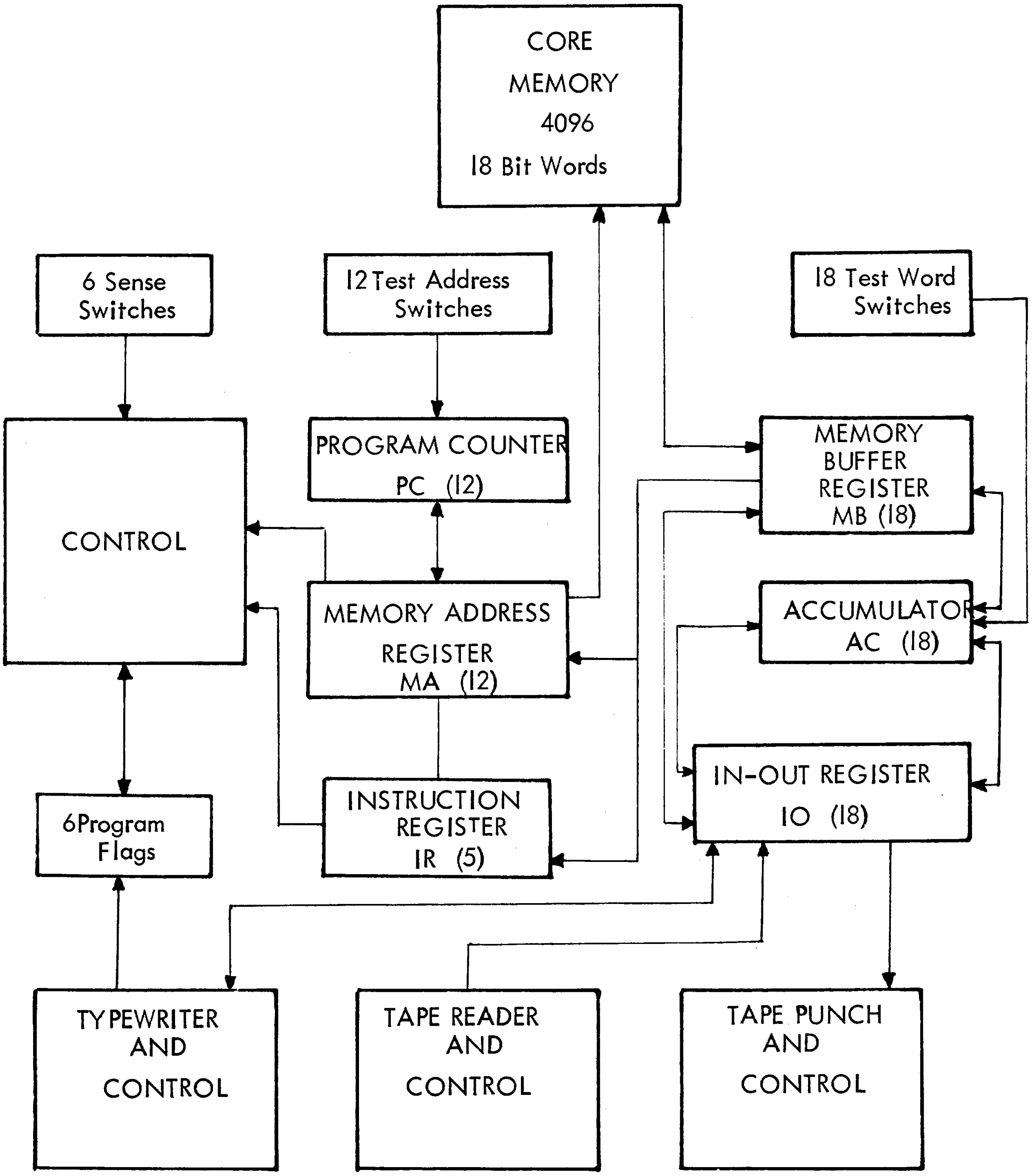


FIG. 2.1 - PDP-1 SYSTEM BLOCK DIAGRAM

SINGLE STEP	The computer will stop at the end of each memory cycle if this switch is on. Pushing Continue will step one cycle at a time.
SINGLE INST.	The computer will stop at the end of each complete instruction. Continue will step one instruction at a time.

4. Input-Output

The PDP-1 is designed to operate a variety of input-output devices. All such in-out operations are performed thru the IO register using the IOT command. The IOT command produces pulses on time pulses TP_7 and TP_{10} to equipment specified by the address section of the command. Thus on an IOT 03, TP_7 would clear the type buffer and TP_{10} transfers the contents of IO_{12-17} to the type buffer and initiates action in the typewriter.

The normal IOT commands are listed in Appendix A. Several of the combinations are designated by DEC for specific equipment or functions. At M. I. T. the class IOT 1X has been set aside for use as private IOT's by users. Here he can connect any special equipment and operate it by giving the proper IOT. On all commands the class may be further decoded using bits 7-11. Thus in IOT XX03 the middle bits could be used to indicate which one of several typewriters transfer. If bit 5 is a ONE, the computer will halt and wait for the completion pulse from the device being activated. When this is received, the computer continues the normal instruction sequence. Bit 6 determines whether or not a completion pulse will be received from the device. When different from bit 5, a completion pulse will be received; if the same, no completion will be received.

If a in-out command is given without a wait, computation may continue while the IOT occurs. However, if instructions further along in the program assume data from the transfer, or assumes the transfer to be finished, the sequence of instructions must include the IOT instruction 730000 which does nothing but wait for the completion pulse. This instruction must be given before the safe minimum time of the device. This is the known minimum time before completion after the command is given.

5. Sequence Break System

The PDP-1 normally comes equipped with a single channel sequence break system (SBS). On receipt of a pulse from an external devices, such as a completion pulse, event pulse, etc., the computer will interrupt automatically and go to a particular program sequence associated with the external device. This sequence normally processes the information received, then returns to the main program.

The interrupt may occur at the end of an instruction (cycle zero for 1 cycle instructions or cycle one for 2 cycle instructions), or it may interrupt in the middle of a two cycle instruction under certain conditions (such as an indirect chain of instructions). Upon the interrupt, the C(AC) is stored in location 0, C(PC) is stored in location 1, C(IO) is stored in location 2, and control is transfered to location 3. This may be the start of the special sequence, or a JMP to the sequence. After the sequence is finished, the break is terminated by doing the sequence of instructions

```
lac     0
lio     2
jmp  i  1
```

The last instruction automatically terminates the break.

An optional feature is a sixteen channel SBS. The sixteen channels are arranged in a priority chain with channel zero having highest priority. In this system, rather than storing in locations 0, 1, 2 and 3, the register contents are stored in location 4n, 4n+1, 4n+2 and 4n+3 where n is the channel number. Storing the register contents in different locations for different channels allows a break-on-break operation where a lower priority request never interrupts a high priority request.

B. BASIC DESCRIPTION OF THE TIME SHARE SYSTEM

The programming for the TS system consists of two parts, the executive routine and the administrative routine. The executive routine is a permanent part of core memory which will handle the needs of the time sharing system on a second-to-second basis. It will handle the so-called instruction traps and time-out interrupts. The administrative routine is a separate program brought into memory on request to perform such jobs as: assignment of equipment; regulation of memory protection; providing services such as an assembler, debugging routines, editing programs, error indication for illegal instructions; and other miscellaneous jobs.

In brief the time sharing system works like this. There are seven users consoles consisting of typewriters, sense switches, and various other lights and switches which are served in turn by the executive routine. In addition there are two paper tape punches, two CRT displays, and two paper tape readers that are shared by all users but are assigned to a particular user at a given time by a request to the administrative routine.

Assume several users are using the computer. A particular program is in core and is being executed. Since we do not wish the computer to stop due to one user's errors, and thus keep others from executing, certain provisions must be made. All halt instructions, illegal operation codes, requests for manual run, and illegal instruction cause a trap to the executive routine (ER). Similarly certain IOT commands must trap as the program does not know if the equipment has been assigned to it, or which one to address if one has been assigned. The ER then executes the command using the correct assignment, or put out an error indication thru the administrative routine.

A program may well compute or require characters faster than the I/O equipment can take care of or supply them. Normally the computer waits in an in-out halt for the completion pulse before processing the next character. Under the TS system it goes to another program while waiting. For maximum efficiency several characters are computed at once and stored in a buffer in the ER. Then the next program is brought in. At frequent intervals a time out interrupt occurs where in control is momentarily transferred to the ER. Here one character is taken from each buffer and transmitted, if the I/O device is ready to accept. If not, it is skipped this time. Control then returns to the program in core. When a certain maximum time has elapsed, or if the ER buffer becomes full, or if the program runs into an error, the program is dismissed and another brought in. In this way no time is wasted and each user's program is in memory often enough that the user thinks he has the computer to himself.

C. REGISTERS ADDED IN THE TS SYSTEM

Various registers have been installed to make the programming easier and to keep track of equipment assignments. These include; the Print Status flip flops, PS_n; the lock flip flops, LK_n; the Console Flag flip flops

CF_n; an 18 bit Assignment Register, AR; and a Console Number register CNR. These will be described in the sections below.

1. Print Status

Separate flip flops numbered PS_1 thru PS_9 are associated with each console (typewriter) and punch in the system. (Note the punches are considered as consoles 8 and 9 in this scheme.) They are turned on and off with the instructions PSO and PSF which are two of many added for the TS system. When the command is given, the flip flop corresponding to the console number held in the CNR is activated. PS_n being on indicates the computer (ER buffer) has characters to type or punch out for console n. If it is OFF no action needs to be taken.

2. Lock

Flip flops numbered LK_1 thru LK_9 likewise exist and are operated by the instructions LKY and ULK. In the case of the typewriters LK being ON indicates the computer will not accept characters typed in on the typewriter. In fact, if LK is a ONE, the keyboard of the typewriter is locked, thus its name. Whenever the computer is expecting type in, as after being notified of a TYI, it unlocks the keyboard.

3. Console Flag

The console flag flip flops CF_1 thru CF_9 indicate whether further service is needed by the n^{th} console. If CF_n is a ONE, further attention is required on console n. For flags one thru seven, CN_n is turned ON if PS_n is OFF and a typewriter key is struck, or if PS_n is ON and the typewriter is finished with the previous character. For flags eight and nine, CN_n is turned ON if PS_n is ON and the punch is finished with the previous character.

4. Assignment Register

The Assignment Register, AR, is an 18 bit register which holds various pieces of information pertaining to the user currently in memory. The register is reset every time a new user is brought into core. The assignment of the bits is as follows:

BITS	FUNCTION
0-2	Console-in-Memory, CIM Binary equivalent of console number
3-5	External IOT channel assigned to user (in binary)
6, 7	Paper tape reader assignment - one bit for each reader
8-14	Memory protection
15	Test word switch assignment for use with LAT
16, 17	Display assignment - one bit for each CRT

The CIM normally takes on the values 1 thru 7 for the seven user's consoles. In certain special applications no user is present (as in real time control) and this is represented by a CIM of 0. This register is used to control which of the eight sets of 6 sense switches are to be considered by the program in core. The set on the computer control panel is used with a CIM of 0. The register also indicates by means of a light which user is active in core at a given time.

An external IOT channel assignment is necessary since each user has written his program as if he had the computer to himself. It might turn out that several user's wrote their programs intending to use the same private IOT command. To remedy this, seven channels (0 indicating no assignment) have been set up and are gated into a OR circuit by this assignment. When the user connects his equipment to the computer, he informs the administrative routine (which sets up the AR) of the number of the channel connected to.

The paper tape reader, test word switch and display assignments can be described together. If the bit is a ONE, the commands referring to this device are legal and is executed by the program. If it is a ZERO all commands referring to the device are considered illegal and a trap is made to the ER. The administrative routine prints out an error indication and the user is not allowed to return until he acknowledges and makes a correction, or waits until the device is free for his use.

Memory protection is a shield to prevent the ER from being entered by errors in the users program, such as jumps to , or deposits in, the ER.

Bits 8-14 of the AR are compared against bits 8-14 of the memory buffer. If the MB is greater than the AR, a trap is started in the same manner as for unassigned equipment. The ER is to be situated at the top of memory, therefore it will surely be in the 6000's and/or 7000's octal. In the comparison bits 6 and 7 of the MB are checked against ONES, and bits 15-17 are ignored. This means the variable boundary of protection is in the range of 6000 ± 7_o to 7777, thus a maximum of 7 registers may be "wasted" at the bottom of the ER. The comparison is fairly complicated, and it was felt that this degree of "coarseness" was not detrimental to performance.

5. Console Number Register

The console number register is a four bit register for storing the console number n. This number is used to clear and set the PS and LK flip flops upon the proper command and is used in the ER in various places.

There are three ways provided for setting the CNR. On an IOT instruction trap, the executive routine must set the CNR for use in the PS and LK instructions. This is accomplished in one of two ways: the command Load Console Number, LCN, sets the CNR from bits 0-3 of the I/O; the command Set Console Number, SCN, sets the CNR from bits 0-3 of the AR, which were the console-in-memory indication. Thus SCN can be used to set the CNR in one cycle on a trap to do with the typewriters. On a punch trap, the ER must first calculate the correct number, place it in the I/O, and then use LCN. This is because the punches can be assigned to any user and therefore the punch use bears no relation to the console-in-memory.

The third way of setting the CNR is used during the time out trap while the ER is processing characters. The outputs of the CNR go into a rotating priority chain. This assures that all consoles will be serviced with equal opportunity. The last console served is placed lowest in the chain. The two punches are placed ahead of the rotating chain on an alternating basis. This is due to the fact that the punches punch at a rate of 63 characters/second, or 16 milliseconds per character, while the typewriters take ≈ 100 milliseconds per character. If the punches are not kept fully supplied with characters, they drop out of sync and tend to chatter, which is very hard on them mechanically. The CF with the highest priority is encoded into a 4 bit binary number for later use, in setting the CNR.

The command Read Console Number, RCN, takes this encoded number and performs four functions with it; (1) it sets the CNR for use with the PS and LK flip flops; (2) it reads into bits 0-3 of the I/O after clearing the I/O, (3) it resets the rotating priority chain if n was between 1 and 7 and (4) it resets the CF_n just read.

D. INSTRUCTIONS FOR THE TIME SHARING SYSTEM

The instructions added to the machine to accomplish the programming for the time sharing system are called the executive instructions and are given the instruction code IOT 77 = EXC. They are further decoded within this class using bits 7-11 as was indicated earlier - thus EXC XX00. The instructions added and their codes are listed in Appendix B. Several of these have already been described - PSO, PSF, LKY, ULK, RCN, LCN, LAR and SCN. In addition there are four skip commands, six check status commands, four user commands, four SBS commands and a few other miscellaneous commands.

1. User Commands

Four commands have been installed to make it easier for the user to communicate with the time sharing system. These are legal to use in his program - BPT, ARQ, HNG and UHG. The command BPT is used to indicate a break point is request at the location given by the BPT. It is usually inserted by a debugging routine or administrative routine. The only action it causes is a trap when encountered. ARQ is for automatic request of equipment assignment. The user may request equipment in two ways, a typed request to the administrative routine and thru the use of ARQ, if he knows he will be operating under the TS system beforehand. When ARQ is encountered, the trap brings in the administrative routine which assigns equipment according to requests indicated by specific bits in the AC or IO.

The commands HNG and UHG are hang and unhang instructions. In certain applications the user wishes to remain in a tight program loop waiting for some event to happen, such as a break on the SBS. Since there are many ways of programming this "limbo" loop, it is not possible for the TS system to detect that the user's program is waiting for an external event. Thus the program could consume many units of computation without producing useful work. The HNG instruction is provided so that this possibility may be avoided. This sets an indicator and dismisses the user. The next time his turn comes, the ER looks at this indicator (HG flip flop). If it is on, he is not brought back unless the event has happened. In this case the break occurs immediately upon returning control from the ER and the break sequence is entered. This sequence must include the instruction UHG to reset the HG flip flop. UHG does not cause a trap and is the only EXC command that does not.

2. Skip Commands

Four commands have been added similar to the PDP-1 regular skip class commands to allow the ER to sense certain conditions. The command SPS causes the PC to be advanced 1 location if the PS flip flop is ON. SLK causes a skip if LK is ON, and SHG skips if HG is ON. The command SDY causes a skip if a display has been assigned to the user in core and the user is holding a Display Button on his console. This is used to allow him more than the usual amount of time in memory if he wishes to observe a display pattern.

3. Check Status

The standard IOT command CKS allows the user to interrogate several external devices as to whether or not certain conditions exist. These include typewriter busy or stuck, punch busy, etc. In the time sharing system its not that easy. First we must determine which punch or typewriter. Then it is only considered busy, as far as the user is concerned, if the buffers are full or empty, not whether the device is actually working. If the buffer is not full, the user's program may "type out" characters to it even though the typewriter is actually busy.

Three flip flops are provided for this purpose with six commands to turn them ON and OFF. PUN and PUF indicate punch busy or not busy (buffer full), and TON and TOF do likewise for the typewriter on type out. TIN and TIF indicate buffer not empty and empty (character typed in and not received by the program thru a TYI).

4. Miscellaneous Commands

Two commands are provided to save the states of the program flags and HG flip flop when a user is dismissed and to restore them when he is returned. These are RPF, read into the IO, and LPF, load from the IO. As with all EXC commands requiring transferal of information, the information must go thru the IO. The IO is loaded from or deposited in memory using LIO and DIO. This is because to store directly in memory requires the normal two memory cycles taking a 2-cycle op code, of which there are not enough spares available on the PDP-1. The present method makes all EXC commands one-cycle commands which can be decoded from the simple command IOT 77.

If during the operation of his program the user notes that it is in an unwanted loop, or giving wrong information, etc -- in general operating incorrectly -- he has no way of stopping it for debugging. All Control Buttons on the main computer console are disabled when the computer is operating in the TS mode to prevent manual operation of the computer which would deprive other users of computer time. A button and a switch have been provided on the user's console for this purpose. The command RBU reads the switches and buttons from the n consoles into the IO, with BU_{1-7} going into IO_{0-6} and SW_{1-7} going into IO_{9-15}.

The switch indicates that this console wishes to be considered as a user. The ER reads the switches frequently, thus noting a condition OFF → ON. When this is discovered the user is put on the active list and the administrative routine is brought in to handle his request. If an error is discovered during his program, he may push a button which signals the ER to transfer his control to another location. This may be a debug routine such as DDT or one contained in his program, or it may be a signal to bring back the administrative routine. If his program is hopelessly lost he may have ruined his own debug routine, in which case his only recourse is to turn his switch OFF and back ON. The ER interperates this as a new user request, but he can tell the administrative routine differently.

5. SBS System

The last four commands pertain to the 16 channel SBS and are not installed in the present TS system. It is anticipated the SBS will work something like this. The ER will contain a table of 16 locations swapped on a dismissal. When a break occurs on channel n, location n will set the MB telling the system where to start depositing the AC, PC and IO. Location n is also stored in a 12 bit Break Address Register, BAR. This accomplishes two things -- it allows equipment breaking on channel n to be transformed to look like a break on channel m, and it relieves the restriction of the first 100_8 registers onthe SBS system making the location of stored registers completely arbitrary. If several users write their programs in terms of channel m, but must connect their equipment to channels n-1, n, n+1, etc. this funnels the requests to the correct location, since each user of the SBS has his own SBS assignment table.

To properly dismiss a channel n now, we can no longer jump deferred thru location 4m+1 since this bears no relation to the actual channel n. Therefore all jump i's are compared to the BAR. When agreement if found, the channel ON at that time is dismissed and the BAR is reset to the next lower priority channel, it there is one.

The commands for the SBS are LBC, RWB, BCN and BCF. All events from outside are stored in the 16 Waiting Break flip flops. The outputs of these normally go directly to the priority chain. However, now we only want the priority chain to consider the channels to which the user in core has his equipment connected. The command LBC sets flip flops which control gates between the Waiting Break flip flops and the priority chain. This is set from a register in the ER which is set up by the administrative routine.

The command RWB reads the Waiting Break flip flop for ER action. It is necessary to check these since a break may have arrived on a channel assigned to one of the "console zero" users. This is the manner in which this external device signals that attention is desired. The normal PDP-1 commands of enter and leave sequence break mode, ESM and LSM, must not turn on or off the entire SBS system when given by a user, but only the channels assigned to him. The computer must always work in sequence mode. Two ways could be provided to do this--either automatically through hardware based on the Console-in-Memory number or by two commands BCN and BCF. BCN will turn ON the channels indicated in the IO and BCF will turn them OFF. These would be used by the programmer instead of ESM and LSM. However this requires special programming for the TS system.

E. INSTRUCTION TRAPS

As discussed earlier, certain instructions cause traps to the ER. These include certain IOT commands, illegal commands, and unassigned equipment commands. All traps have a unique trap number, TN, associated with them, which is put into the AC on a trap for programming convenience. A table showing the traps and their trap numbers appears in Table 2-1. The Trap Number Register, TN will accomodate 40 traps (octal), but only 0 thru 14 (octal) are included in a system having only the one channel SBS.

When a trap is made the AC, PC, and IO are saved in the same manner as in the SBS, the execution of the instruction which caused the trap is inhibited, and control is transferred to location 7003. The ER then determines whether this command was legal and executes it if it was.

1. IOT Commands

The IOT commands which cause automatic traps to the executive routine are those referring to the punch, typewriter, drum, and the SBS if the 16 channel system is installed. The typewriter and punch are slow devices and we do not wish the computer to have to stop when executing a command pertaining to these. When a TYO, PPA and PPB is given the trap is made, the ER stores the character in a buffer if the buffer is not already full, and control is returned to the program or it is dismissed if the buffer is full. On a TYI, the trap is made, the ER extracts a character from the buffer, and control returns to the program. If the buffer were empty, the program would be dismissed.

The drum and SBS (TN 7-12, 15-22) are high speed operations, but a trap must be made because of assignment problems. In the same manner as the external IO assignments, the user may have written his program with reference to one drum field or break channel, but this may be already used by another user. The ER will translate a request for a field or channel into a command for the field or channel assigned to this user. In the case of the drum, it must also change the arguments of the commands to allow for the extra time spent in the ER.

2. Illegal Commands

Commands considered illegal for the user are those which stop the computer; store in, or jump to, protected memory; refer to unassigned equipment; have illegal operation codes; and those which are built into the machine to permit operation of the time sharing system. If he uses any of these the trap is made and appropriate action is taken by the executive and administrative routines.

		TN_b							
		0	1	2	3	4	5	6	7
TN_a	0	TIME OUT	*	HNG	TYO	TYI	PPA	PPB	DIA
	1	DBA	DCC	DRA	BPT	ARQ	DSC	ASC	ISB
	2	CSB	LSM	ESM					
	3								

$TN = TN_a TN_b$

* ILLEGAL PDP-1 OP CODE (00,14,36,74)
ILLEGAL MEMORY REFERENCE
HLT
MANUAL RUN
IOT 77 · BPT · ARQ · HNG · UHG
UNASSIGNED READER, DISPLAY or
TEST WORD SWITCHES

TABLE 2.1 - TRAP NUMBERS

Trap numbers 2, 13 and 14 are those EXC commands which may be in the user's program and which cause nothing to happen except a trap so the ER may take the appropriate action as described earlier. Trap number 1 contains all the other illegal commands. These include the command HLT, all commands with illegal operation codes (00, 14, 36 and 74), the illegal references to memory, requests for manual run (single step or single instruction), all EXC commands except BPT, ARQ, HNG and UHG, and all commands pertaining to equipment which has not been assigned to the user, such as reader, display and test word switches.

F. TIME OUT INTERRUPTS

Time out interrupts are the mechanism by which the ER actually does its character processing to and from the external equipment. A time out interrupt is handled in the same manner as an instruction trap and is given trap number 0. In the ER the console flags CF are looked at and the one with the highest priority is serviced. Then the next highest is serviced, and so on until no more are left, at which time control returns to the user's program. This in no way affects his program; it is effectively just a stop of a small duration.

The requirement that the punches must be kept running if they have more than one character dictates the maximum time between successive time out interrupts. The character time on the punch is 16 milliseconds. To be on the safe side it was thought there should be between one and two interrupts during this period. A convenient source of time pulses is the next to the most significant bit of the drum counter. The drum takes 33 1/3 milliseconds per revolution, and this will change state every 1/4 revolution or 8 1/3 milliseconds. This could also be used as a convenient means of synchronizing the ER to the drum for swaps.

CHAPTER III

LOGICAL DESIGN

The logical design of the electronics for the time sharing system is, for the most part, fairly straight forward. The entire design is made using the DEC logic conventions (see Appendix D) and it is constructed using DEC 5 megacycle and 500 kilocycle system module plug in units. For the most part it is contained in two additional bays which are attached to the present PDP-1 computer. The basic computer consists of four bays numbered 11, 1, 2 and 3. The two additional bays are numbered 4 and 5. (Figure 3. 1) The magnetic drum is a separate system occupying two additional bays which is connected by cables. The system has been designed in such a way as to have the minimum amount of modifications to the PDP-1 as possible. These mostly consist of buffers for certain signals and disabling functions. All other equipment is located in bay 4.

All hardware added affects functions which take place on cycle ZERO of the two machine cycles, or during one of the special cycles. The flow charts for these are shown in Figs. 3. 2 and 3. 4. The changes necessitated by the T-S system are shown in Figs. 3. 3 and 3. 4. With reference to these flow charts, the sequence of machine operation can be explained for the added instructions and for the old instructions under the T-S system. The main registers and features of the T-S system are diagrammed in Fig. 3. 5, along with their lines of communication. The purposes for most of these were described in the last chapter. The more detailed drawings of the actual circuitry may be found in Appendix D.

A. CYCLE ZERO OPERATION

From Figs. 3. 2 and 3. 3, it may be seen that when the T-S switch is OFF operation proceeds as in a normal PDP-1. However, when in the T-S mode several new instructions are possible (the EXC class) and several other instructions are handled differently (the instructions that trap). There are two modes within the T-S mode -- the executive mode (EM^1) and the non-executive mode (EM^o). The executive mode is ON whenever control is in the executive routine. When in the executive mode, all instructions, including the ones that normally trap, are legal and the EXC class of instructions is

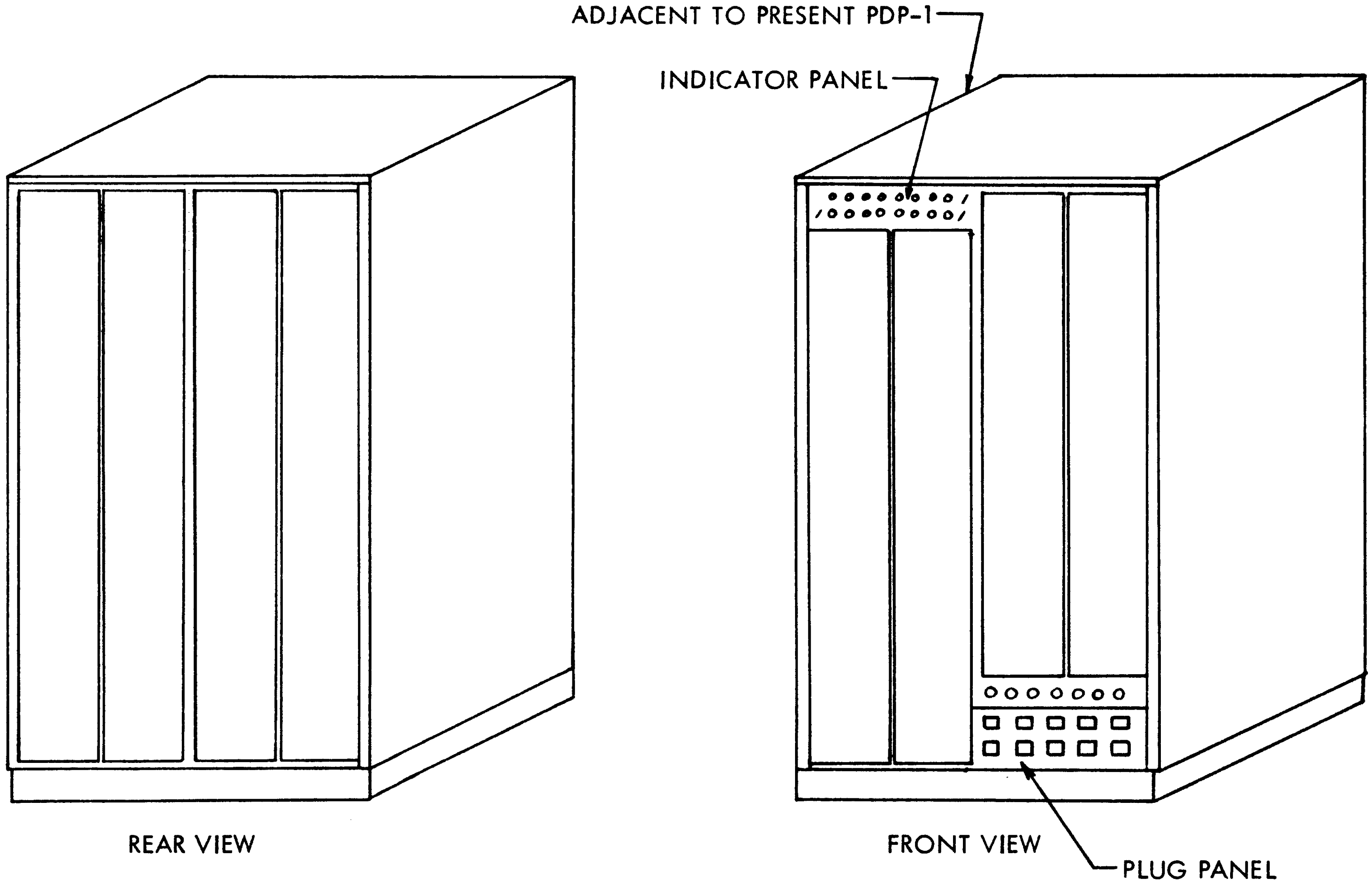


FIG. 3.1 - PHYSICAL LAYOUT - ADDITIONAL BAYS

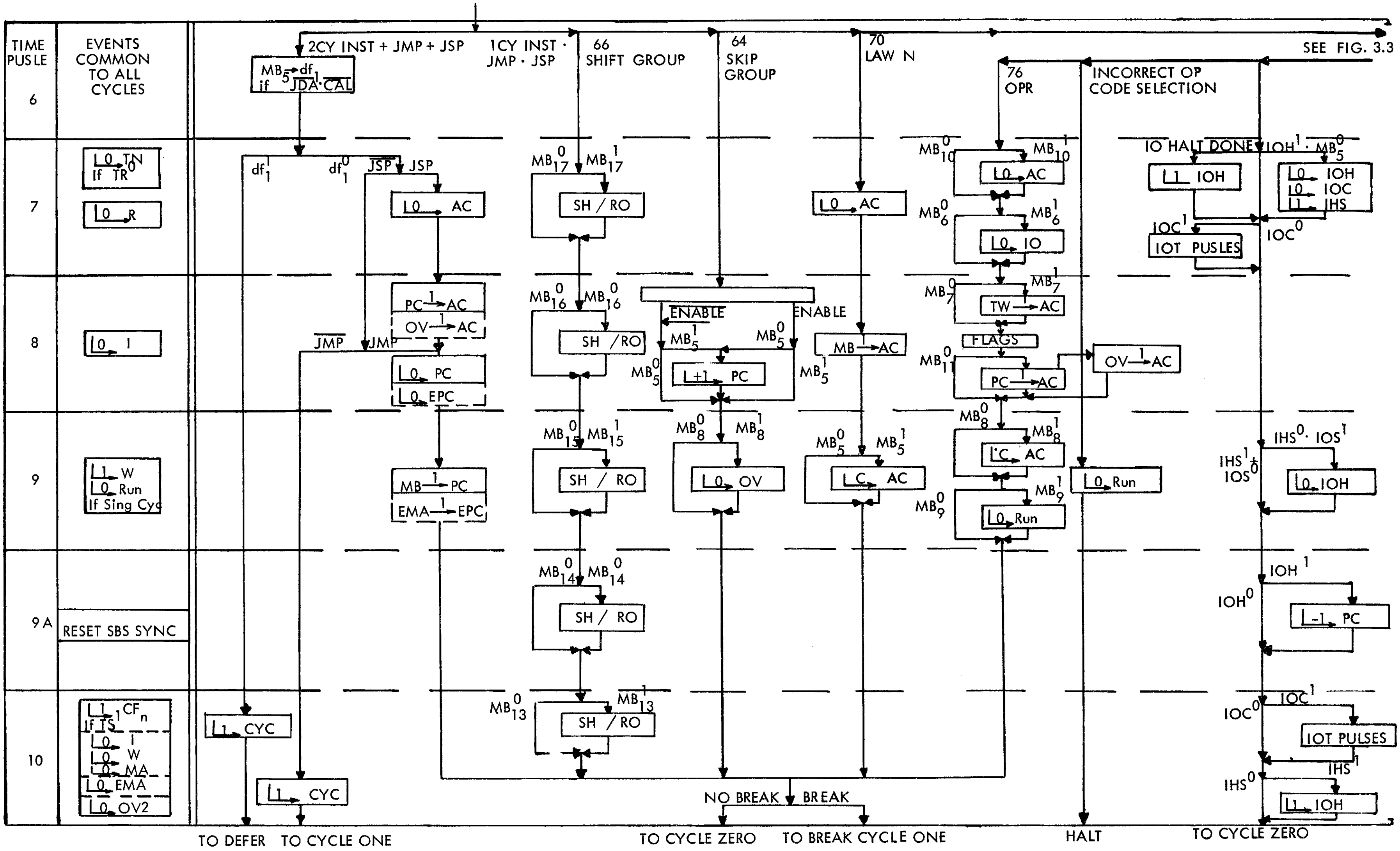


FIG. 3.2 - CYCLE ZERO FLOW DIAGRAM

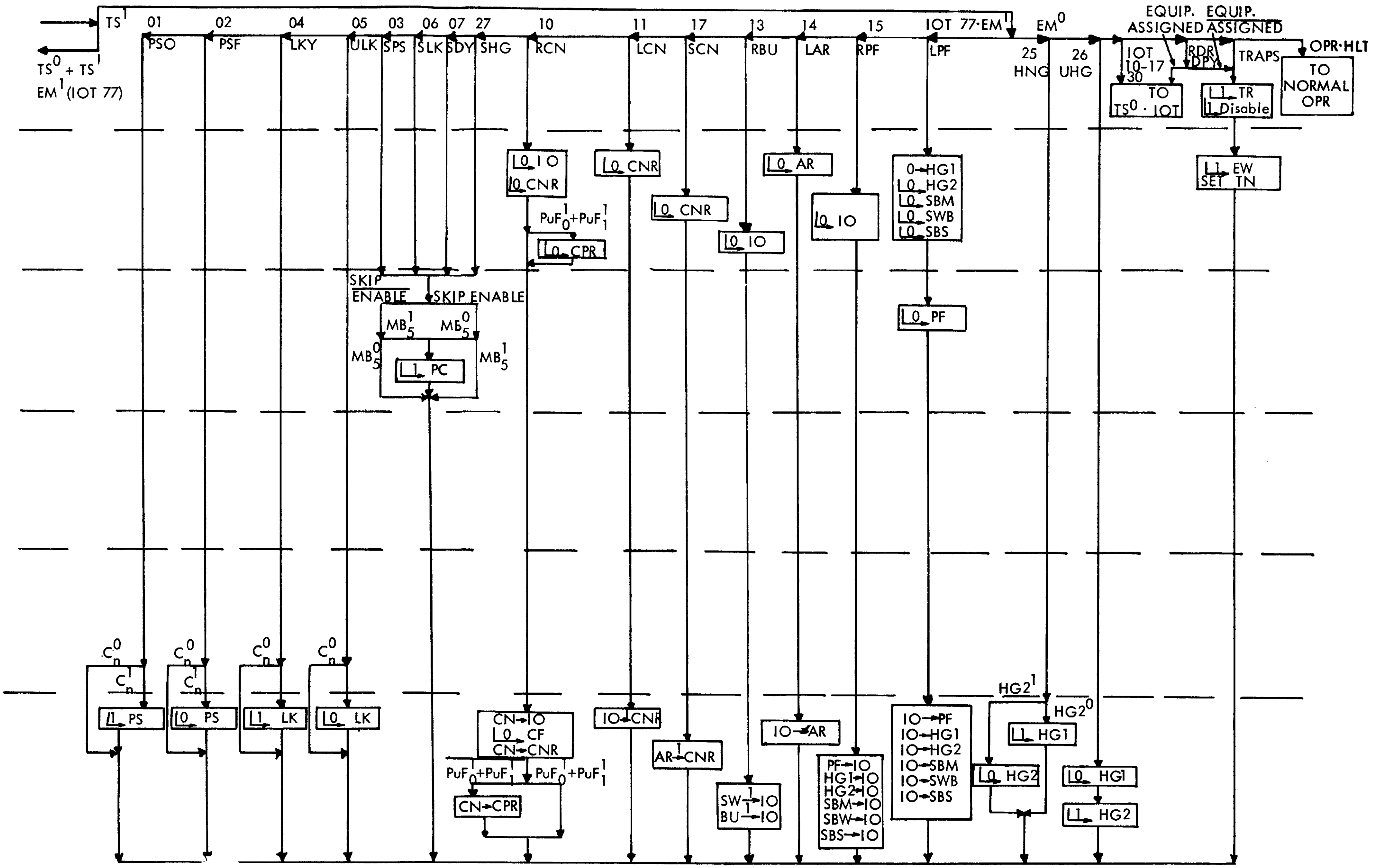


FIG. 3.3 - CYCLE ZERO MODIFICATIONS

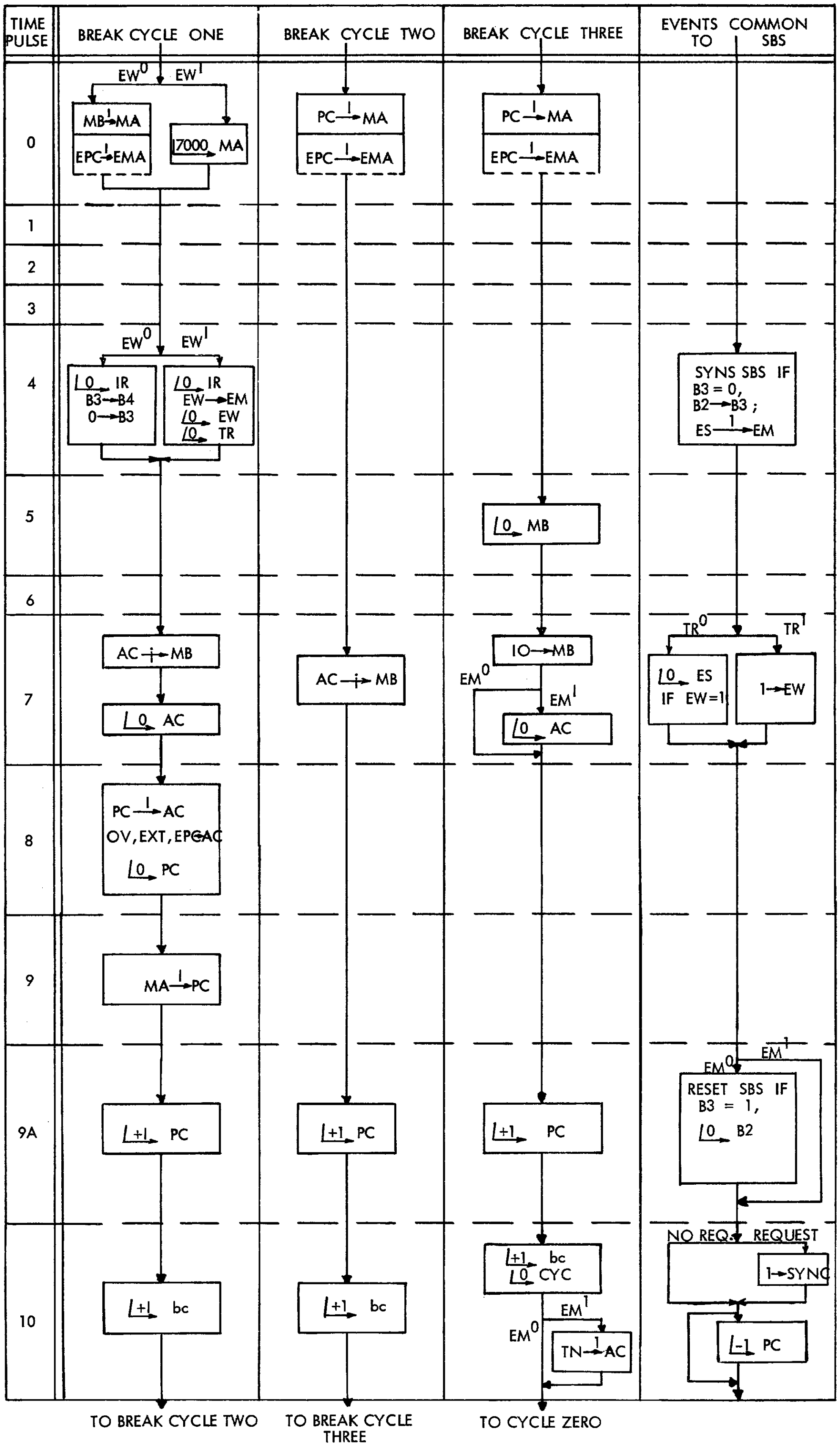


FIG. 3.4 - SPECIAL CYCLE FLOW DIAGRAM

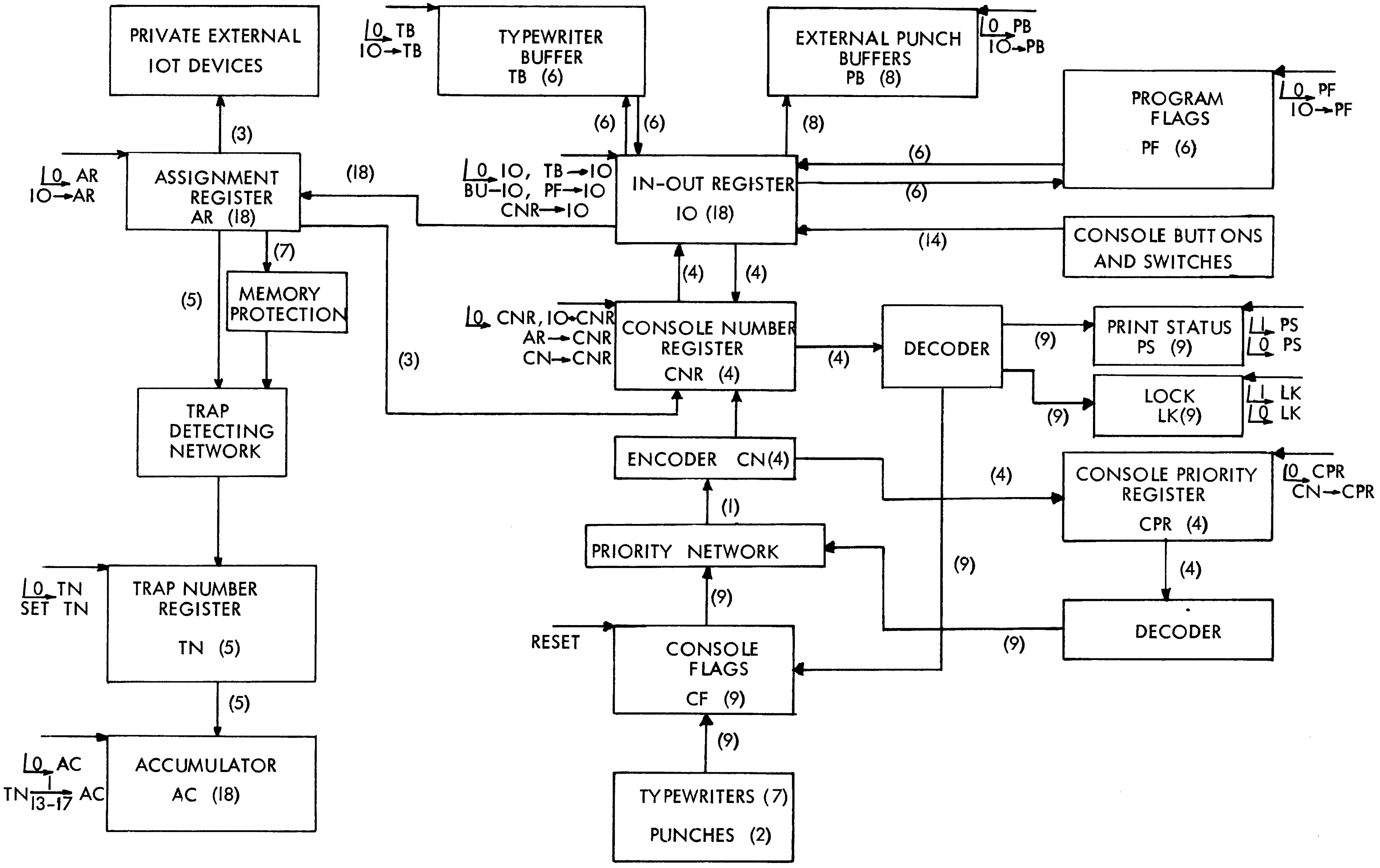


FIG. 3.5 - FLOW LINES, TIME SHARING SYSTEM

available to the machine. The design nicely falls into several groups which can be described separately.

1. New Instruction Decoding

The executive instructions are decoded in a straightforward manner from an auxiliary set of memory buffer decoders (Fig. 5.4). These are driven by a special set of bus drivers located in the PDP-1 to prevent loading of the regular MB drivers which are used for some external equipment (Fig. 5.3).

Thirty two instructions are decoded with 00, 12 and 24 not used. Space is left for additional decoders if desired. Several of the instructions which are implemented in the PDP-1 also have their timing derived here. These include RPF, LPF and RBU.

2. Print Status, Lock, Skip and Type Enable

The PS and LK flip flops work identically so only one will be described. The commands PSO and LKY turn ON the flip flops which are conditioned by the console number (decoded) held in the CNR at the time this command is giveny. Likewise PSF and ULK turn these flip flops OFF. Note that $PS_{8,9}$ and $LK_{8,9}$ refer to the punches 1 and 2 in this scheme (Fig. 5.5). The several conditions for SKIP are decoded with this group. When the command skip on print status (SPS) is given by the ER, it only wishes to address the PS flip flop associated with the console in the CNR. Print Status Enable (PSE) is true whenever the addressed Print Status flip flop is on. When the command SPS is given and PSE is true, the level SKIP ENABLE is true. If PSE is false, the level SKIP $\overline{\text{ENABLE}}$ is true. These two levels are OR'ed into the regular PDP-1 skip logic which causes the skip. These levels are conditioned by bit 5 of the MB in a manner similar to the Skip Group (see Appendix A4). The command SLK is handled indentically to SPS. SDY skips if the user in memory (CIM) is holding down his console display button (DYB). The instruction SHG investigates the HG flip flops only.

The seven Type Enable levels serve two purposes. They light an indicator on each of the seven consoles telling the user it is permissable for him to type in, and they unlock the typewriter for typing in. A user may type if PS is OFF and LK is OFF. Print Status ON indicates the computer is trying to type out, and PS OFF but LK ON indicates the users buffer is full in the ER.

3. Flag Registers and Console Priory

This circuitry contains the heart of the T-S system. The Console Flags (CF) are set whenever the external device (typewriter or punch) has finished with its last instruction andis ready for another. It is reset whenever that console has been serviced by the ER as indicated by an RCN. For a typewriter the flag is set on one of two conditions

1. Print Status is OFF and the pulse ST (for strobe type) appears indicating a key has legally been struck. If a key has illegally been stuck, as if LK is ON, the pulse ST does not appear.
2. Print Status is ON and the typewriter is finished typing out the last character. These are both levels so are strobed into CF on TP_{10}. The completion pulse from the typewriter cannot be used since when PS is first turned ON, no pulse appears until a character has been typed, but no character can be sent out until CF is set and serviced by the ER.

The flag for a punch (PuF) is set in a manner similar to (2) above since only output may occur.

The Flag registers are arranged in a priority chain so that no console may get better service than another. (Fig. 5. 6) This priority is rotating within the punches and within the typewriters, with the punches being serviced before the typewriters. The reason for this is purely mechanical. If a punch is not fed characters often enough it will go out of synchronization causing it to "chatter." In a pure rotating chain it would still get serviced every time out, but it may be the first on one time out and the last on the next time out, causing an interval slightly too long for decent operation.

The priority function is easily derived. The number of the last console serviced is stored in the Console Priority Register (CPR). (Fig. 5.6) The contents of this register are decoded and fed back into the priority chain. The point in the closed chain at which it is fed back is opened causing a new console (namely, the next one in the chain) to have highest priority for the next service. The break simulates the level NONE HIGHER. If this level is true and the CF is OFF, the level propogates to the next flag. If CF is ON, that flag sets up a level F and the NONE HIGHER line drops for all following flags. Thus only one F may be true at a time. The level F is encoded into a three bit console number CN_{1-3} indicating the typewriter with the highest priority now waiting for service.

The punches are handled a little differently. They are considered as consoles 8 and 9 as far as the console number is concerned. Therefore bit 0 of the CPR and CNR is a ONE for a punch. This feature is used to determine the console number for a punch. The Punch Flags are OR ed together to form this bit and this level is used to temporarily set to ZERO the number generated by the typewriter console flags. Bit 0 of the four bit CPR is not needed in the rotating chain and is used to hold the last punch served. It is ZERO for punch 0 and ONE for punch 1. In this manner punch 1 is served if punch flag 1 is ON and either punch 0 was last serviced or punch 0 does not want service now. The levels P0 and P1 are thus derived to mean "serve punch 0 or 1 now."

The CNR is loaded from IO_{14-17}, AR_{0-2} or the function just described on the commands LCN, SCN and RCN respectively. Its output is decoded into the 9 console numbers C_n for setting the PS and LK flip flops. The output of CPR is decoded for use in the priority chain. In this respect the output 0 was arbitrarly decided to have the same function as the output 7, namely interrupting the chain before flag 1.

4. Assignment Register

The Assignment Register is loaded from the IO on the instruction LAR. Bits 0-2 of the AR hold the current console-in-memory. These are decoded to permit operation of the correct set of sense switches and for indication purposes. Bits 3-5 are used for assignment of external IO equipment. Bits 6 and 7, 16 and 17, and 15 hold the assignments for the readers, displays and test word switches respectively -- one bit per device coded uniary. The rest of the bits, 8-14, are used for memory protection. These bits are checked against bits 8-14 of the MB. See Fig. 5.7. When the MB is greater than the contents of AR_{8-14}, a level is set for use by the trap logic.

5. Trap Logic

It has already been seen that certain instructions (Table 2.1) must trap to the ER when working in the T-S mode. When a trap is made, the instruction must be decoded, and a trap started, early enough to prevent any operation of the trapped instruction. The instructions are first decoded on TP_5 of cycle ZERO, and the earliest actions that must be prohibited are command pulses on TP_7 of an IOT instruction. This gives 0.8 microseconds to detect and start the trap. The conditions for each trap are determined separately (in parallel) and OR'ed together to form the possible trap condition. (Fig. 5.8) This condition enables the TRAP level of $CYC^{o} \cdot EM^{o}$. The signal T_1^1 is also present. Cycle ZERO must be included to prevent the possibility of data received in cycle ONE from causing a trap if it happens to have the same bit configuration as one of the illegal commands.

The level TRAP must initiate the trap and disable the function that caused the trap. When this function is disabled, the TRAP level will of course fall. Therefore TRAP must be stored in a flip flop TR. This must be set before TP, as mentioned earlier, so TP_6 is the only choice. To have the TRAP level come up before TP_6 requires that all logic involved before the TR flip flop be of the 5 mc type. This includes the logic for determination of an illegal memory reference.

6. Typewriter Logic

The logic for the typewriter is very similar to the standard DEC typewriter logic. Since the problems of synchronizing with a program no longer exist, most of the logic pertaining to completion pulses may be omitted, as may the type buffer status flip flop. The correct typewriter may be easily selected on a TYO by gating this pulse with the console number. To "lock" the keyboard to prevent unwanted type in is very simple also. The gate allowing the STROBE TYPE pulse may be conditioned by TE_n (type enable) as can the gate which clears the type buffer. When this "locked" the character is still typed on the paper, but doesn't get into the computer. Programming in the ER could be done to show such characters in red, if desired.

A separate set of such logic is associated with each typewriter. The type buffers in each set of logic are labled TB1, TB2, etc. These are mixed together in a TYPE MIXER (Fig. 5.7). On a TYI, the appropiate group of lines is pulsed depending on which console number is in the CNR, and the contents of the correct type buffer are strobed into the PDP-1 input mixer directly rather than thru gates, since it is already gated here.

B. TRAP CYCLE OPERATION

The circuitry needed to perform the storing of the AC, PC, IO and other miscellaneous operations is almost identical to that of the SBS system. For this reason the trap logic was made to look like another break channel -- which is valid with either the single or 16 channel system. The three flip flops associated with the normal break system are called Sync, Waiting Break and Break Started. Similarly the T-S trap flip flops have been named ES, EW and EM. The normal break system has been modified to load the trap number into the AC on break cycle 3 when EM is ON. (See Figs. 3.4 and 5.10) Otherwise operation is that of a normal break.

The logic is arranged such that a request by the time sharing system has priority over the other break channel. In a 16 channel system one may want a few channels of higher priority. The system will break whether or not we are "in" the sequence break mode. Effectively we can be in the break mode at all times by ignoring the break mode flip in deriving a request level. This is desired so the user may turn on or off his channel without affecting operation of the T-S system.

The flip flop ES is a synchronizing flip flop which holds the time-out pulse. If by chance EW should be set by both a time out and a trap on the same cycle, the trap will take precedence in the following manner: The AC is set from the TN register. On a timeout TN is not set, therefore the AC will contain 0, but on a trap TN is set by the trap condition. Thus when both occur the AC will hold the trap number establishing the trap in the AC rather than the timeout.

Once the break has been started, the EM flip flop (break started) is turned ON and remains ON until the sequence is dismissed by jumping deferred thru location 7001 as in the normal break system. If both a trap and a timeout occur, EW and TR will be ON and ES does not get reset. Then EW is reset before TR, so that level will be stored to cause a timeout interrupt as soon as the trap has been serviced. The request for a break is given when EM is OFF and EW is ON causing the next request to come as soon as EM is turned OFF by the end of the trap break.

The logic for determining where to start the deposit of the AC must be modified to start at 7000 rather than 0000. This simply means setting three bits to ONE on the condition TP_o $BC1 \cdot EW^1$. Since there are so many miscellaneous signals used in the trap system, and there happens to be several module positions available near the regular break channel where these are derived, it was decided to install all this logic for the trap system in that location.

C. CONSOLE OPERATION

Each of the seven consoles will consist of six sense switches, a push button, a display lever, and a on-off switch in addition to the typewriter and indicator lights already mentioned. (Fig. 5. 2) The sense switches are gated into the normal sense switch operation if that user is active in memory. The push buttons and on-off switches set levels which are read into the IO on the command RBU, and the display levers are sensed on the command SDY.

Two additional switches are provided for turning the main system ON. The switch TS_1 enables the entire system, including instructions, traps, etc. The switch TS_2 disables all functions on the main computer console. The two switches were provided to allow the main console functions when trying to debug the T-S system itself.

D. MODIFICATIONS AND ADDITIONS TO THE PDP-1

Several minor modifications and additions are necessary in the PDP-1 itself to implement the T-S system (Fig. 5. 10). The switch TS_2 is used to disable the console functions by stopping the start pulses and forbidding them to turn off the RUN flip flop. The break encoder is modified by gating it to work only when a break occurs and EW is OFF. Otherwise the new break encoder described earlier will work. Two gates are installed in the single channel break system to prevent the flip flops from being set and cleared if the EW flip flop is ON indicating the other channel (trap) has precedence. The other modifications include interchanging the wires MB_5^o and ENABLE and the wires MB_5^1 and $\overline{\text{ENABLE}}$ in the skip logic to permit MB_5 to condition the added skips; and prohibiting the in-out halt except on an RPA or RPB instruction.

1. Additions

The additions are mainly gates to allow machine registers to be set by the T-S system, and disabling buffer inverters. Gates are provided for setting the AC from TN, the PF from the IO, and the IO from the PF. On a trap, the instructions IOT, HLT and LAT are disabled by grounding the main instruction decoder for IOT and the lines LAT and HLT + ILL OP CODE + MAN'L RUN. The AC is zeroed on a trap or timeout by enabling the normal AC zero logic. In a similar manner the PF are cleared on LPF but here two inventer are needed, one to provide the pulse, and the other to indicate "all program flags".

Outputs to the T-S system include: the level STO which indicates a store or JMP class command for use in determining an illegal memory reference; the pulses Power Clear TP_6, TP_{7-4} and TP_{10-4}; and the levels IOT, LAT and HLT + ILL OP CODE + MAN'L RUN for use in the trap logic.

E. INTERCONNECTING WIRING

The above hardware is installed in a few spare module positions in the PDP-1 near where they will be used. In addition, several module positions have been taken for the purpose of proceding cable connectors to the T-S bay. The cables from bay 4 are terminated in a blank system module board and plugged in as any other module. This provides easy disconnect in the case of trouble, and a very handy way of wiring the changes (Fig. 5.11).

F. HARDWARE LAYOUT

The entire logic for the T-S system is contained in bay 4 and a few module positions in the main computer. The arrangement in bay 4 is indicated in Fig. 3.6. The top panel contains the switches TS_1 and TS_2, as well as indicator lights for the AR, CNR, CF and CPR. Panels H, J and K hold the electronics for two typewriters. Any additional are installed in panels on the rear door. For this reason the power supplies for the system are all located on the rear doors of bay 5.

	BAY 4	BAY 5
Y	INDICATORS	
Z	ASSIGNMENT REGISTER	ASSIGNED PROGRAM FLAGS
A	ASSIGNMENT REGISTER AND TRAP	BUFFERS AND IOT DECODERS
B	TRAP	INPUT MIXER
C	INSTRUCTION DECODE AND TIMING	PATCH PANEL
D	CONSOLE NUMBER	PATCH PANEL
E	CONSOLE NUMBER	TAPER PIN PANEL
F	PRINT STATUS, LOCK	TAPER PIN PANEL
H	TYPE MIXER	TPAER PIN PANEL
J	TYPEWRITER	OUTPUT CONNECTORS
K	TYPEWRITER	OUTPUT CONNECTORS
L	TYPEWRITER	

FIG. 3.6 - HARDWARE ASSIGNMENT IN BAYS

Bay 5 does not contain time-sharing equipment, but it contains various provisions for external IO connections. Provision is made for pulse outputs corresponding to IOT 10-17 on each of seven channels. These channels are gated by bits 3-5 of the AR as indicated earlier. This allows several users to program for the same channel, in the same manner as a drum field or break channel. Several panels of taper pin and banana plug patch panels are provided for mixing signals and connecting to a wide range of output connectors near the bottom of the bay. The doors of this bay have been shortened to allow users to connect their equipment semi-permanently without disrupting the appearance of the computer (see Fig. 3.1).

G. COMPLETION PULSE, STATUS BIT AND PUNCH PROVISIONS

One of the main considerations in building the T-S system is to make the system operate in such a way that it looks like a normal PDP-1 to the user. Thus completion pulses and status bits should be presented to the user for use in his program.

In the T-S system a completion pulse may be missed from the reader or display if it arrives when another user is in memory. Special provisions have been made for storing these completion pulses until the correct user arrives back in memory (Fig. 5.12). A simulated completion pulse is then given and the user's program processes the information, if he so desires.

1. Status Bit Provisions

It was described earlier how six new instructions must be added to turn off and on the status bits for the punch and typewriter in the T-S mode. In the non-T-S mode the regular status bits apply. All status lines, except for the punch and reader, are duplicated onto another leg of the input mixer in the T-S mode. Three flip flops -- punch, type out and type in -- are provided which set the status lines. These flip flops are set and cleared by the instructions PUN, PUF, TON, TOF, TIN and TIF if the CNR is identical to the CIM (AR_{0-2}). In this way the command will only change the status flip flop of the user in memory when one of his devices is being services, and not when the ER is servicing another console.

2. Extended Punch Buffers

For the same reason that the punches must be services first on a time out interrupt, to keep them in operation without "chattering," provisions must be made to prevent chatter during a drum swap. This takes up to 33 miliseconds, denying the punch of 2 or 3 characters. The only way to overcome this is to have three punch buffers in series which keep the punch supplied during this interval (Fig. 5.12). The ER can store three characters in these buffers, then keep supplying them one at a time as needed to keep all three buffers full. The condition for setting the punch flag with this kind of operation is buffer 3 (furthest from the punch) empty and PS on the TP_{10}. This can be derived from the flip flops associated with the extra punch buffers.

CHAPTER IV

PROGRAMMING FOR THE SYSTEM

The programming for the time-sharing system consists of two parts -- the executive routine (ER) and the administrative routine. The former is a small program of about 450 registers which occupies upper memory at all times the system is in operation. Its function is to handle the second-to-second administration of the system including such things as time out interrupts, traps, character processing and input-output buffers for the external equipment. The administrative routine is a much longer program which is called in by the ER whenever it is needed. This routine handles all equipment assignment requests of the user, comments in the case of errors, etc. In the course of its work it modifies the ER to handle the current conditions.

A. THE EXECUTIVE ROUTINE

The executive routine consists of several subprograms which will be described separately. These include the character service routine, trap dispatch routines, and dismiss routine. When a trap is made by the system, the contents of the AC, PC, and IO are stored in locations ac, pc, and io of the ER, the trap number is put into the AC, and control is transferred to location io + 1. This is accomplished by the SBS system which has been modified to store these registers starting at location 7000_8.

As was mentioned earlier the area occupied by the ER is at the top of memory and is protected against any attempts by a user's program to alter the contents of any register in the ER. This protected area is divided into two sections as shown in Fig. 4. 1. The lower section is referred to as the user's executive area. It contains information used by the ER but which pertains only to the user in memory. This includes the trap dispatch tables, optional sections of the ER such as drum or SBS options, and constants and registers such as the length of a user's quantum and his assignment register. This lower section is swapped with the user's program when he is dismissed.

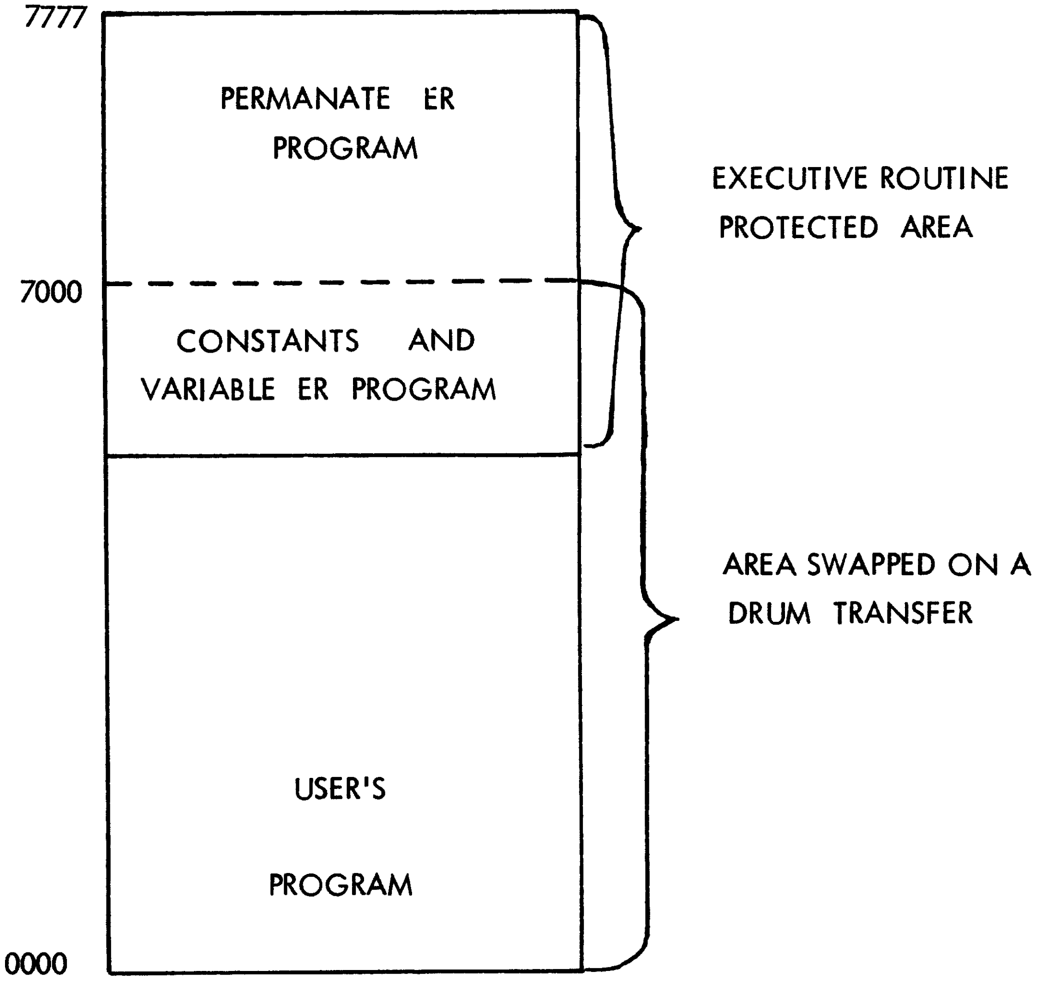


FIG. 4.1 - MEMORY USE IN TIME SHARE SYSTEM

A general flow chart of the ER program is shown in Fig. 4.2. When the ER is entered, a dispatch is made on the proper trap number. These fall into three general classes: input-output traps which require character transmission to or from a buffer; equipment traps where it must be decided whether the equipment request command is legal and then perform the necessary calculations or operations; and time out interrupts which perform service to external equipment. The first two classes return to the user after performing their function, unless it is not possible to transmit the character (buffer full or empty) in which case the user is dismissed. The time out service determines whether the users quantum is finished, then dismisses or returns to the user. The dismiss sets up the machine for the drum swap, and determines whether there is time enough left to service more equipment before the drum is ready. If not the swap is started immediately, otherwise more service is given. More detailed flow charts of the ER are given in Figs. 4.3 thru 4.11. A copy of the program itself is found in Appendix C. The individual sections will be explained in more detail below.

1. Trap Dispatch

Upon first entering the ER, a dispatch is made on the trap number to the correct program sequence. The main sequences are those for time out interrupt, and IOT instruction traps TYO, TYI, PPA and PPB. The drum and SBS sequences are optional by request. There are also dispatches for several miscellaneous commands vital to the administrative routine. The trap dispatch table is established by the administrative routine at each user's initial request based on his equipment requests. See Fig. 4.3.

2. Time Out Interrupt

This routine determines the next console desiring service of its external equipment, if any. If it finds one, it goes to a service routine, services, and returns to determine the next console. When there are no more requests, action is started to return to the user's program if his quantum is not up. He may lengthen his quantum once by depressing the display button on his console. If his quantum is up he is dismissed.

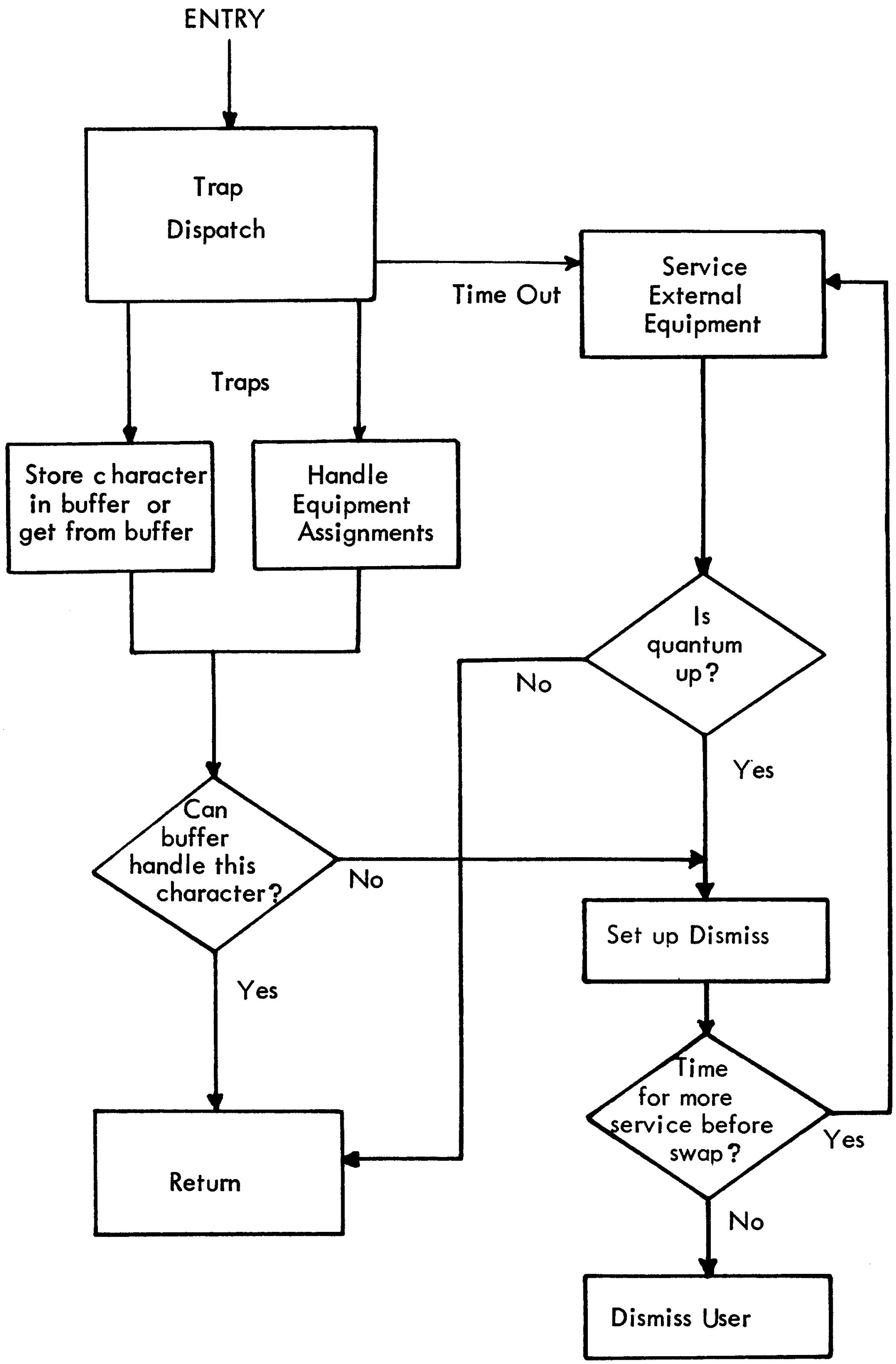

ENTRY
Trap Dispatch
Time Out
Service External Equipment
Traps
Store character in buffer or get from buffer
Handle Equipment Assignments
Is quantum up?
No
Yes
Can buffer handle this character?
No
Yes
Set up Dismiss
Time for more service before swap?
Yes
No
Return
Dismiss User

FIG. 4.2 - GENERAL FLOW DIAGRAM OF EXECUTIVE ROUTINE

3. Initialize and Index the ADM Tables

These are three subprograms which are called from various places throughout the ER for the purpose of keeping track of the several consoles information. The ER contains a character buffer for each typewriter or punch connected to the machine. These are ring buffers, that is when the bottom of the buffer is reached, the top location is considered next.

With each buffer there are associated three pointers called the ADM pointers. Pointer ADM (1, n) points to the location in the n^{th} buffer where the next character is to be placed -- the input pointer. ADM (2, n) points to the position in the n^{th} buffer where the next character is to be taken from -- the output pointer. ADM (3, n) points to the end of the buffer (or rather the end plus one). These locations are in the address portions of the pointers. In addition bit 0 if ADM (1, n) is a ONE if the buffer is full. Bit 0 of ADM (2, n) is a ONE if the buffer is empty.

The subprogram sat (Fig. 4.4) initializes three locations to hold the pointers pertaining to the current console being serviced as determined by the program calling sat.

The subprograms ite and itf index these pointers and test for empty or full buffers (Fig. 4. 5). Ite indexes the output pointer and checks against ADM (3) for the bottom of the buffer. If agreement the pointer is reset to the top. Then it is checked against the input pointer. If equal the buffer must be empty since the output pointer has caught the input pointer. Bit zero is set for future use. It works in exactly the same manner but it checks for a full buffer by checking whether the input pointer has caught the output pointer.

4. Time Out Service

Before the service routine is described, a word should be said about the operation of the PS and LK flip flops. This part of the ER is programmed with the following convention. If PS is ON, the typewriter or punch has material it wants to output. When PS is OFF, the punch is free, or the typewriter can accept typed in material if the LK flop flop is OFF. If locked, any material typed does not get into the computer.

This subprogram handles all character processing between the computer and the external equipment (Fig. 4. 6). The correct pointers are first set up, and depending on the requesting devices the correct commands are given to service the device. On a punch or type out the buffer cannot now be full, so the status bits are turned off and the appropriate pointers are reset. If the request was a type in, the command is executed and the keyboard is locked to prevent further type in if the buffer becomes full.

5. Type Out Routine Trap 3

The type out command can only be given by the program in memory, so the console number is set from the AR. The buffer can accept material if the condition $PS^1 + PS^O LK^1 = PS^1 + LK^1$ exists. Otherwise the program still has a request in for more typed material and the user is dismissed (Fig. 4. 7). If the condition is met, the ADM table is set up using sat, and the character is stored in the buffer, if it is not full. If it is, the user is dismissed. PS and LK are turned ON indicating service required on a time out and that the buffer is in use by a TYO. The input pointer is indexed using it and if the buffer is full the type out status bit is turned on before returning to the user.

6. Type In Routine, Trap 4

This routine is very similar to the type out routine given above (Fig. 4. 7). The TYI command is considered legal if the condition $PS^O \cdot LK^O$ exists, and the ADM tables are then set up. If not, the user is dismissed. The IO is loaded from the buffer, if it is not empty. If empty he is dismissed. The LK flip flop must be turned OFF and PS must be turned OFF before this last dismissal since this may be the first request for a type in and the user must be allowed to type even if the program is dismissed. The output pointer is indexed and the type in status bit is turned off, if the buffer is empty, before returning.

7. Punch Paper Tape, Traps 5 and 6

The punches may be called by any user, (Fig. 4.8) therefore CN is set by loading from the IO. The table is set up, the user is dismissed if the buffer is full, and the character is stored in the buffer if not full. Next the input pointer is indexed and the punch status bit is turned on if the buffer is now full.

The difference between a PPA and a PPB is on the method of storage. A PPA punches 8 holes from bits 10-17 of the IO. A PPB punches 6 holes from bits 0-5 of the IO, with hole 7 always blank and hole 8 always punched.

When a PPB trap occurs, the character is rotated into bits 12-17 and bit 10 is made a ONE. Then it can be punched out with a PPA command. This allows frequent mixing of PAA's and PPB's.

8. Drum Instructions, Traps 7, 8, 9 and 10

This is an optional part of the programming which is requested from, and put into the user's executive area by, the administrative routine if the user wants his program to execute drum transfers. This accomplishes two objectives. The user may have written his program assuming a drum field, n, which is already assigned to something else. In this case all references to drum fields must be modified to read field m, the one assigned to him. This assignment is made by the administrative routine.

The other objective has to do with the time required to go thru the executive routine on such a drum request. The user probably has requested a break on a certain address to allow himself time for a short subroutine of known length before the transfer request. The ER takes extra time to execute the break request and also on the transfer request. Therefore the address must be modified to give a break earlier than that requested by the user. A similar condition exists on the read drum address instruction. The constants fc1 and fc2 must be supplied by the administrative routine.

9. Dismiss Routine

The function of the dismiss routine is to prepare the way to bring in a new user when it has been deemed time to dismiss the present user. The normal entry is at location dms (Fig. 4.9); for the present we will ignore entry dm0.

The first function of the dismiss is to save the program flags so they can be used by the ER, then to determine if any new requests have arrived from the users for special service. The command RBU reads the current buttons and switches into $IO_{0,1}$ and $IO_{9,10}$ respectively. If no new request have arrived it starts setting up the write field, WFLD.

Any new request cause the appropiate indicators in the fld field to be set. If bit 0 is a ONE, that user wants back; if bit 1 is a ONE he wants back having pushed his button, and if bit 1 is a ONE, he has turned on his switch indicating the administrative routine is wanted. Thus if only bit 0 is ON he wants his original program back. This is normally ON; if the buffer becomes empty on a TYI or if the wrong punch command is given, dismiss and wait is given which turns it off until the condition is cleared.

Location dm1 starts the process of drum transfer. Fdp is a pointer indicating which user is currently in memory. This is indexed, modulo 4, and the new user is determined. If he is waiting (bit 0 is ZERO) the program loops back to dml (Fig. 4. 10). This continues until a valid user is found or it is determined all users are waiting, in which case the only thing that can be done is service more consoles if, and when, they want service. When a valid user if found, the next two bits are investigated. If both are zero, the new read field (RFLD) and word count are set up in the AC, and dm2 is the next instruction.

If one of the bits is non-zero control is transferred to bug, a subprogram which determines what the user wants. The switch of course requests the administrative routine. The button may request the administrative routine called adm, another field containing some debugging routine called flt, or a return to a debugging routine in his own program -- in which case the main program must be brought back and the pc changed. The location to which the button is to given control is found in the locations deb for each program. It is assumed that user 0 will never make mistakes since this is a debugged program running without an operator.

The deb location contains the RFLD and word count for the users button. This is set by the administrative routine when the user request service. After the read field is set, control jumps to dm2, which stores the RFLD and word count just determined for future use. Now that the information for the swap has been completely determined, there still might be time for more console servicing before the drum comes to the correct position (see Fig. 4. 11). If there is sufficient time, more characters are processed by the service routine. However, now a dismiss by the service routine enters the dismiss routine where it was left above which again checks the amount of time left. After the first check back for characters, a closed loop will probably be formed until the drum comes to correct location.

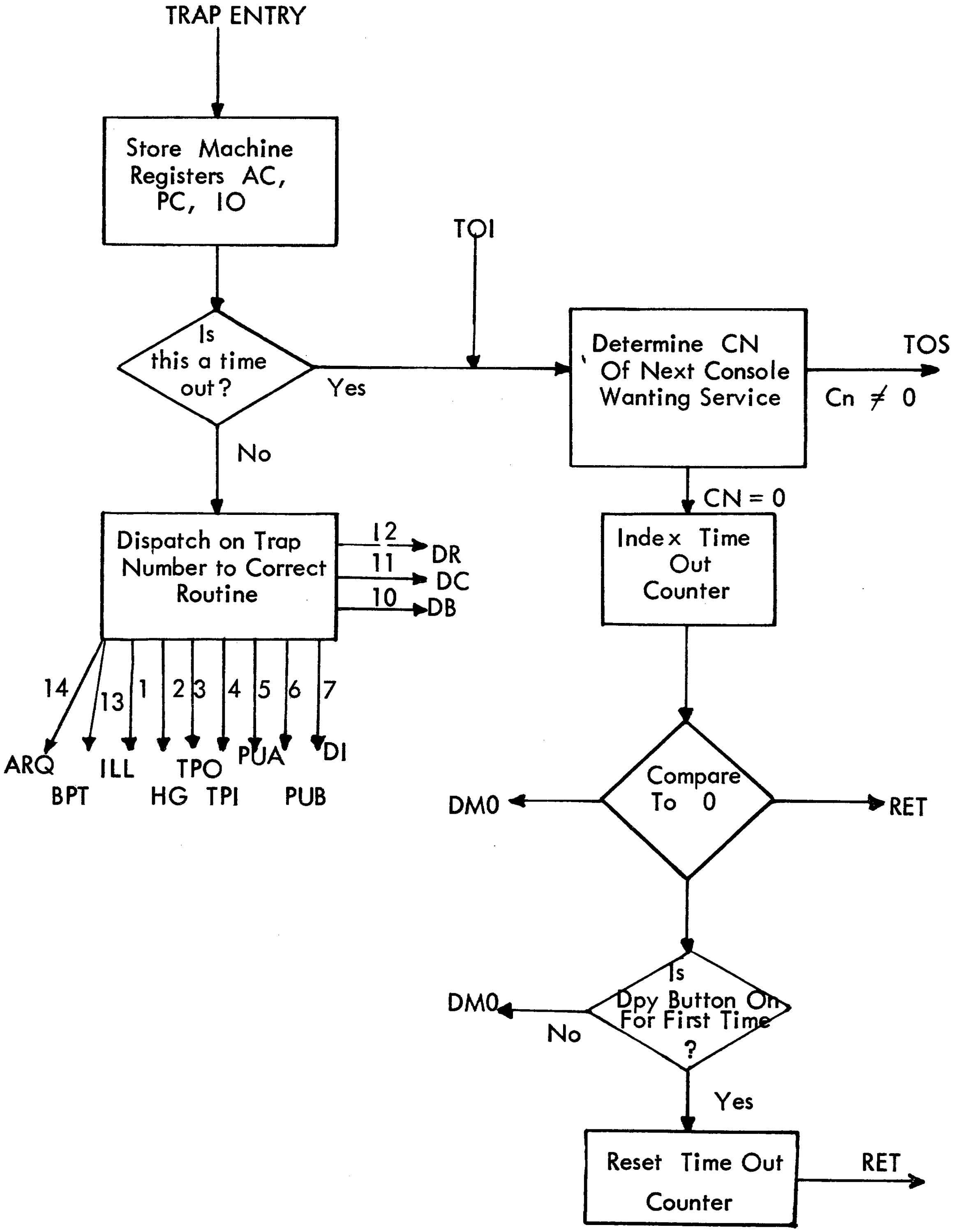


FIG. 4.3 - FLOW DIAGRAM - TRAP DISPATCH AND TIME OUT INTERRUPT

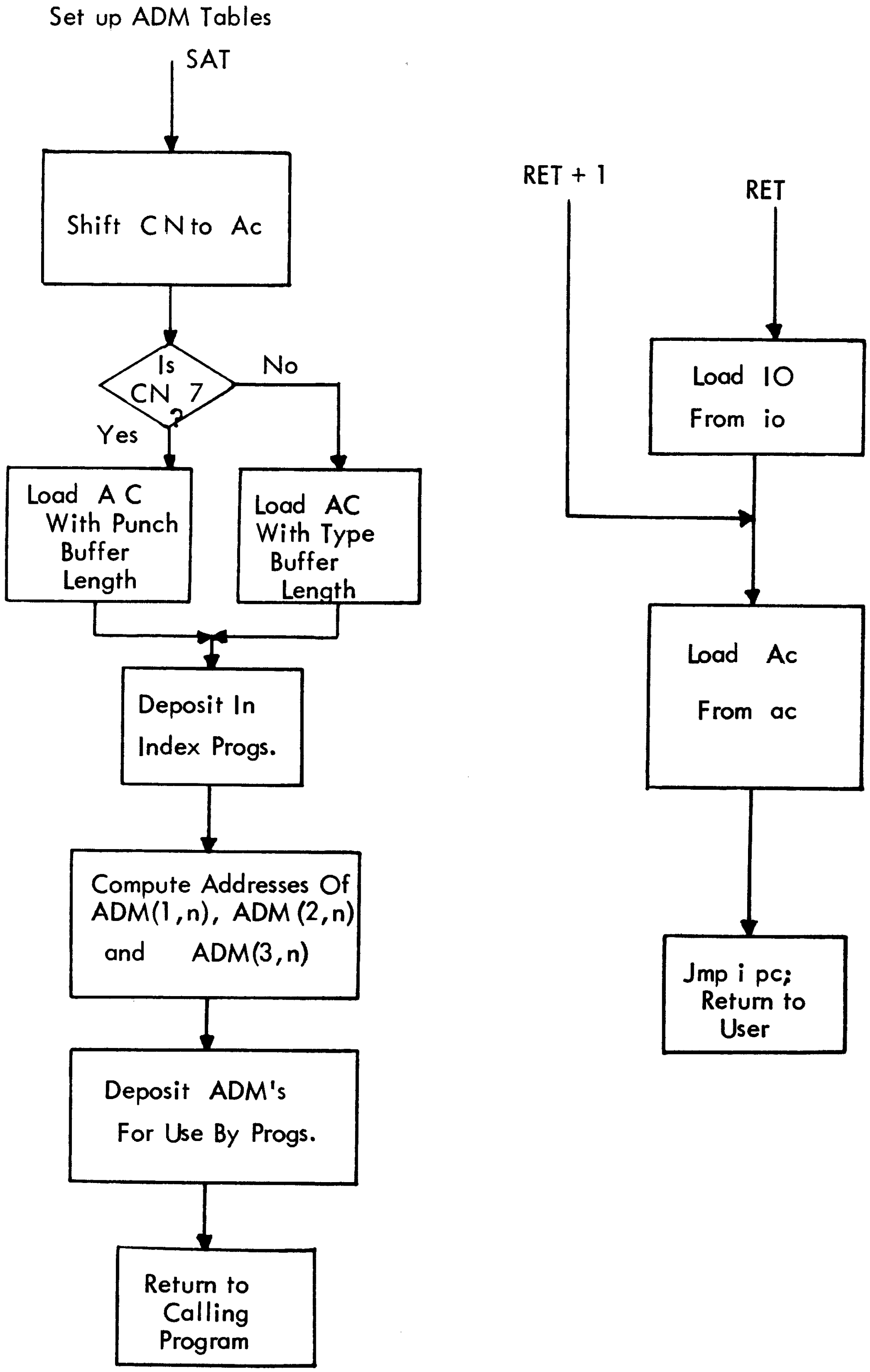

Set up ADM Tables
SAT
Shift C N to Ac
Is CN 7 ?
No
Yes
Load A C With Punch Buffer Length
Load AC With Type Buffer Length
Deposit In Index Progs.
Compute Addresses Of ADM(1,n), ADM (2,n) and ADM(3,n)
Deposit ADM's For Use By Progs.
Return to Calling Program
RET + 1
RET
Load IO From io
Load Ac From ac
Jmp i pc; Return to User

FIG. 4.4 - FLOW DIAGRAM - INITIALIZE TABLES; RETURN

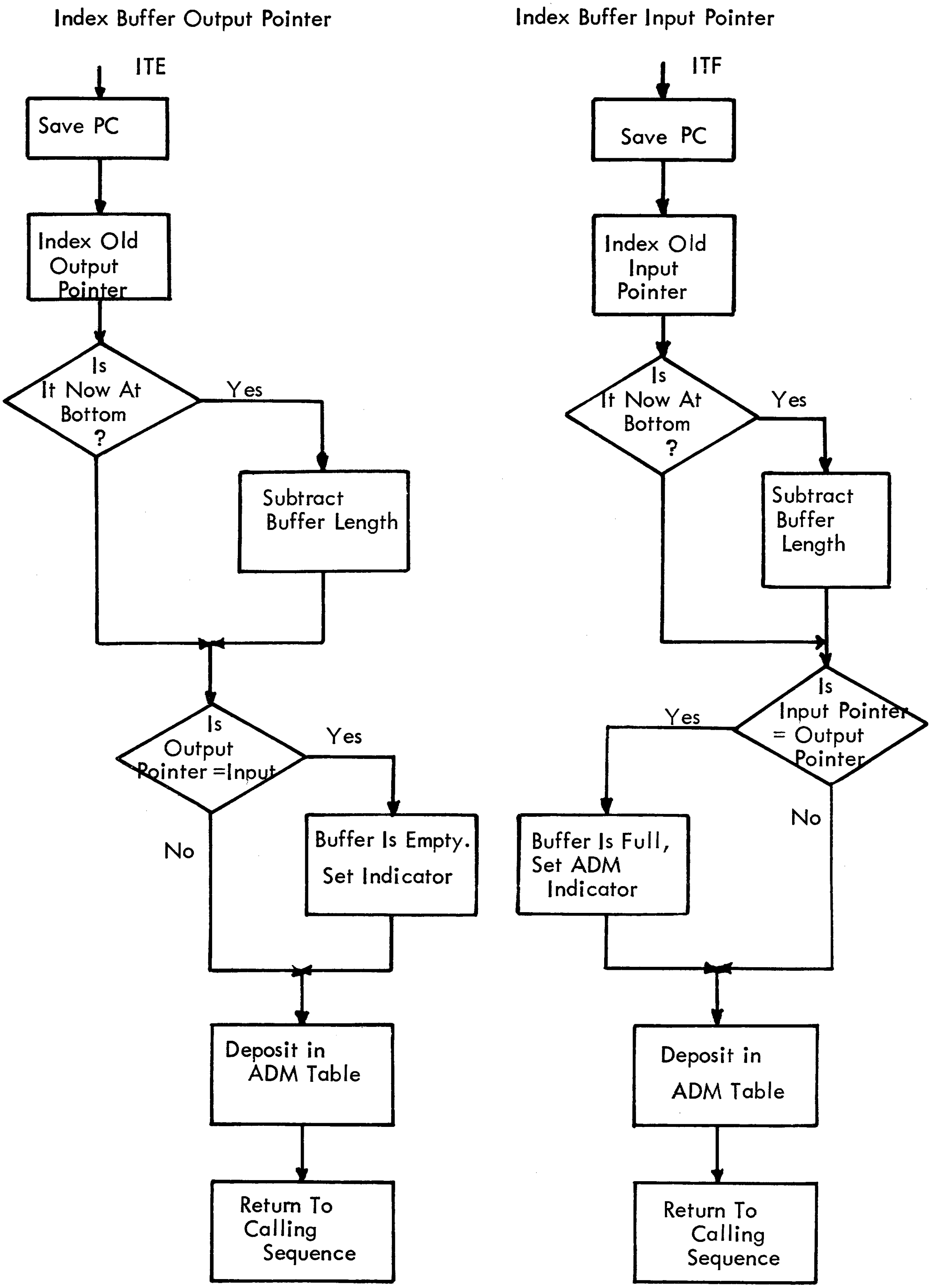


FIG. 4.5 - FLOW DIAGRAM - INDEX AND TEST TABLES

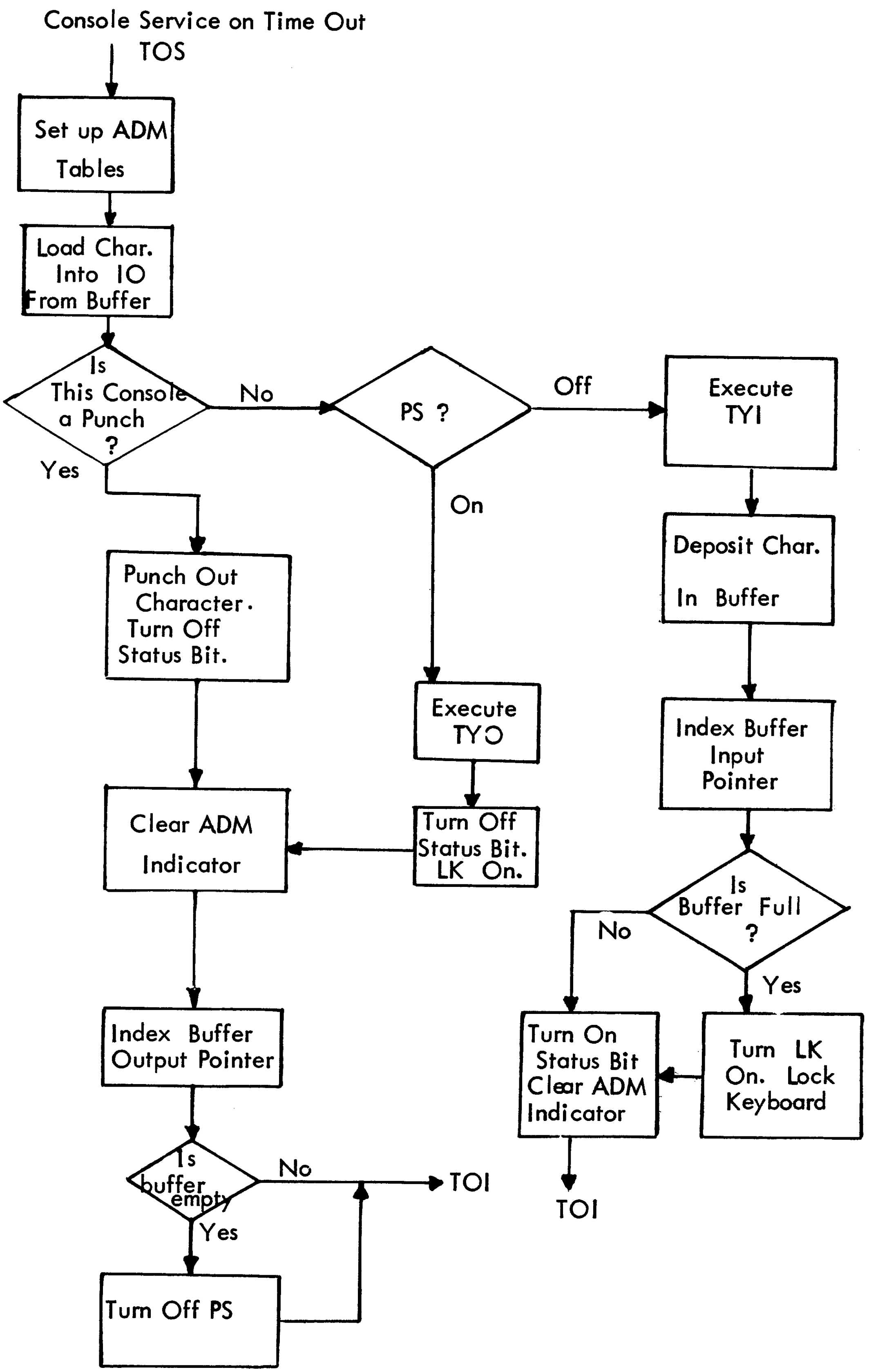

Console Service on Time Out
TOS
Set up ADM Tables
Load Char. Into IO From Buffer
Is This Console a Punch ?
No
Yes
PS ?
Off
On
Execute TYI
Deposit Char. In Buffer
Index Buffer Input Pointer
Is Buffer Full ?
No
Yes
Punch Out Character. Turn Off Status Bit.
Execute TYO
Turn Off Status Bit. LK On.
Clear ADM Indicator
Index Buffer Output Pointer
Is buffer empty
No
TOI
Yes
Turn Off PS
Turn On Status Bit Clear ADM Indicator
Turn LK On. Lock Keyboard
TOI

FIG. 4.6 - FLOW DIAGRAM - TIME OUT SERVICE

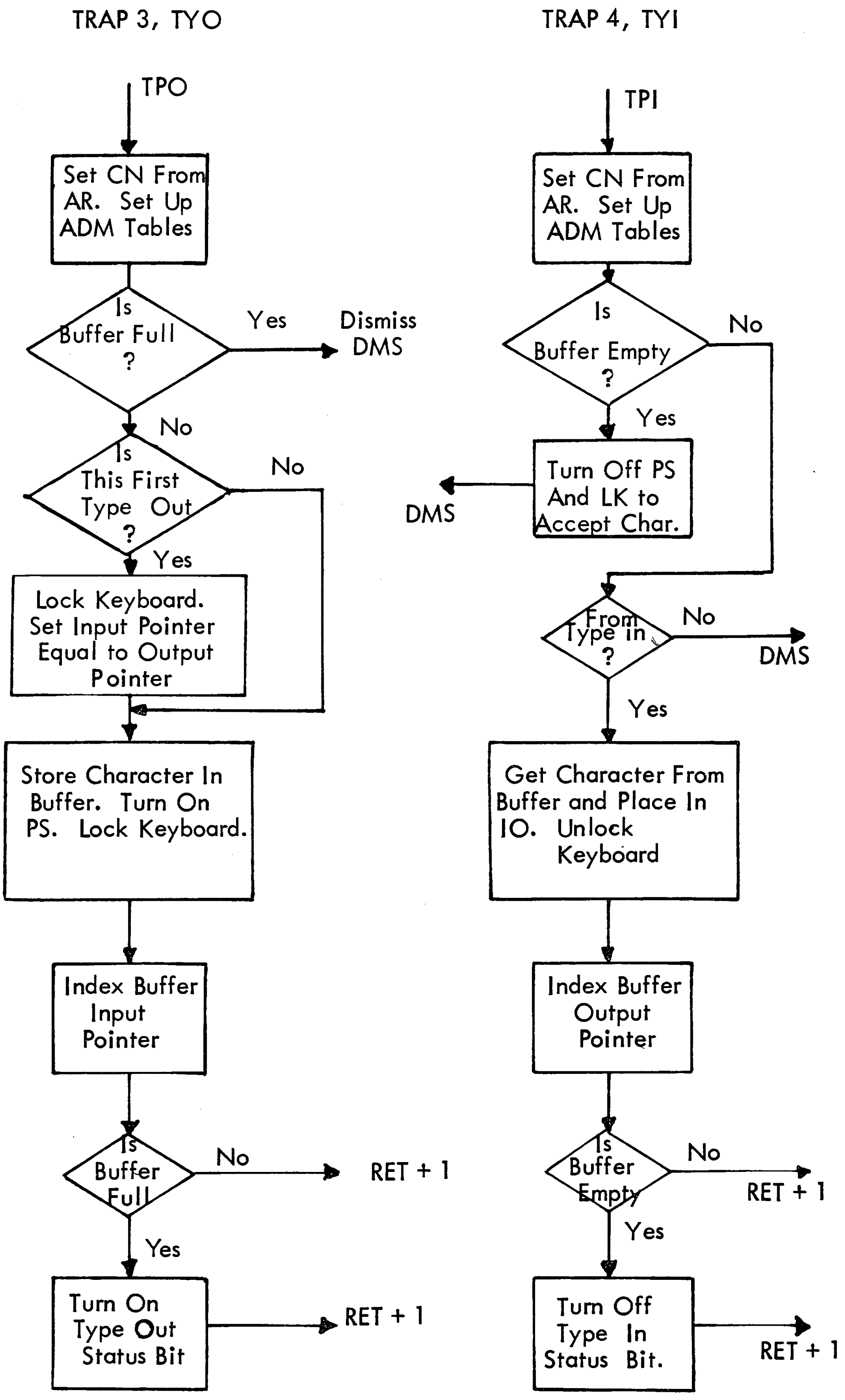

TRAP 3, TYO
TPO
Set CN From AR. Set Up ADM Tables
Is Buffer Full ?
Yes
Dismiss DMS
No
Is This First Type Out ?
No
Yes
Lock Keyboard. Set Input Pointer Equal to Output Pointer
Store Character In Buffer. Turn On PS. Lock Keyboard.
Index Buffer Input Pointer
Is Buffer Full
No
RET + 1
Yes
Turn On Type Out Status Bit
RET + 1
TRAP 4, TYI
TPI
Set CN From AR. Set Up ADM Tables
Is Buffer Empty ?
No
Yes
Turn Off PS And LK to Accept Char.
DMS
From Type in ?
No
DMS
Yes
Get Character From Buffer and Place In IO. Unlock Keyboard
Index Buffer Output Pointer
Is Buffer Empty
No
RET + 1
Yes
Turn Off Type In Status Bit.
RET + 1

FIG. 4.7 - FLOW DIAGRAM - TYPE ROUTINES

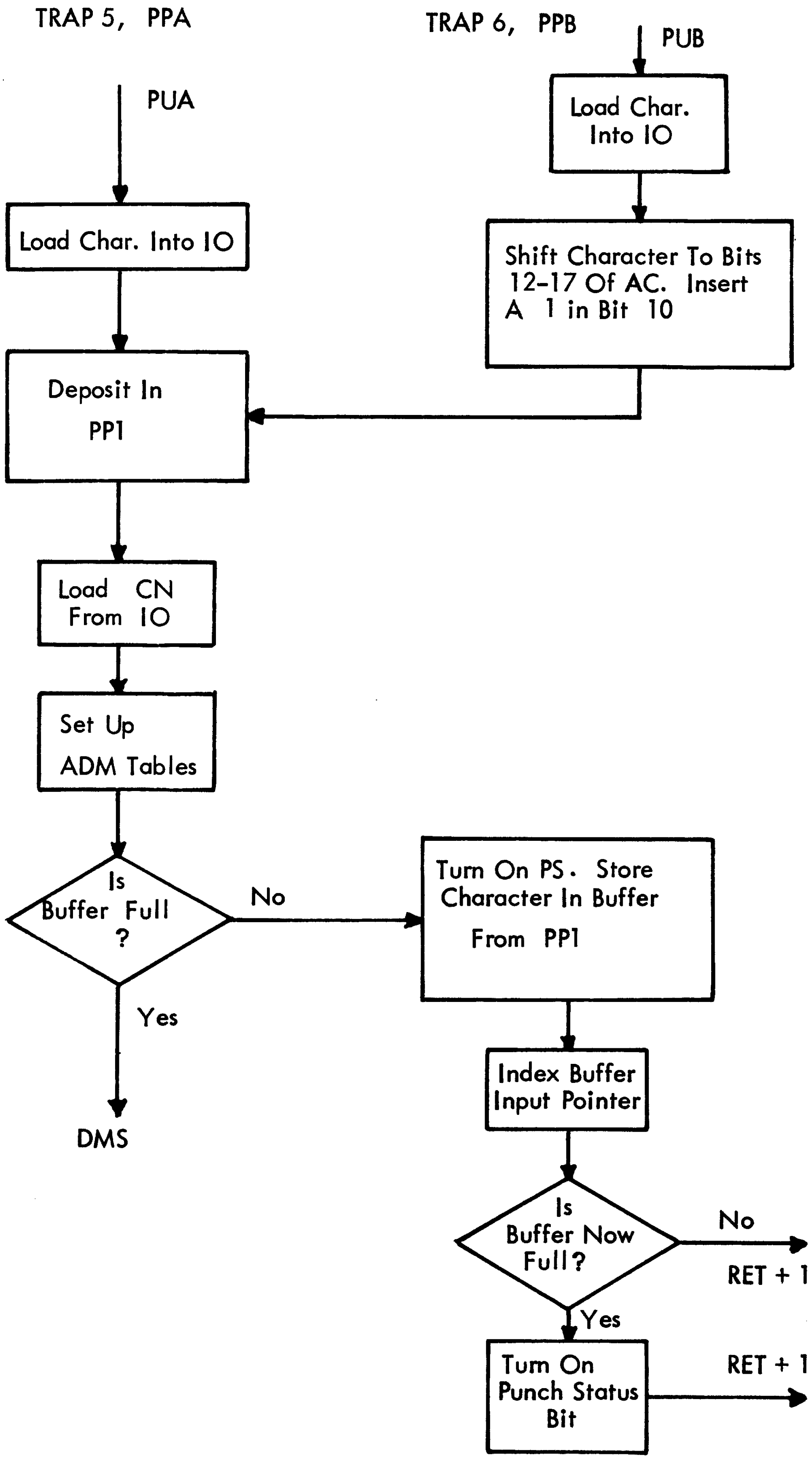


FIG. 4.8 - FLOW DIAGRAM - PUNCH PAPER TAPE

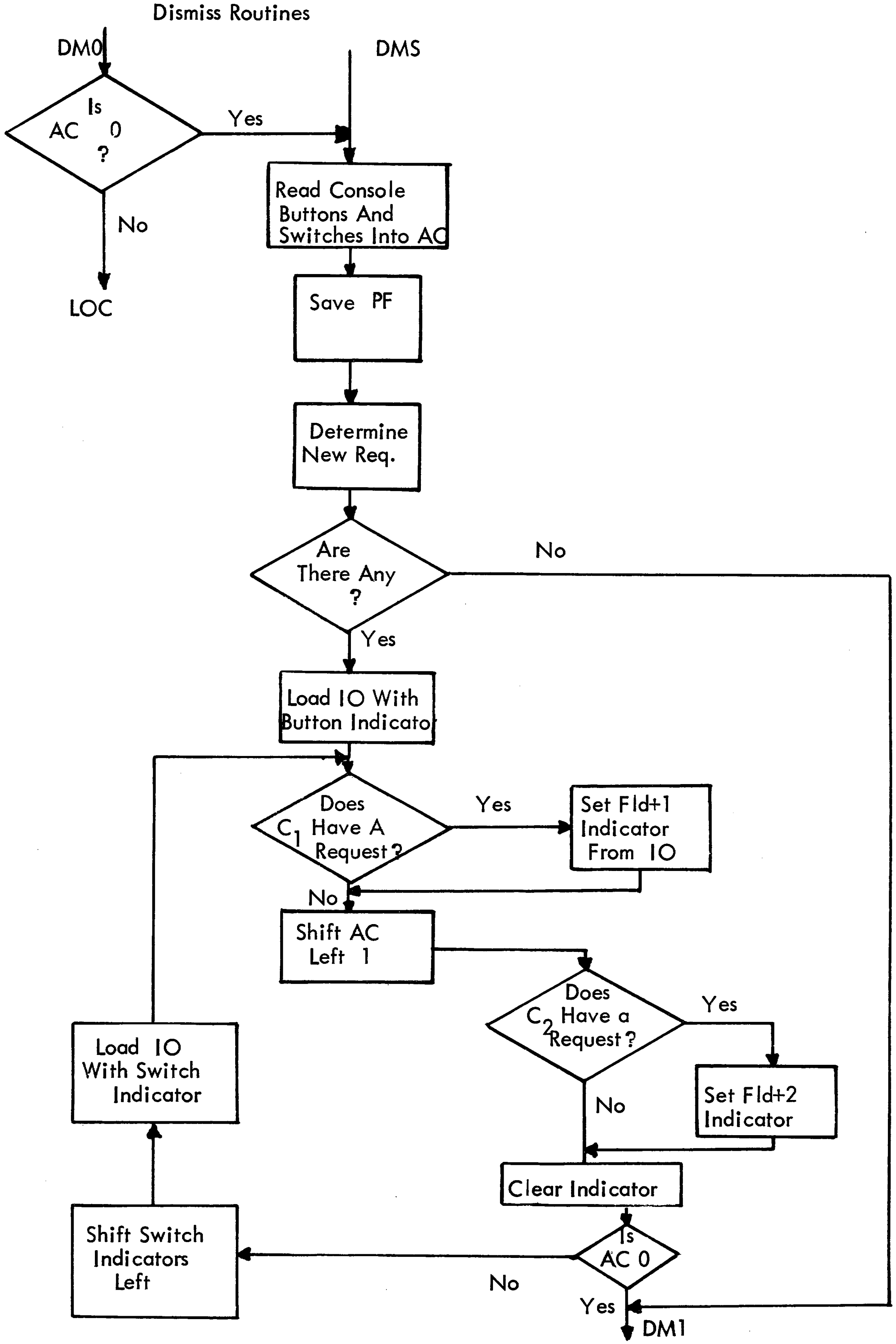

Dismiss Routines
DM0
DMS
Is AC 0 ?
Yes
No
LOC
Read Console Buttons And Switches Into AC
Save PF
Determine New Req.
Are There Any ?
No
Yes
Load IO With Button Indicator
Does C_1 Have A Request ?
Yes
Set Fld+1 Indicator From IO
No
Shift AC Left 1
Does C_2 Have a Request ?
Yes
Set Fld+2 Indicator
No
Clear Indicator
Is AC 0
No
Yes
DM1
Load IO With Switch Indicator
Shift Switch Indicators Left

FIG. 4.9 - FLOW DIAGRAM - DISMISS ROUTINE

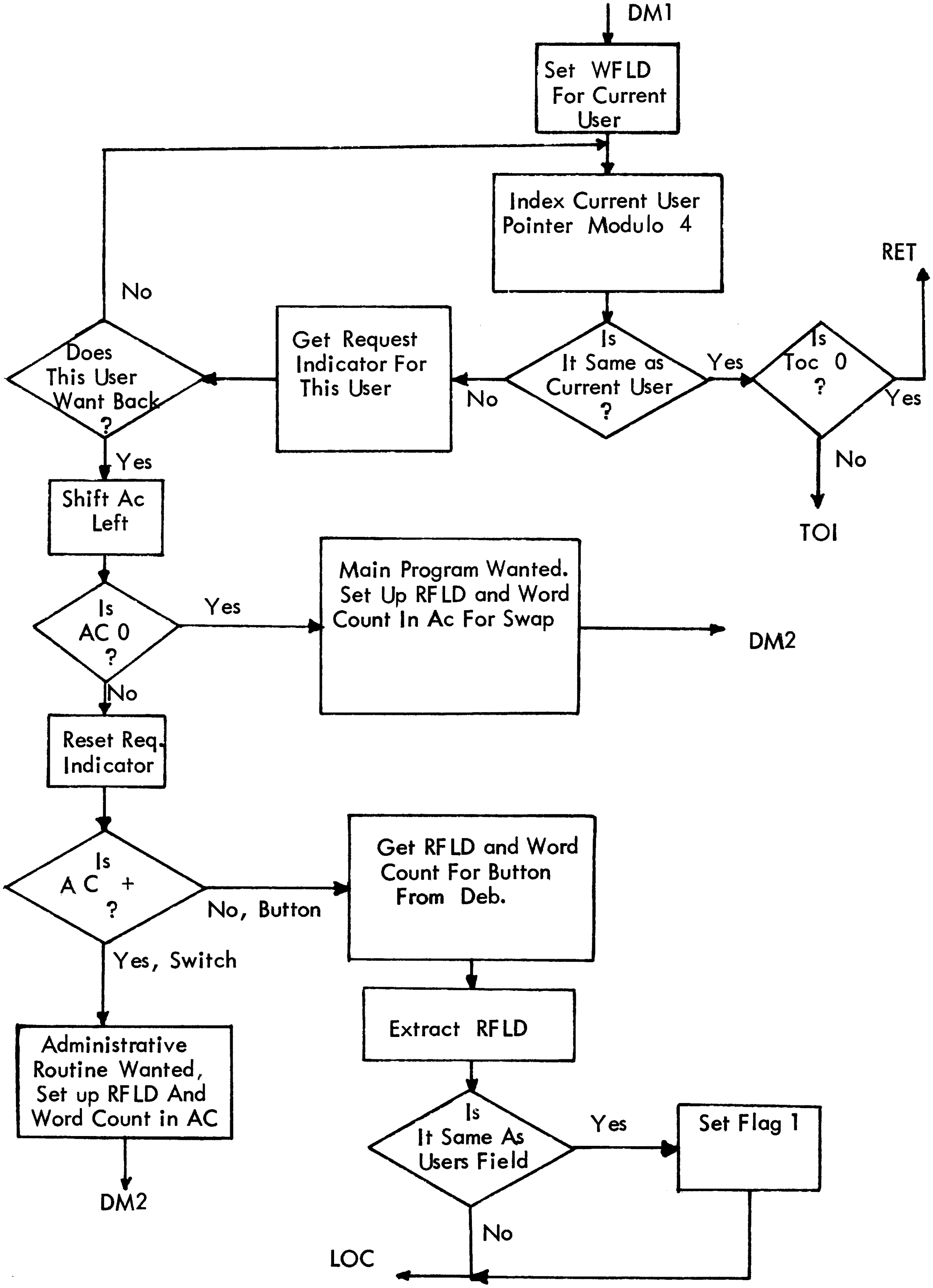

DM1
Set WFLD For Current User
Index Current User Pointer Modulo 4
Is It Same as Current User ?
Yes
No
Is Toc 0 ?
Yes
RET
No
TOI
Get Request Indicator For This User
Does This User Want Back ?
No
Yes
Shift Ac Left
Is AC 0 ?
Yes
Main Program Wanted. Set Up RFLD and Word Count In Ac For Swap
DM2
No
Reset Req. Indicator
Is A C + ?
No, Button
Get RFLD and Word Count For Button From Deb.
Yes, Switch
Administrative Routine Wanted, Set up RFLD And Word Count in AC
DM2
Extract RFLD
Is It Same As Users Field
Yes
Set Flag 1
No
LOC

FIG. 4.10 - FLOW DIAGRAM - DISMISS AND DEBUG ROUTINES

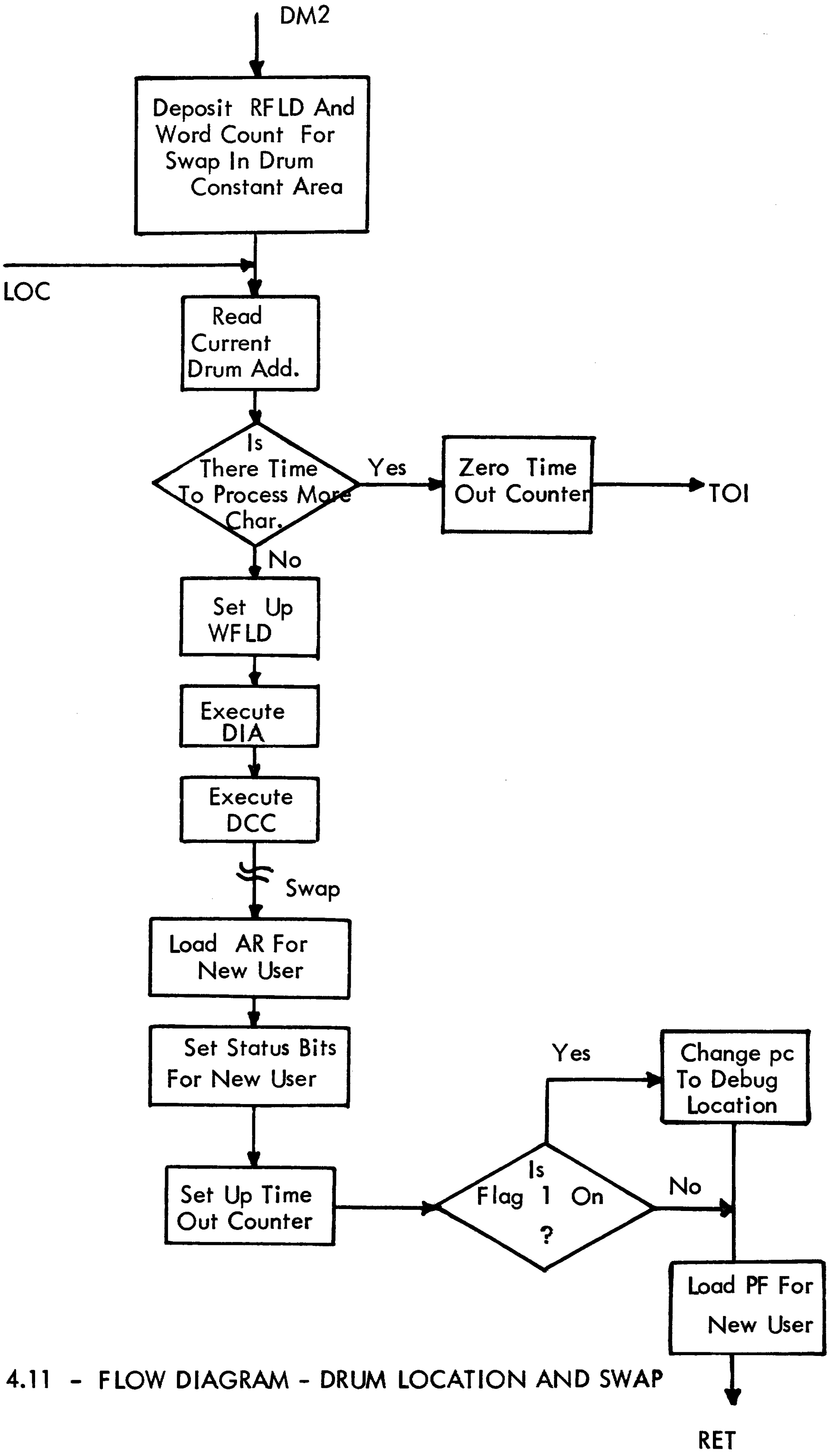


FIG. 4.11 - FLOW DIAGRAM - DRUM LOCATION AND SWAP

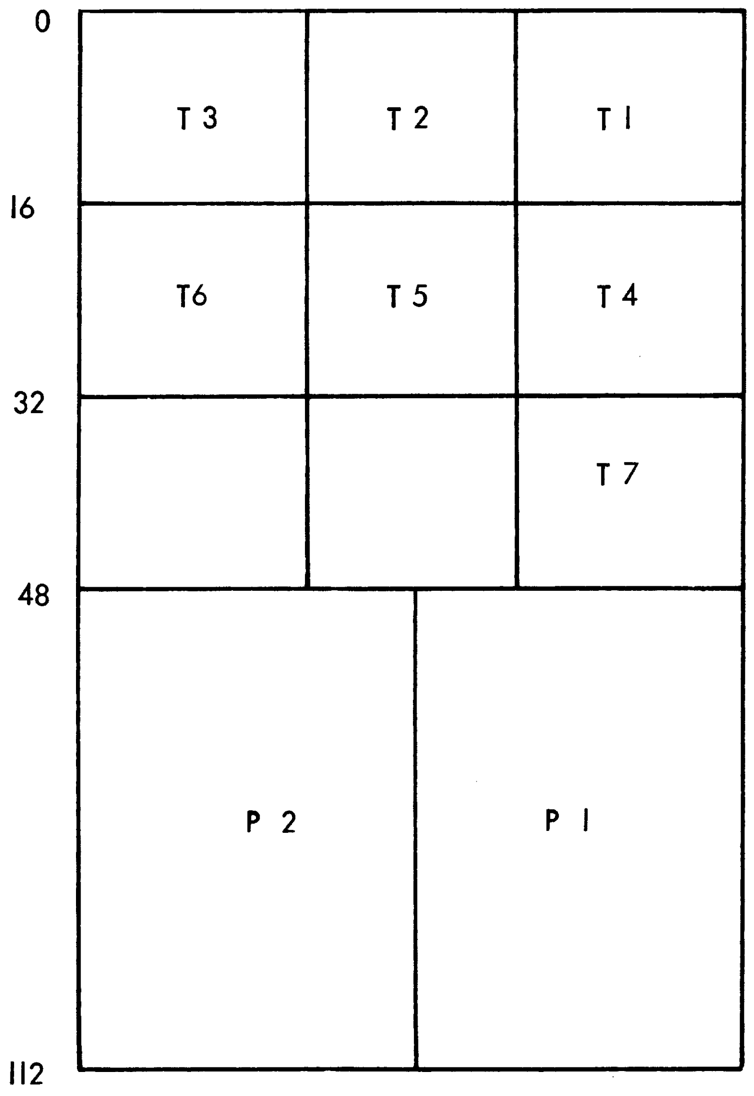


FIG. 4.12 - EXECUTIVE ROUTINE BUFFER STORAGE

When it is determined there is not enough time to process more characters, the actual drum transfer takes place. After the transfer all pertinent registers and status bits are set with the new information before returning to the users program.

B. FUTURE MODIFICATIONS TO THE EXECUTIVE ROUTINE

This version of the ER was written for only two typewriters and one punch. It is easily expandable to seven typewriters and two punches by just extending the tables and the character buffers. However, for the larger system several economies may result from a different attack on the programming for the character buffers. Right now the buffers are 16 and 64 long, taking 96 registers. With seven typewriters and two punches this would take 240 registers.A great saving can be made by "packing" the buffers. This puts three characters in a single address location by utilizing a subprogram to arrange them accordingly (see Fig. 4. 12). This would save 128 registers. The packing and unpacking programs should take about 20-30 registers, resulting in a large saving of memory.

C. ADMINISTRATIVE ROUTINE

A program for the administrative has not yet been written, however its functions can be explained. When a user first requests service by turning his switch ON, the ER calls the administrative routine (called AR in this section only, not to be confused with the assignment register). The AR processes any requests for external IO equipment; sets the deb location to the field, depending on where the user wants his button to transfer, assigns a maximum length of time allowed each user by setting location n in the user's executive area; inserts any optional versions of the ER into the user's executive area; and sets up the trap dispatch table according to the user's request.

When it is called by a button or switch during a program, the AR acts as a sophisticated debugging routine. It must read areas of the user's program on the drum to determine the place of error, and communicate with the user thru the typewriter. Only one AR is provided, so it must contain provisions for handling several programs at once. If two successive user's want the AR, it does not swap, but changes register n and transfers to the section pertaining to the other user. (By setting the memory protection to 7777, the AR may change any ER registers it desires).

The third use of the AR is to determine the reasons for traps due to illegal commands and print out an error comment to the user. It then stands ready to make any corrections the user dictates, or to call in the user's debugging routine if so directed.

V APPENDIX

A. PDP-1 INSTRUCTION LIST

1. Basic Instructions

In the list below, (see Table 5.1) C(Y) refers to the contents of memory at address Y, C(AC) and C(IO) refer to the contents of the AC and IO respectively. The operating time is given assuming no indirect addressing. The C(Y) is unchanged uless the instruction is a deposit or index.

Mnemonic Code	Octal Code	Operation	Operation Time (μsec)
add Y	40	Add C(Y) to C(AC)	10
and Y	02	Logical AND of C(Y) with C(AC)	10
cal Y	16	Equals JDA 100	10
dac Y	24	Deposit C(AC) in Y	10
dap Y	26	Deposit address part of AC in Y	10
dio Y	32	Deposit C(IO) in Y	10
dip Y	30	Deposit instruction part of AC in Y	10
dis Y	56	Divide step	10
dzm Y	34	Make C(Y) zero	10
idx Y	44	Add one to C(Y). Leave in Y and AC	10
ior Y	04	Inclusive OR of C(Y) with C(AC)	10
iot	72	See In-Out Transfer Group	—
isp Y	46	IDX and skip if positive	10
jda Y	17	Equals DAC Y plus JSP (Y+1)	10
jfd Y	12	Jump memory field according to C(Y)	10
jmp Y	60	Take next instruction from Y	5
jsp Y	62	Jump to Y and save PC in AC	5
lac Y	20	Load AC with C(Y)	10
law n	70	Load AC with the number N	5
law-N	71	Load AC with the number - N	5
lio Y	22	Load IO with C(Y)	10
mus Y	54	Multiply step	10
opr	76	See Operate Group	5

Code		Operation	(μsec)
sad Y	50	Skip next instruction if C(AC) differs from C(Y)	10
sas Y	52	Skip next instruction if C(AC) is same as C(Y)	10
shift	66	See Shift group	5
skp	64	See Skip Group	5
sub	42	Subtract C(Y) from C(AC)	10
xet Y	10	Perform instruction in Y	5+
xor Y	06	Exclusive OR of C(Y) with C(AC)	10

2. Operate Group

The operate instruction group performs miscellaneous operations on various Central Processer Re gisters. The address portion specifies the operation to be performed. These instructions are a micro set and can be OR'ed together to give the union of the function. Thus opr 3200 will clear the AC, put TW to AC, and complement the AC all in one cycle.

cla	762000	Clear AC
clf	76000f	Clear program flag f Clear all flags on f=7
cli	764000	Clear IO
cma	761000	Compliment AC
hlt	760400	Halt
lat	760200	Inclusive OR of TW switches with C(AC)
nop	760000	No operation
stf	76001f	Set program flag f Set all flags on f=7

3. Shift Group

Shift is an arithmetic operation. The sign bit is left unchanged and vacated bits are filled with the sign. Rotate is a logical operation and cycles the bits (including sign) in a closed ring. The number of steps is the number of ONE's in bits 9-17 of the instruction.

ral	661	Rotate AC left
rar	671	Rotate AC right
rcl	663	Rotate combined AC and IO left
rcr	673	Rotate combined AC and IO right
ril	662	Rotate IO left
rir	672	Rotate IO right
sal	665	Shift AC left
sar	675	Shift AC right
scl	667	Shift combined AC and IO left
scr	677	Shift combined AC and IO right
sil	666	Shift IO left
sir	676	Shift IO right

4. Skip Group

The skip group allows skips on various conditions in the machine. selected by the address portion of the instruction. This is also a micro set and can be combined to form the inclusive OR of the separate skips. Thus skip 3000 will skip on AC overflow or IO positive, still in 5 microseconds. If bit 5 is a ONE, the instruction becomes Do Not Skip if the indicated condition is present, Skip if it is absent.

sma	640400	Skip on minus AC (sign =1)
spa	640200	Skip on plus AC (sign = 0)
spi	642000	Skip on plus IO (sign = 0)
sza	640100	Skip on ZERO (+0) AC
szf	64000f	Skip on ZERO flag f (f=7is allflags)
szo	641000	Skip on ZERO overflow and clear overflow
szs	640050	Skip on ZERO sense switches (s=7is all switches)

TABLE 5.1 - PDP-1 INSTRUCTION LIST

MBD_F \ MBD_E	0	1	2	3	4	5	6	7
0	*		AND		IOR		XOR	
1	XCT		JFD		*		CAL	JDA
2	LAC		LIO		DAC		DAP	
3	DIP		DIO		DZM		*	
4	ADD		SUB		IDX		ISP	
5	SAD		SAS		MUS		DIS	
6	JMP		JSP		SKP		SHIFT	
7	LAW	LAW	IOT		*		OPR	

* = Illegal Op Code

MBD_B \ MBD_A	0	1	2	3	4	5	6	7
0		RPA	RPB	TYO	TYI	PPA	PPB	DPY
1	← USER'S PRIVATE IOT'S →							
2								
3	RRB			CKS				
4								
5	DSC	ASC	ISB	CSB	LSM	ESM		
6	DIA	DBA	DCC	DRA				
7								EXC

TABLE 5.2 - BASIC IN-OUT TRANSFER GROUP

5. In-Out Transfer Group

The variation within this group of instruction performs all the in-out control and information transfer functions. If bit 5 is a ONE, the computer will halt and wait for a completion pulse. Bit 6 determines whether a completion pulse will or will not be received from the in-out device. When it is the same as bit 5, a completion pulse will not be received. When different, it will be received. Bits 7-11 can be used to further decode a an iot instruction (see Table 5. 2).

asc	720051	Activate (turn on) SBS channel n
cks	730033	Check status. Set IO bits as follows
		Bit — If a ONE, indicates
		0 — Displayed point sensed by light pen
		1 — Paper tape reader busy
		2 — Typewriter busy (TYO)
		3 — Typewriten key struck (reset by TYI)
		4 — Paper tape punch busy
csb	720053	Clear a break request on channel n
dba	720061	Break on drum address given in IO
dcc	720062	Deposit count as follows and continue
		IO 1-5 drum read field
		IO 6-17 number of words transferred
		AC 6-17 memory core initial address
		AC 1-5 memory field
dia	720060	Deposit initial drum address from IO bits 6-17. Drum write field from IO 1-5
dpy	730007	Display one point on CRT. AC bits 0-9 are x coordinate, IO bits 0-9 are y coordinate
dra	720063	Read drum address into IO
dsc	720050	Deactivate (turn off) SBS channel n
esm	720055	Enter sequence break mode
exc	72xx77	Executive routine commands for time sharing system

isb	720052	Initiate a break on channel n
lsm	720054	Leave sequence break mode
ppa	720005	Punch paper tape alphanumeric from IO bits 10-17
ppb	720006	Punch paper tape binary from IO bits 0-5
rpa	720001	Read paper tape alphanumeric into IO bits 10-17
rpb	720002	Read paper tape binary. Read 3 punched lines into IO bits 0-17
vib	720030	Read reader buffer. In sequence break mode, rpa and rpb do not read the reader buffer into the IO
tyi	720004	Read the typewriter buffer into IO bits 12-17. To determine when information is available, CKS and PF1 are set when a key is struck
tyo	720003	Type out from IO bits 12-17

B. TIME SHARING INSTRUCTION LIST

All instructions added for the time sharing system are a subgroup of the iot transfer group. Iot 77 is the EXC group (72xx77) which is decoded as follows. All instruction take 5 microseconds (see Table 5. 3)

Mnemonic Code	Octal Code (EXC xx)	Operation
arq	31	Bits in the AC are interperted to be equipment request by the administrative routine
bpt	30	Location of a break point inserted by the administrative routine
kng	25	Turn on HG flip flop and execute a wait in ER
lar	14	Load assignment register from IO_{0-18}
lbc	20	Load (turn on) break channels from IO_{0-15}
lcn	11	Load CNR (Console Number Register) from IO_{0-3}
lky	04	Turn on LK_n; n contained in CNR. Lock keyboard

	0	1	2	3	4	5	6	7
0		PSO	PSF	SPS	LKY	ULK	SLK	SDY
1	RCN	LCN		RBU	LAR	RPF	LPF	SCN
2	LBC	RWB	BCN	BCF		HNG	UHG	SHG
3	BPT	ARQ	PUN	PUF	TON	TOF	TIN	TIN

$$(MBD_F)(MBD_E)(MBD_D)(MBD_C)(MBD_B)(MBD_A) = 72XX77$$

TABLE 5.3 - EXECUTIVE INSTRUCTIONS

lpf	16	Load program flags from IO_{12-17}
psf	02	Turn off PS_n ; n contained in CNR
pso	01	Turn on PS_n ; n contained in CNR
puf	33	Turn off punch status bit
pun	32	Turn on punch status bit
rbu	13	Read console buttons into IO_{0-6} and console switches into IO_{9-15}
rcn	10	Read next console wanting service Store in CNR and IO_{0-3}
rpf	15	Read program flags into IO_{12-17}
rwb	21	Read SBS Wait FF into IO_{0-15}
scn	17	Set CNR from bits 0-3 of AR
sdy	07	Skip next instruction if console display button is ON.
shg	27	Skip next instruction if HG flip flop is ON
slk	06	Skip next instruction if LK_n is ON; n contained in CNR
sps	03	Skip next instruction PS_n is ON; n contained in CNR
tif	37	Turn off type in status bit
tin	36	Turn on type in status bit
tof	35	Turn off type out status bit
ton	34	Turn on type out status bit
uhg	27	Turn off HG flip flop
ulk	05	Turn off LK_n ; n contained in CNR. Unlock keyboard

C. EXECUTIVE ROUTINE PROGRAM

The following pages contain a sample executive routine program. It is written, using the PDP-1 instructions and the new instructionsadded for the time sharing system, in a programming language presently in use at M. I. T. called MACRO. A few conventions of this language are

1. A space is taken as a plus sign.
2. Tab and carriage return are terminating characters.

3. A comma defines the characters typed since the last terminator as a symbol having the value of the current location. Symbols must have three or less characters, at least one of which is a letter.

4. (period) is a symbol meaning the current location. It is also used when the address is to be filled in by program during execution.

5. The symbol "|" after a terminator defines the materials following it as a comment which is ignored by MACRO. The comment continues until the next tab or carriage return.

6. MACRO allows what are called "macro instructions" to be used. These are short, frequently used sequences which can be defined by a name of from 4 to 6 symbols. Whenever this name is encountered in the chain of commands, the instructions this name represents are automatically inserted.

7. The character "|" after a group of characters defines the current location indicator to be equal to the value of the symbol.

8. MACRO interprets the symbol "ns" after a shift or rotate instruction to mean shift or rotate n times and automatically sets the correct bits in the address section.

This program is a sample only. As the hardware is not fully installed the program has not been tested for logical errors although every effort has been made to make it a correct operating program. It has been assembled and checked for typing mistakes however.

D. SYSTEM DRAWINGS

1. DEC Logical Notation

The logical notation used by Digital Equipment Corporation is very useful in designing a system this large, especially when NOR and NAND circuits are involved. These are shown in Fig. 5. 1a. An inverter (with all its associated biases) is a rectangular box with the three leads. These are PNP transistors. A resistor with a large solid dot is a standard DEC clamped load resistor. Levels are indicated by diamonds -- an open diamond indicates the level is true when the level is at ground, a closed diamond indicates assertion when the levelis at - 3 volts. Pulses are indicated by open and closed arrows. An open arrow is a positive pulse (from ground) and a closed arrow is a negative pulse (from ground). Arrowheads or diamonds are used at all branch points to indicate signal

(continued
on page 78)

TIME SHARING EXECUTIVE ROUTINE

```
/This program for the executive routine of the time sharing
/system assumes two typewriters, one punch and one reader.
/The buffers in the executive routine for input and output
/are not packed - i.e., they are located in succession in
/memory.  Three tables of pointers called the ADM pointers
/are included to facilitate use of these buffers.  The
/buffers are of the ring type.  ADM(1,n) points to the
/position in the n th buffer where the next character is
/to be placed.  Bit 0 of this pointer is a ONE if the
/buffer is full, and a ZERO if not.  ADM(2,n) points to the
/position in the n th buffer where the next character of
/output is to be taken from.  Bit 0 of this pointer is a
/ONE if the buffer is empty and a ZERO if it is not empty.
/ADM(3,n) points to the location after the last location of
/the n th buffer (this turns out to be the first location
/of the next buffer).

/The buffers fld thru fld+3 contain the information on
/whether to bring the user in on his next time around.
/Bit 0 being a ONE indicates the program is still active.
/Bit 1 being a ONE indicates a button has been pushed.
/Bit 2 being a ONE indicates a switch has been turned on.
```

```
/EXECUTIVE ROUTINE PROGRAM FOR TIME SHARING SYSTEM

/Coding of new Executive instructions.


exc=720077
pso=exc 100
psf=exc 200
sps=exc 300
lky=exc 400
ulk=exc 500
slk=exc 600
sdy=exc 700
rcn=exc 1000
lcn=exc 1100
rbu=exc 1300
lar=exc 1400
rpf=exc 1500
lpf=exc 1600
scn=exc 1700
lbc=exc 2000
rwb=exc 2100
bcn=exc 2200
bcf=exc 2300
hng=exc 2500
uhg=exc 2600
shg=exc 2700
bpt=exc 3000
arq=exc 3100
puf=exc 3200
pun=exc 3300
tof=exc 3400
ton=exc 3500
tif=exc 3600
tin=exc 3700
dia=iot 60
dba=iot 61
dcc=iot 62
dra=iot 63


define  swap
        rcr 9s
        rcr 9s
        terminate
```

```
/Area swapped in a drum transfer


/drum transfer traps 7, 8, 9, 10.  If the drum is not
/assigned to the user, these go to the administrative
/routine.  If it is, proper adjustment must be made in
/the arguments to account for time spent in the executive
/routine and different field assignments.  This is an
/optional part of the executive routine.

6737/
/dia routine, trap 7

di,     xx                       /to be programmed


/dba routine, trap 8

db,     xx                       /to be programmed


/dcc routine, trap 9

dc,     xx                       /to be programmed


/dra routine, trap 10

dr,     xx                       /to be programmed

/display routine for a lengthened quantum, optional

dy,     sad dy1                  /is this a first request?
        jmp .+3                  /yes, grant request
        dzm dy1                  /no, dismiss,reset indicator
        jmp dm0
        idx dy1                  /set indicator indicating grant
        lac dyn
        dac toc                  /extend quantum
        jmp ret                  /return to user
dyn,    -30                      /30 more time periods
dy1,    0                        /indicator
```

```
/trap dispatch table, set up by administrative routine

tdt,    jmp adm         /illegal commands
        jmp hg          /hng
        jmp tpo         /tyo
        jmp tpi         /tyi
        jmp pua         /ppa or jmp adm if not assigned
        jmp pub         /ppb or jmp adm if not assigned
        jmp di          /dia or jmp adm if not assigned
        jmp db          /dba or jmp adm if not assigned
        jmp dc          /dcc or jmp adm if not assigned
        jmp dr          /dra or jmp adm if not assigned
        jmp adm         /bpt
        jmp adm         /arq


/various constants and storage areas

n,      77715           /time out counter constant
ar,     0               /assignment register
pf,     0               /program flags


/storage for registers on trap

ac,     0
pc,     0
io,     0


/Area not swapped in a drum transfer

/main executive routine

7000/
beg,    sza i           /test for time out
        jmp toi         /if time out
        dac trn         /save trap number
        add (tdt-1      /dispatch on trap number
        dap .+1
        jmp .
trn,    0
```

```
/time out routine, trap 0

        dzm toc          /character processing only
toi,    rcn              /read console number
        rcl 4s           /shift to AC
        sza              /test for no console
        jmp tos          /a console wants service
        isp toc          /index time out counter
        jmp ret          /return to users program
        sza i            /is quantum just up?
        sdy
        jmp dm0          /dismiss user
        jmp dy           /display routine for more time,
                         /or jmp dm0 if not assigned
toc,    0                /time out counter


/service to a console during a time out

tos,    rcr 4s
        jsp sat          /set up ADM tables
        lio i adb        /get character from buffer
        lac sa1          /test for punch
        spa              /punches are greater than 10
        jmp ty           /typewriter service
        ppa              /punch contents of buffer
        puf              /buffer cannot be full
agn,    lac ada
        and (377777      /clear full indicator
        dac i a
        jsp ite          /index ADM table and test if empty
        spa              /skip if not empty
        psf              /empty, turn off PS
        jmp toi          /process next character

ty,     sps
        jmp ti           /tyi if PS is off
        tyo              /execute command
        tof              /buffer cannot be full
        lky              /no reason to grant type in
        jmp agn

ti,     tyi              /execute command
        dio i ada        /deposit in buffer
        jsp itf          /index ADM and test if full
        spa              /skip if not full
        lky
        tin              /buffer cannot be empty
        lac adb
        and (377777      /clear empty indicator
        dac i b
        jmp toi          /process next character
```

```
/set up ADM tables

sat,    dap sax             /for return
        cla
        rcl 4s              /get CN into AC
        sub (10             /test for punches
        dac sa1
        swap
        lac (ln2            /load for punch
        spi
        lac (ln1            /no punch, load for type
        dap itf 3
        dap ite 3
        lac sa1
        add (ad1-1 10
        dap a               /address of adm(1,n)
        add (ad2-ad1
        dap b               /address of adm(2,n)
        add (ad3-ad2
        dap c               /address of adm(3,n)
b,      lac .
        dac adb             /adm(2,n)
c,      lac .
        dac adc             /adm(3,n)
a,      lac .
        dac ada             /adm(1,n)
sax,    jmp .               /return with ad1 in ac
ada,    0                   /current ad1
adb,    0                   /current ad2
adc,    0                   /current ad3
sa1,    0                   /CN - 10


/index adm table and test for full buffer

itf,    dap itr             /for return
        idx ada             /new input pointer
        sad adc             /test for bottom of buffer
        sub .               /set pointer at top of buffer
        sad adb             /check for full buffer
        ior (400000         /set indicator for full buffer
        dac ada             /deposit for use
        dac i a             /deposit in adm(1,n)
itr,    jmp .               /return
```

```
/index ADM table and test for empty buffer

ite,    dap itx         /for return
        idx adb         /new output pointer
        sad adc         /test for bottom of buffer
        sub .           /set pointer at top of buffer
        sad ada         /check for empty buffer
        ior (400000     /set indicator for empty buffer
        dac adb         /deposit for use
        dac i b         /deposit in adm(2,n)
itx,    jmp .           /return


/return to users program

ret,    lio io
        lac ac
        jmp i pc


/hng routine, trap 2

hg,     jmp .



/tyo routine, trap 3

tpo,    scn             /get console number
        jsp sat         /set up adm tables and return
        spa             /with ad1 in the ac
        jmp dms         /buffer is full
        slk i           /is this a first type out
                        /after a type in?
        jmp tp1         /no
        lky             /yes, lock keyboard
        lac adb
        dac ada         /set input pointer = output pointer
tp1,    lio io
        dio i ada       /store character in buffer
        pso             /turn on print status
        jsp itf         /index adm and test if full
        spa
        ton             /buffer full
        jmp ret+1       /return to users prog
```

```
/tyi routine, trap 4

tpi,    scn             /get console number
        jsp sat         /set up adm tables
        lac adb
        sma             /is buffer empty?
        jmp tp2         /no
        psf             /buffer empty, prepare to
        ulk             /accept characters and
        jmp dms         /wait
tp2,    sps i           /did character come from type in?
        jmp dms         /no, from type out
        lio i adb
        jsp ite         /index adm and test if empty
        spa
        tif             /buffer empty
        jmp ret+1       /return to users prog
```

```
/ppa routine, trap 5

pua,    lio io
        dio pp1         /save character to be punched
        lio (400000     /punch 0 = console 8
        lcn             /set console number
        jsp sat         /set up adm table and return
        spa             /with ad1 in ac
        jmp dms         /buffer is full
        pso
        lio pp1         /get character
        dio i ada       /store in buffer
        jsp itf         /index adm and test if full
        spa
        pun             /buffer full
        jmp ret+1       /return to users program
pp1,    0               /storage for punch word
```

```
/ppb routine, trap 6

pub,    lio io          /get character
        cla
        rcl 6s          /shift to bits 12-17
        ior (200        /insert bit 10
        dac pp1         /deposit character in ppa format
        jmp pua+2       /pick up ppa routine
```

```
/dismiss routine

dm0,    sza
        jmp loc
dms,    rpf                     /save program flags
        dio pf
        clf 7                   /clear all flags for exc. use
        rbu
        lac but
        dio but
        cma
        and but                 /ac contains new button and
                                /switch requirements
        sza i
        jmp dm1                 /no new requests
        lio (500000             /indicator for button
tst,    spa
        dio fld 1               /set c1 indicator
        ral 1
        spa
        dio fld 2               /set c2 indicator
        ral 1s
        sar 2s                  /clear button indicators
        sza i
        jmp dm1                 /no switch requests
        ral 9s
        lio (440000             /indicator for switches
        jmp tst
dm1,    lac fdp
        dac dil                 /set up WFLD
dm3,    idx fdp
        and (3                  /determine next field modulo 4
        dap fdp
        sad dil
        jmp bck                 /no other users, is this one
                                /still valid?
        add (fld
        dap .+1
dm4,    lac .                   /get info. for next console
        sma
        jmp dm3                 /prog. waiting for user response
        rcl 1s
        sza                     /non zero if switch or button,
                                /assume not both
        jmp bug                 /what should they do?
        lac fdp                 /bring next program
        rar 6s                  /set up RFLD
        ior (beg-1              /word count for normal swap
dm2,    dac dci
loc,    dra
        swap             /present drum location to AC
        sub (5000        /depends on length of ER
        spa
        jmp toi-1        /time for more char. processing
        lio dil          /set up WFLD
        rir 6s
        dia              /start drum transfer
        lio dci
        lac dca
        dcc
        lio ar
        lar              /load assignment reg.
        scn
        tin
        tof
        puf
        jsp sat          /set up ADM tables for new user
        spa
        ton              /his buffer is still full
        lac adb
        spa
        tif              /his buffer is empty
        lio (400000      /set for punch
        lcn
        jsp sat          /set up ADM table for punch
        spa
        pun              /punch buffer is still full
        lac n            /set counter
        dac toc
        szf i 1
        jmp .+4          /return to normal program
        idx bg1
        xct bg1          /return to debug in prog.
        dap pc
        lio pf
        lpf              /load program flags
        jmp toi          /serve any consoles
                         /desiring service


bck,    lac toc
        sza
        jmp toi          /all prog. waiting, process
                         /more  characters
        jmp ret          /assume 0 toc means he
                         /still wants back
```

```
bug,    lio (400000
        dio i dm4       /reset request indicator
        spa i
        jmp adm         /switch request
        lac fdp
        sal 1s          /multiply by 2
        add (deb-2
        dap .+1
bg1,    lac .
        dac dci
        cli
        rcr 6s
        rcr 6s          /get field only in AC
        sad fdp         /if field wanted is users field,
        stf 1           /he wants own debug routine
        jmp dm2+1
deb,    0               /drum con for error, by adm rout
        0               /entry pt. in users program
        0               /drum con for error, by adm rout
        0               /entry pt. in users program
adm,    lac (036777     /administrative routine wanted
        jmp dm2
fld,    0               /next console info., bit 0
                        /set by adm routine if required
        0
        0
        0
fdp,    0               /this console pointer
dci,    0               /dcc I/O read field and word count
dca,    0               /dcc AC mem. field and core address
dil,    0               /dia init. loc. and write field
but,    0               /storage for button and
                        /switch calls


/ADM tables and character buffers

ad1,    bf1             /adm(1,1)
        bf2             /adm(1,2)
        bf8             /adm(1,8)
ad2,    bf1             /adm(2,1)
        bf2             /adm(2,2)
        bf8             /adm(2,8)
ad3,    bf1 ln1         /adm(3,1)
        bf2 ln1         /adm(3,2)
        bf8 ln2         /adm(3,8)


ln1=20                  /length of type buffers
ln2=100                 /length of punch buffers
```

```
bf1,
bf1+ln1/
bf2,
bf2+ln1/
bf8,
bf8+ln2/


constants


start beg
```

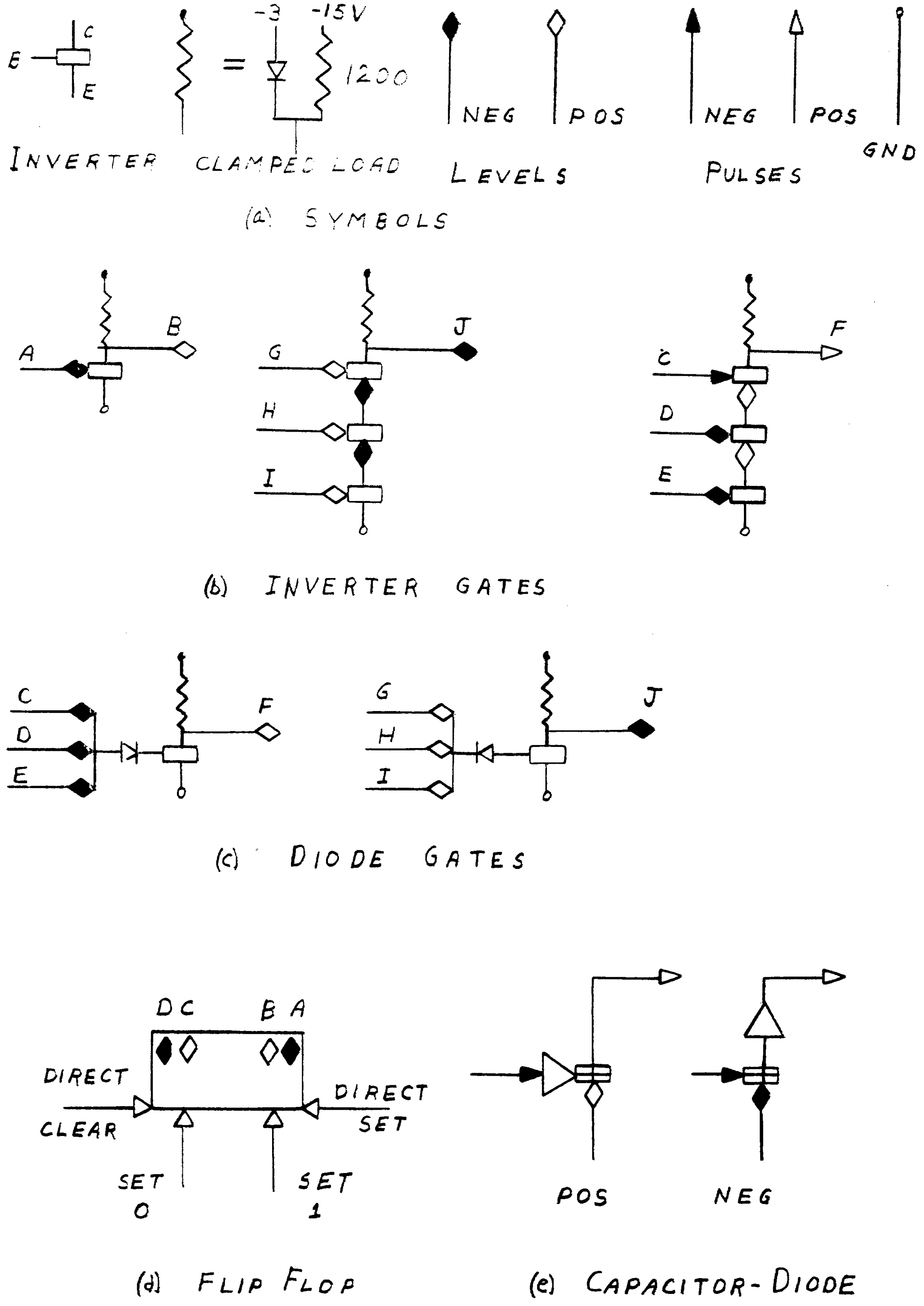
flow. An open circle is a ground connection.

The inverter is analogous to a simple mechanical switch. A true input (-3v) will short the collector and emitter. When A is true B will be true (GND) and vice versa. When stacked in series all inputs must be true for a true output. Thus F=CDE and J=G+H+I. (Fig. 5. 1b). Diode gates are represented as in Fig. 5. 1c. Both positive and negative diode gates are used as indicated. Flip flops are indicated by a large rectangle (Fig. 5. 1d). The flip flop is set or cleared by momentarily grounding the appropriate set line thru an inverter, or by pulsing the direct clear or set lines with a DEC positive pulse. The output is indicated by four diamonds. In the ONE state, A will be -3volts, B will by GND, C will be -3volts and D will be GND. In the ZERO state, all voltages are reversed. Actually, A and C are the same pin connection on the flip flop, as are B and D.

Another type of logic is the capacitor diode logic of Fig. 5. 1e. Again two types of gates exist, positive and negative depending on the polarity of the enabling level. In either case the output will be a positive going pulse relative to the enabling level. The flip flops use the same notation as before, but are slightly different in their construction. The regular have buffered outputs which give a large driving capability and have built in delay so the output of a flip flop may be sampled at the same time it is being changed. The C-D flip flops on the other hand, have no delay. However, they may still be sampled at change time if they are driven from C-D gates. There is a certain charge and discharge time involved in them which causes delay in actually setting the flip flop. These have less driving capability however.

2. Block Schematics

Each bay can hold twelve 19 inch mounting panels, each capable of containing 25 system modules. Following the DEC Convention, the panels are lettered from top to bottom Y, Z, A, B, C, D, E, F, H, J, K and L. Each connector is numbered from 1 to 25 left to right as seen from the wiring side. The 22 pins on each connector are lettered A to Z, (except for G, I, O and Q), top to bottom. Thus the location of each pin in the system can be specified uniquely, e. g., 4 E15 T -- pin T on connector 15 of panel E in bay 4. In the drawings which follow the hardware



(a) SYMBOLS

(b) INVERTER GATES

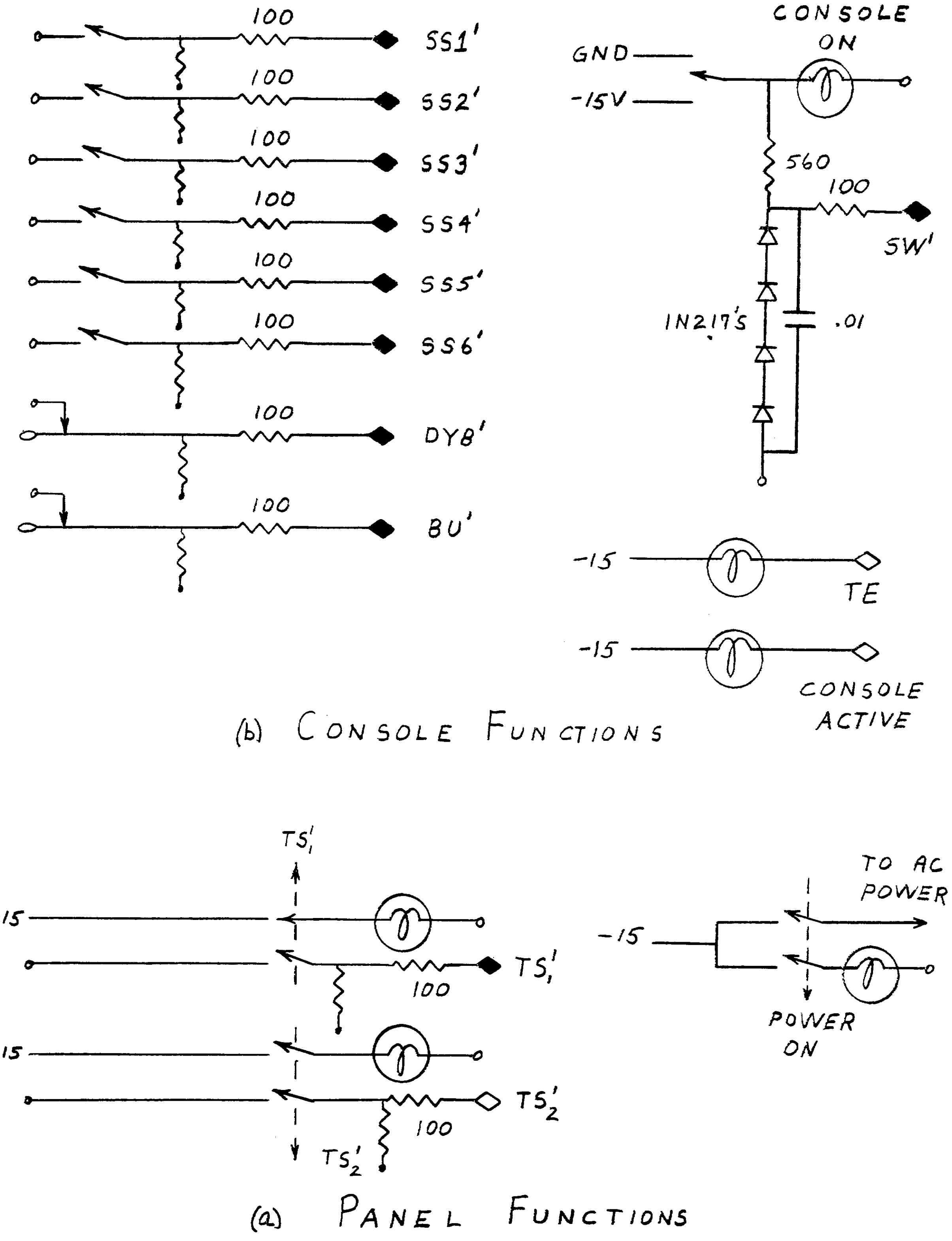
(c) DIODE GATES

(d) FLIP FLOP

(e) CAPACITOR-DIODE

FIG. 5.1 - LOGICAL SYMBOLS

has been grouped into sections outlined by a dotted line. Within each outline there are two numbers, $\overset{4603}{\text{3J22}}$. The upper number is the module model type and the lower number is its location in the computer. Pin numbers are designed by the letters near each wire within the outline. Figures 5.2 thru 5.11 contain the block schematics for the time sharing system.



(b) Console Functions

(a) Panel Functions

FIG. 5.2 - CONTROL FUNCTIONS

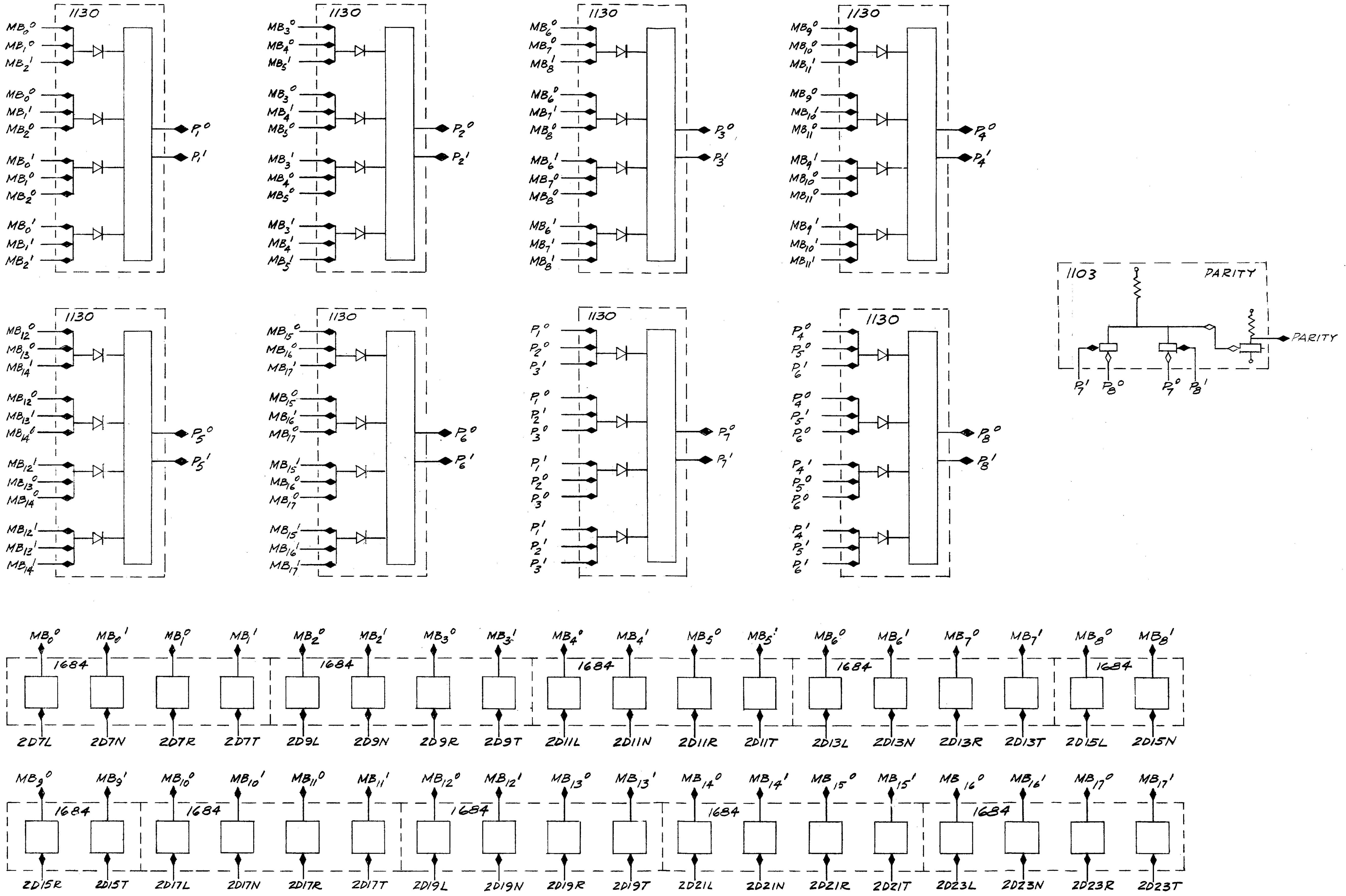
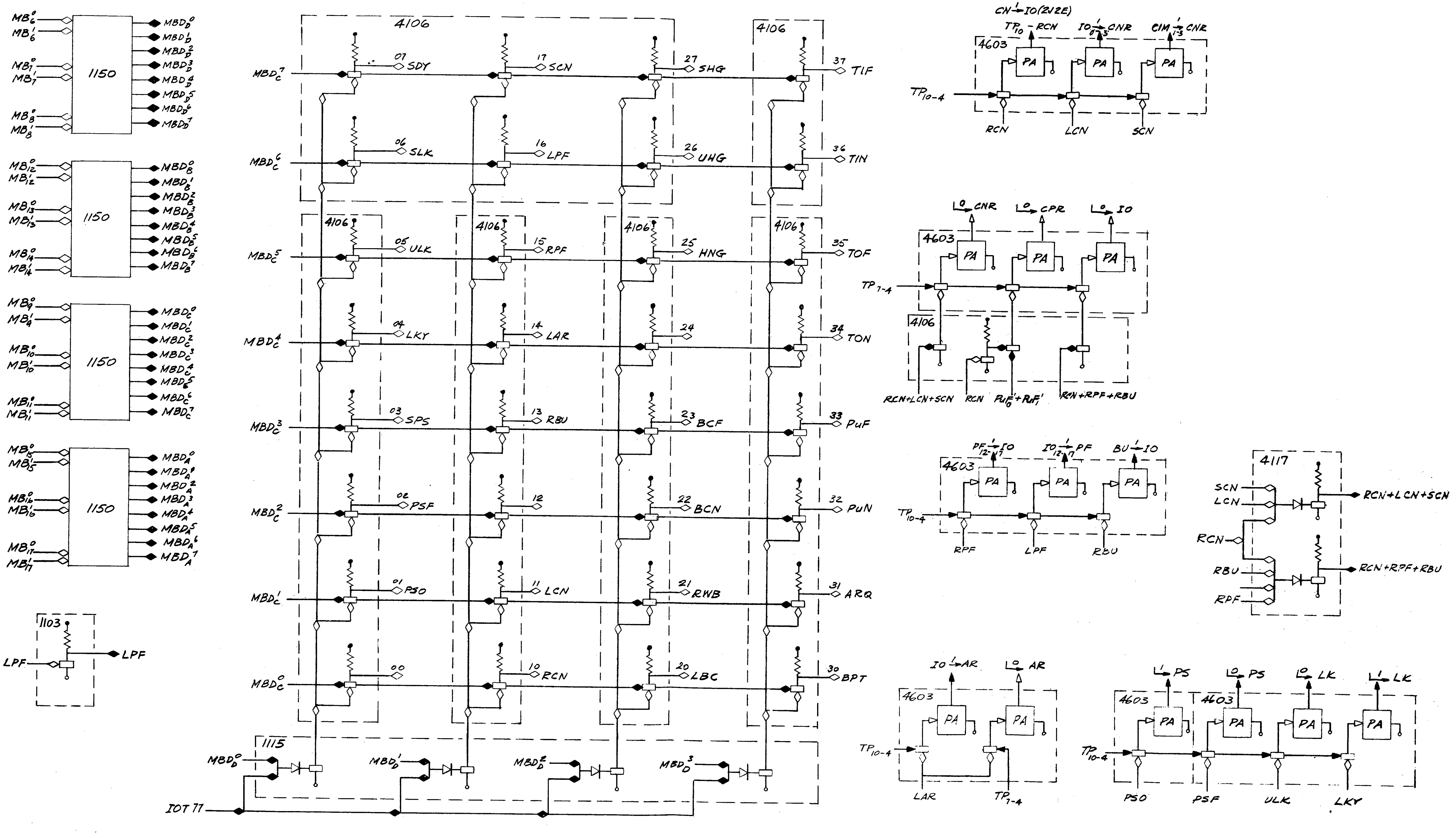


FIG. 5.3 - MB BUFFERS AND PARITY



1150
MB_6^0 MB_6^1 MB_7^0 MB_7^1 MB_8^0 MB_8^1
MBD_D^0 MBD_D^1 MBD_D^2 MBD_D^3 MBD_D^4 MBD_D^5 MBD_D^6 MBD_D^7
MB_{12}^0 MB_{12}^1 MB_{13}^0 MB_{13}^1 MB_{14}^0 MB_{14}^1
MBD_B^0 MBD_B^1 MBD_B^2 MBD_B^3 MBD_B^4 MBD_B^5 MBD_B^6 MBD_B^7
MB_9^0 MB_9^1 MB_{10}^0 MB_{10}^1 MB_{11}^0 MB_{11}^1
MBD_C^0 MBD_C^1 MBD_C^2 MBD_C^3 MBD_C^4 MBD_C^5 MBD_C^6 MBD_C^7
MB_{15}^0 MB_{15}^1 MB_{16}^0 MB_{16}^1 MB_{17}^0 MB_{17}^1
MBD_A^0 MBD_A^1 MBD_A^2 MBD_A^3 MBD_A^4 MBD_A^5 MBD_A^6 MBD_A^7
1103
LPF
LPF
4106
07 SDY
17 SCN
27 SHG
37 TIF
06 SLK
16 LPF
26 UHG
36 TIN
05 ULK
15 RPF
25 HNG
35 TOF
04 LKY
14 LAR
24
34 TON
03 SPS
13 RBU
23 BCF
33 PuF
02 PSF
12
22 BCN
32 PuN
01 PSO
11 LCN
21 RWB
31 ARQ
00
10 RCN
20 LBC
30 BPT
1115
MBD_D^0 MBD_D^1 MBD_D^2 MBD_D^3
IOT 77
$CN \xrightarrow{1} IO (2J2E)$
TP_{10} - RCN
$IO_{0-5} \xrightarrow{1} CNR$
$CIM_{1-3} \xrightarrow{1} CNR$
4603
PA
TP_{10-4}
RCN
LCN
SCN
$\xrightarrow{0} CNR$
$\xrightarrow{0} CPR$
$\xrightarrow{0} IO$
TP_{7-4}
RCN+LCN+SCN
RCN
$PuF_0'+PuF_1'$
RCN+RPF+RBU
$PF_{12-17} \xrightarrow{1} IO$
$IO_{12-17} \xrightarrow{1} PF$
$BU \xrightarrow{1} IO$
RPF
LPF
RBU
4117
SCN
LCN
RCN
RBU
RPF
RCN+LCN+SCN
RCN+RPF+RBU
$IO \xrightarrow{1} AR$
$\xrightarrow{0} AR$
LAR
TP_{7-4}
$\xrightarrow{1} PS$
$\xrightarrow{0} PS$
$\xrightarrow{0} LK$
$\xrightarrow{1} LK$
PSO
PSF
ULK
LKY

FIG. 5.4 - MB DECODER, INSTRUCTION DECODE AND TIMING

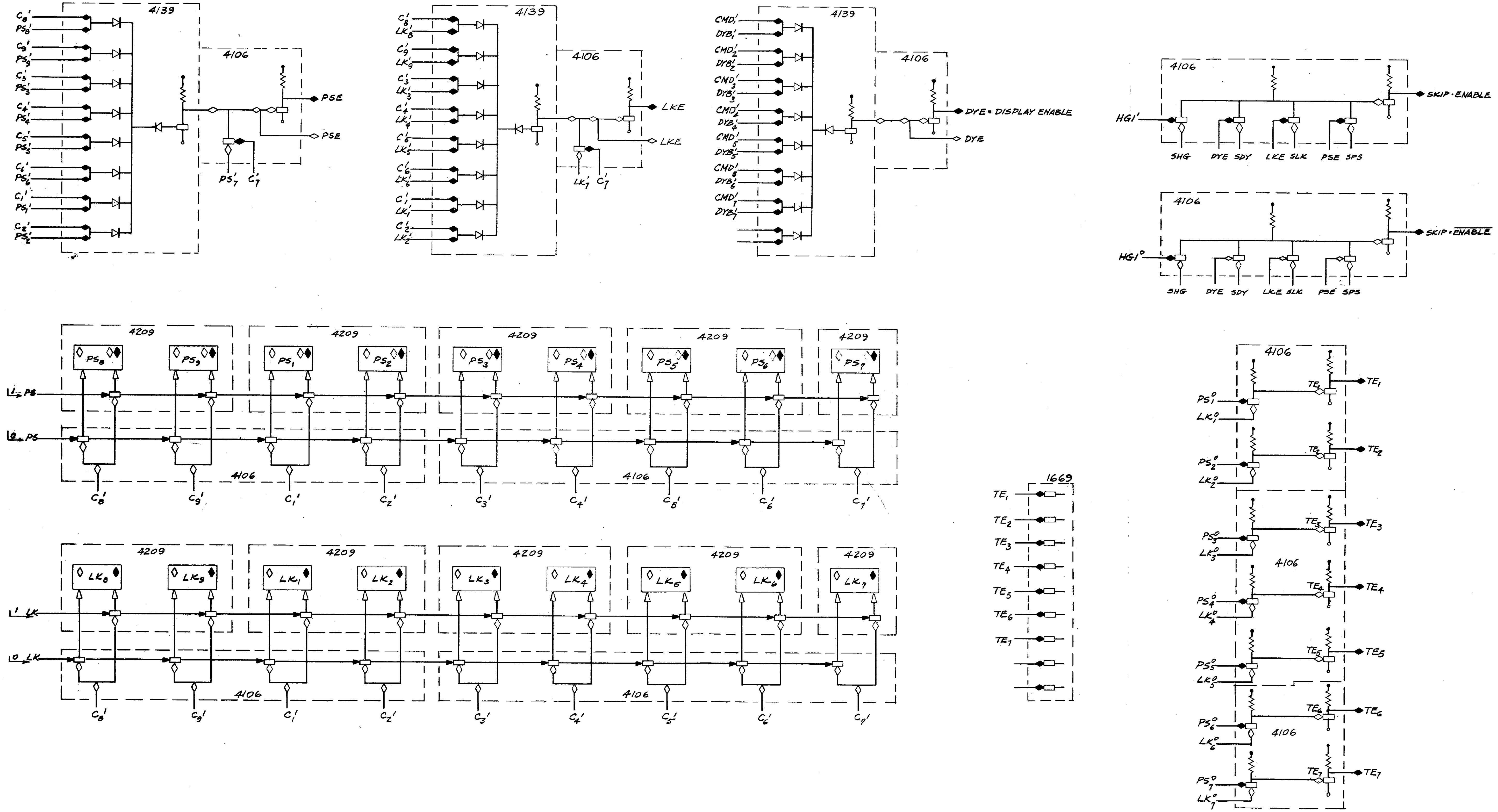


FIG. 5.5 - PRINT STATUS, LOCK, SKIP AND TYPE ENABLE

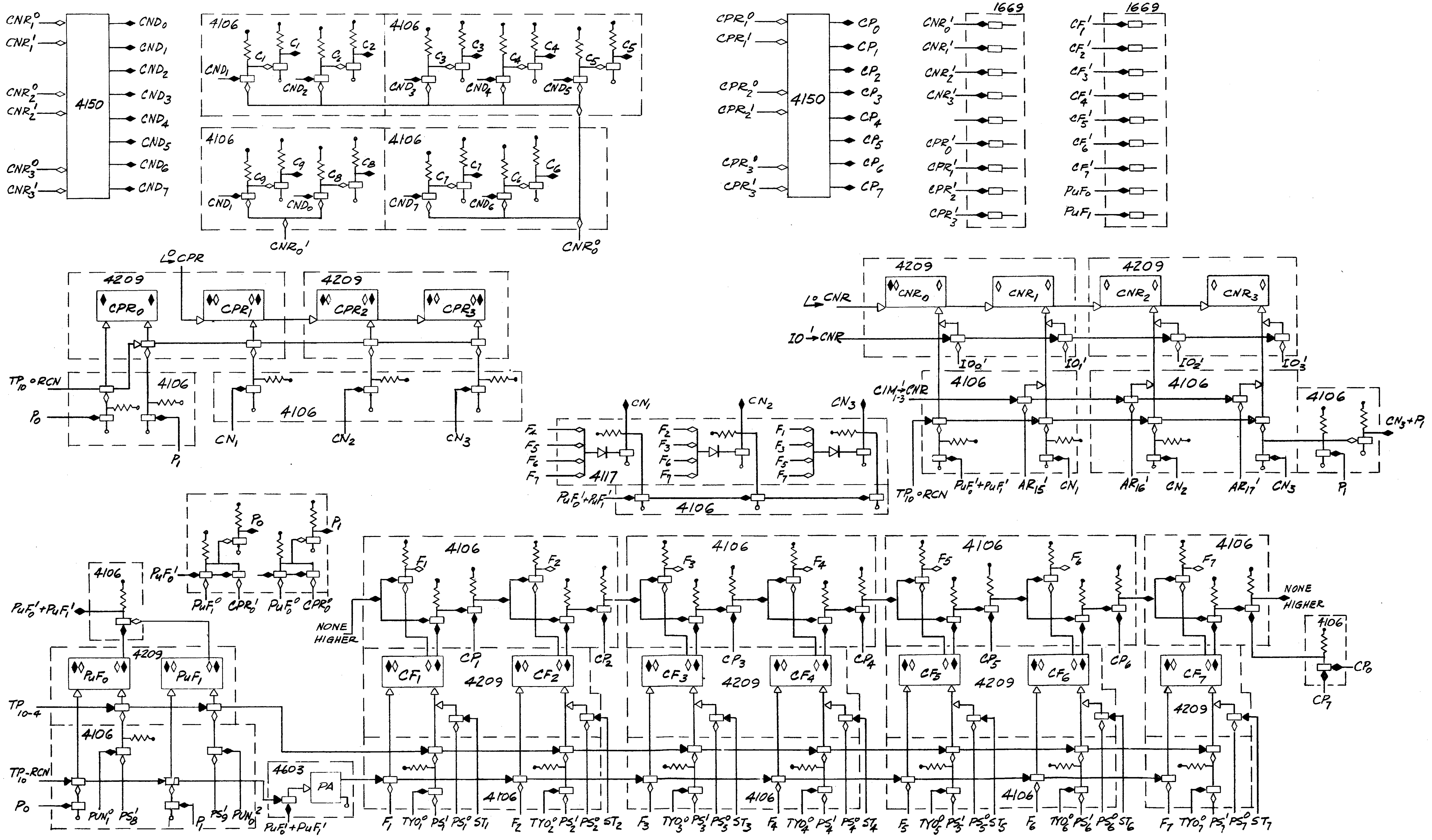

4150
CNR1⁰ CNR1' CNR2⁰ CNR2' CNR3⁰ CNR3'
CND0 CND1 CND2 CND3 CND4 CND5 CND6 CND7
4106
C1 C2 C3 C4 C5 C6 C7 C8 C9
CNR0' CNR0⁰
4150
CPR1⁰ CPR1' CPR2⁰ CPR2' CPR3⁰ CPR3'
CP0 CP1 CP2 CP3 CP4 CP5 CP6 CP7
1669
CNR0' CNR1' CNR2' CNR3' CPR0' CPR1' CPR2' CPR3'
CF1' CF2' CF3' CF4' CF5' CF6' CF7' PuF0 PuF1
L⁰CPR
4209
CPR0 CPR1 CPR2 CPR3
TP10 ∘RCN
P0 P1
CN1 CN2 CN3
L⁰CNR
CNR0 CNR1 CNR2 CNR3
IO'→CNR
IO0' IO1' IO2' IO3'
CIM'→CNR 1-3
4117
F1 F2 F3 F4 F5 F6 F7
PuF0'+PuF1'
AR15' AR16' AR17'
CN3+P1
PuF0' PuF1⁰ CPR0' PuF0⁰ CPR0⁰
NONE HIGHER
PuF0 PuF1
CF1 CF2 CF3 CF4 CF5 CF6 CF7
TP10-4
TP10-RCN
PUN1⁰ PS8' PS9' PUN0²
4603
PA
TYO⁰ PS' PS⁰ ST1 ST2 ST3 ST4 ST5 ST6 ST7

FIG. 5.6 - CONSOLE FLAGS, CNR AND CPR

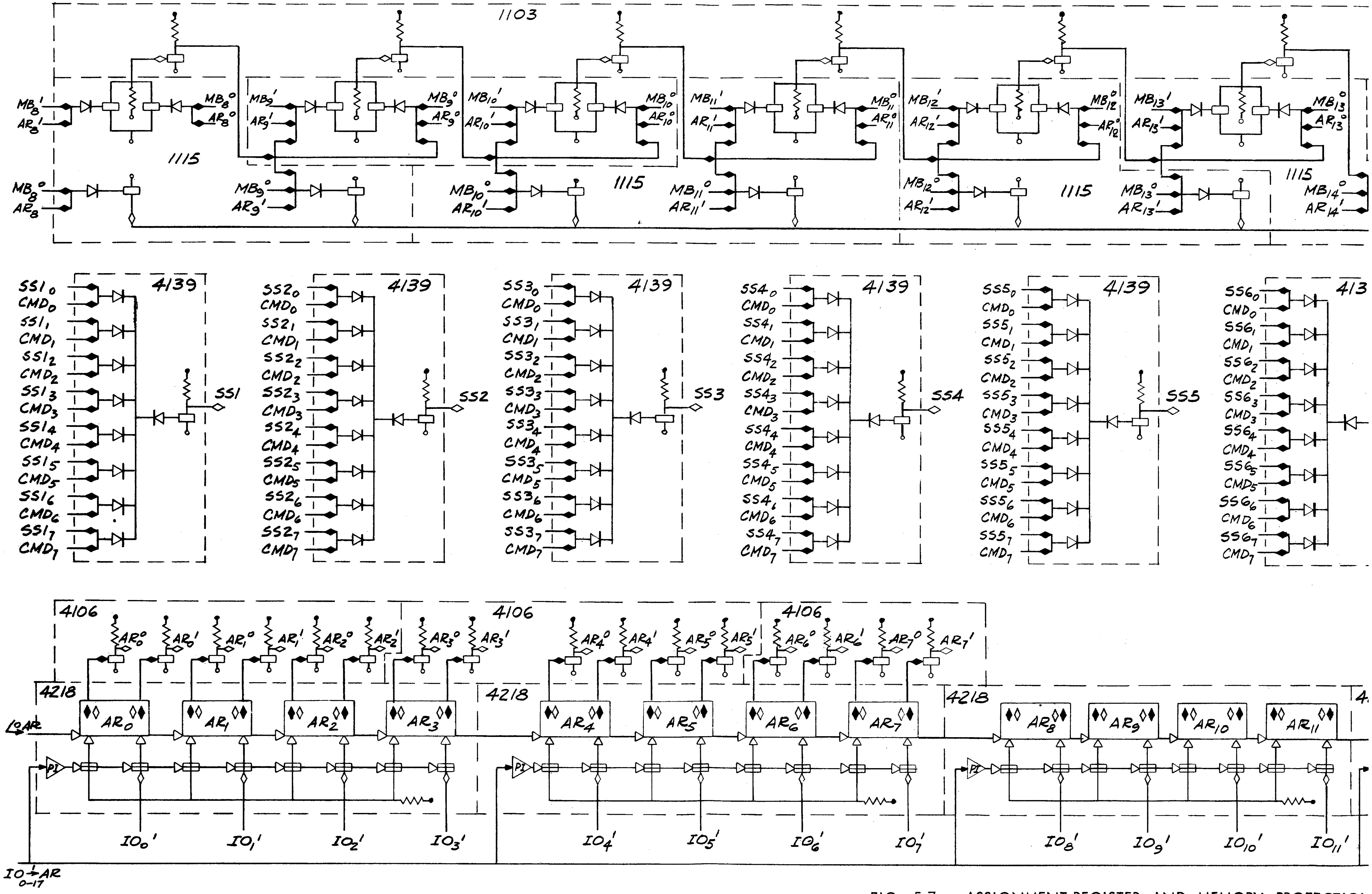

1103
1115
MB8'
AR8'
MB8°
AR8°
MB9'
AR9'
MB9°
AR9°
MB10'
AR10'
MB10°
AR10°
MB11'
AR11'
MB11°
AR11°
MB12'
AR12'
MB12°
AR12°
MB13'
AR13'
MB13°
MB14°
AR14'
4139
SS1
SS2
SS3
SS4
SS5
SS10 CMD0 SS11 CMD1 SS12 CMD2 SS13 CMD3 SS14 CMD4 SS15 CMD5 SS16 CMD6 SS17 CMD7
SS20 CMD0 SS21 CMD1 SS22 CMD2 SS23 CMD3 SS24 CMD4 SS25 CMD5 SS26 CMD6 SS27 CMD7
SS30 CMD0 SS31 CMD1 SS32 CMD2 SS33 CMD3 SS34 CMD4 SS35 CMD5 SS36 CMD6 SS37 CMD7
SS40 CMD0 SS41 CMD1 SS42 CMD2 SS43 CMD3 SS44 CMD4 SS45 CMD5 SS46 CMD6 SS47 CMD7
SS50 CMD0 SS51 CMD1 SS52 CMD2 SS53 CMD3 SS54 CMD4 SS55 CMD5 SS56 CMD6 SS57 CMD7
SS60 CMD0 SS61 CMD1 SS62 CMD2 SS63 CMD3 SS64 CMD4 SS65 CMD5 SS66 CMD6 SS67 CMD7
4106
AR0° AR0' AR1° AR1' AR2° AR2' AR3° AR3' AR4° AR4' AR5° AR5' AR6° AR6' AR7° AR7'
4218
0→AR
AR0 AR1 AR2 AR3 AR4 AR5 AR6 AR7 AR8 AR9 AR10 AR11
PI
IO0' IO1' IO2' IO3' IO4' IO5' IO6' IO7' IO8' IO9' IO10' IO11'
IO→AR 0-17

FIG. 5.7 - ASSIGNMENT REGISTER AND MEMORY PROTECTION

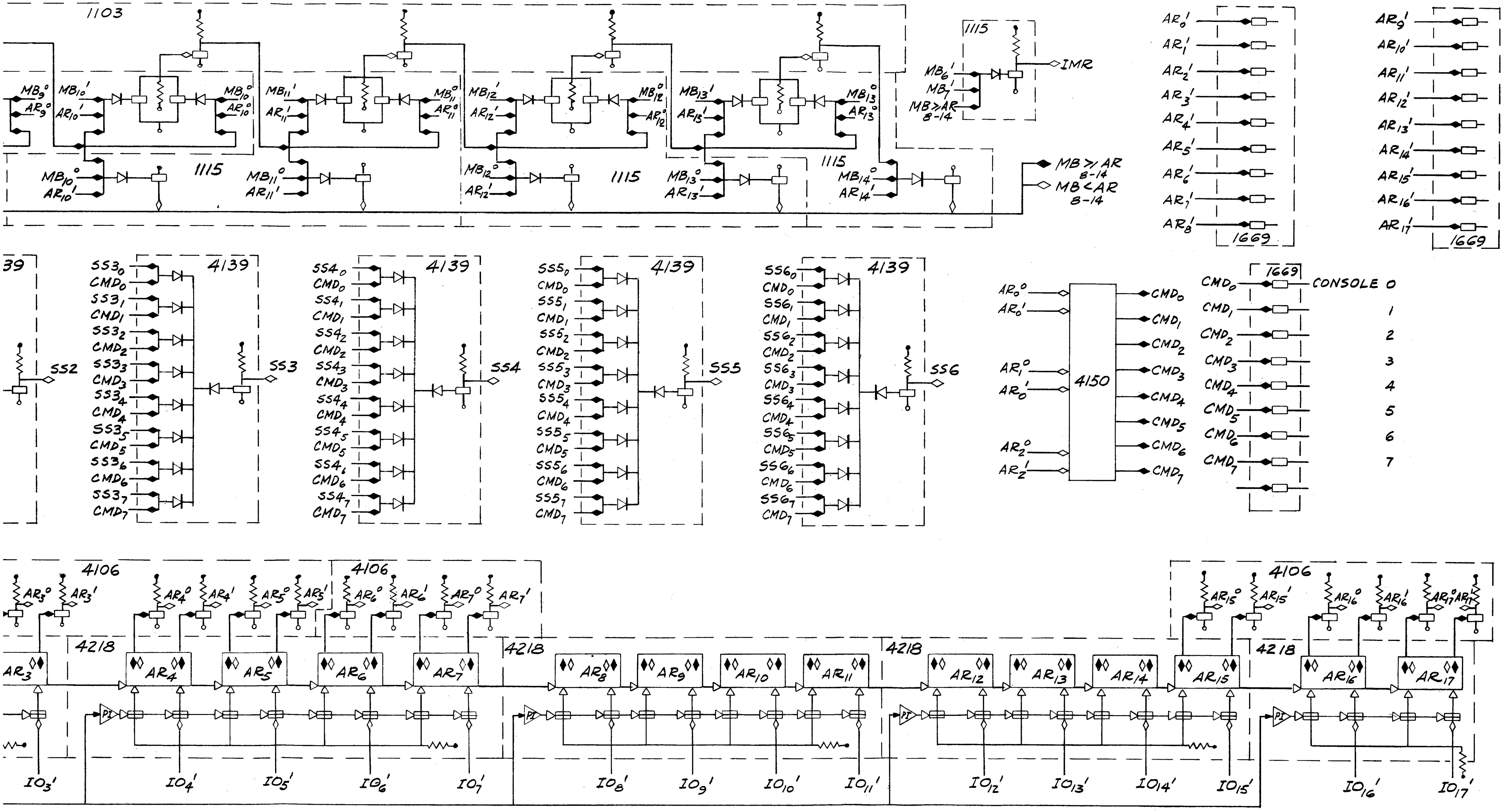


FIG. 5.7 - ASSIGNMENT REGISTER AND MEMORY PROTECTION

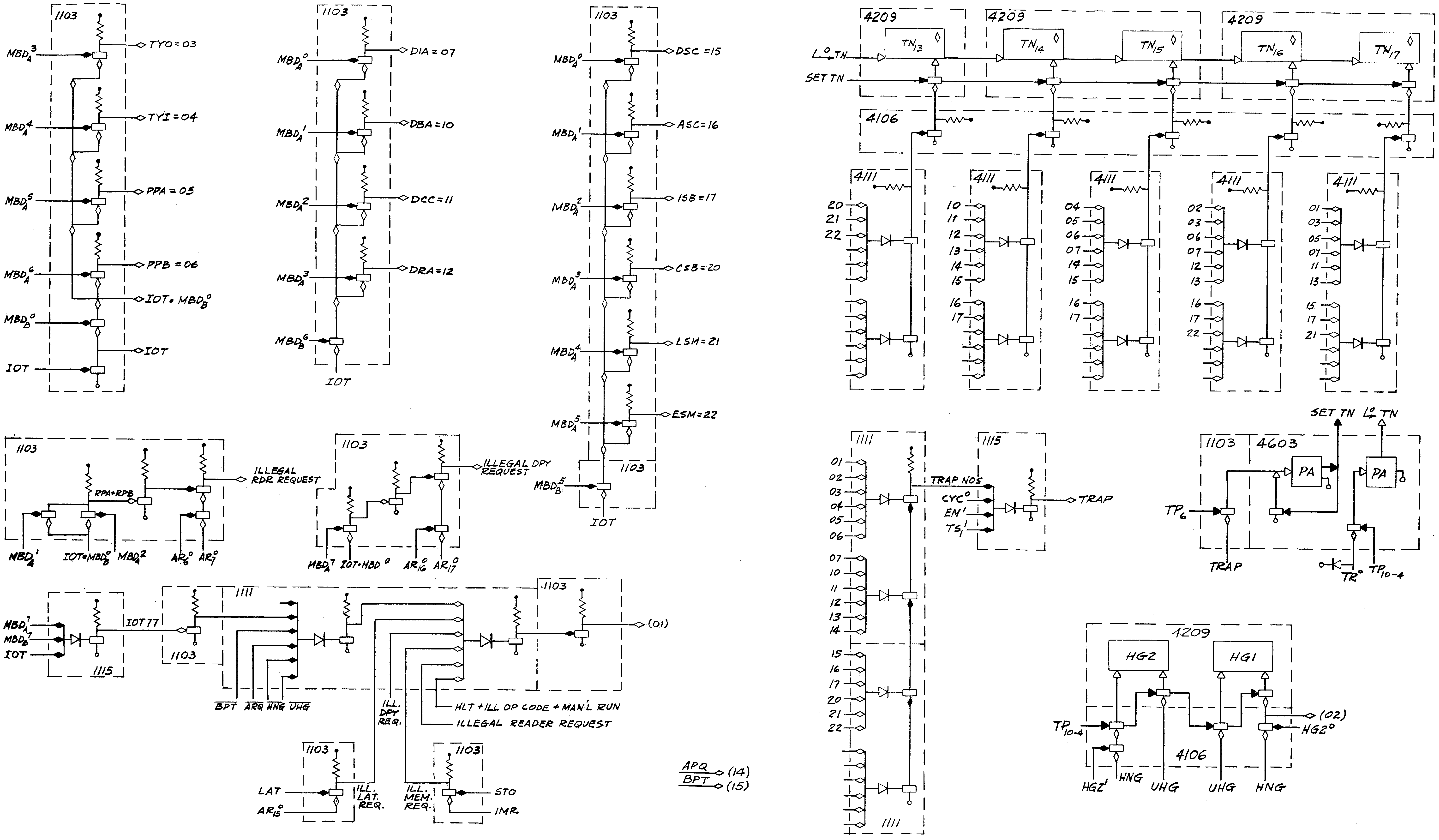


FIG. 5.8 - TRAP LOGIC

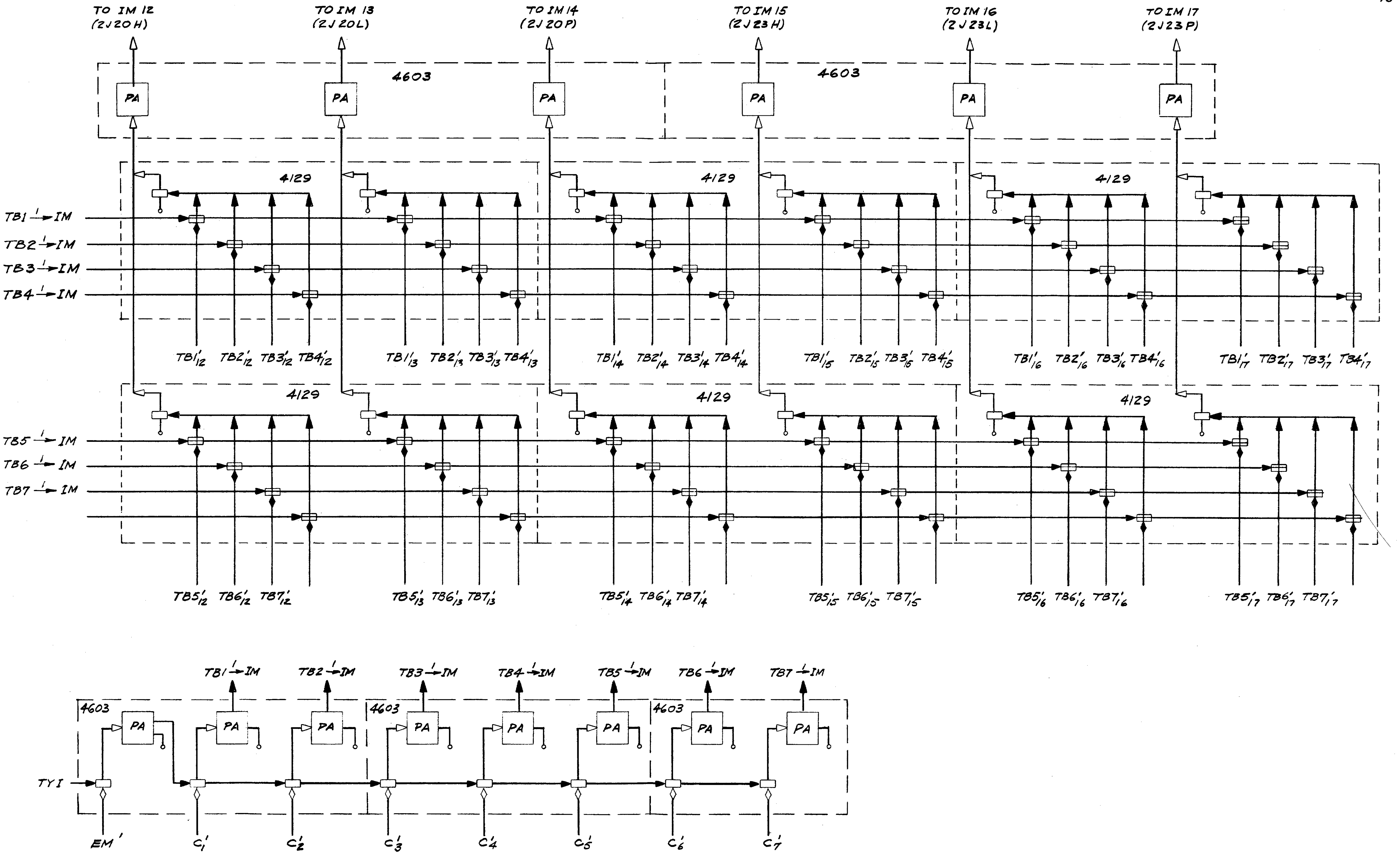


FIG. 5.9 - TYPE MIXER AND CONTROL

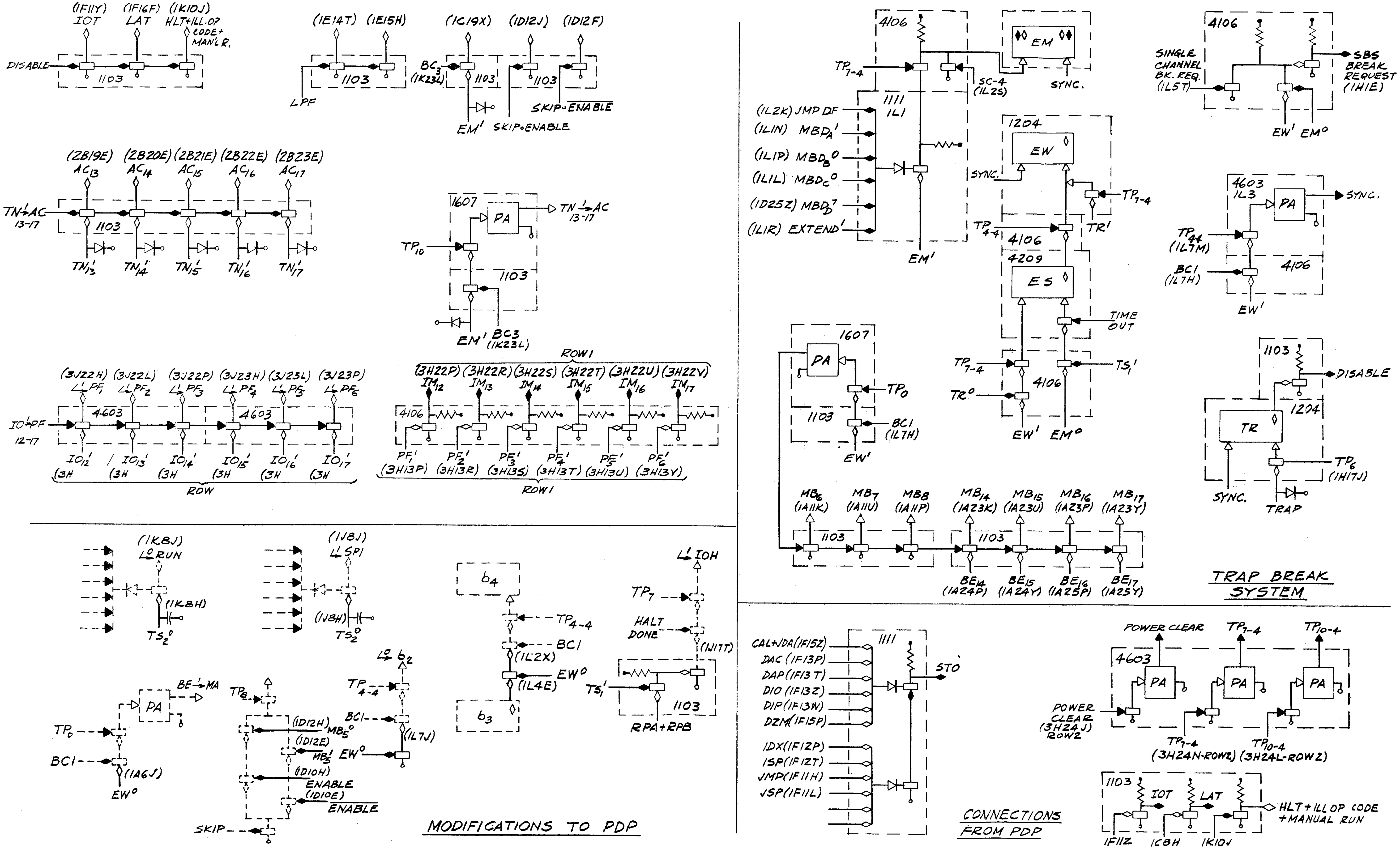


FIG. 5.10 - MODIFICATIONS TO PDP-1 AND TRAP BREAK SYSTEM

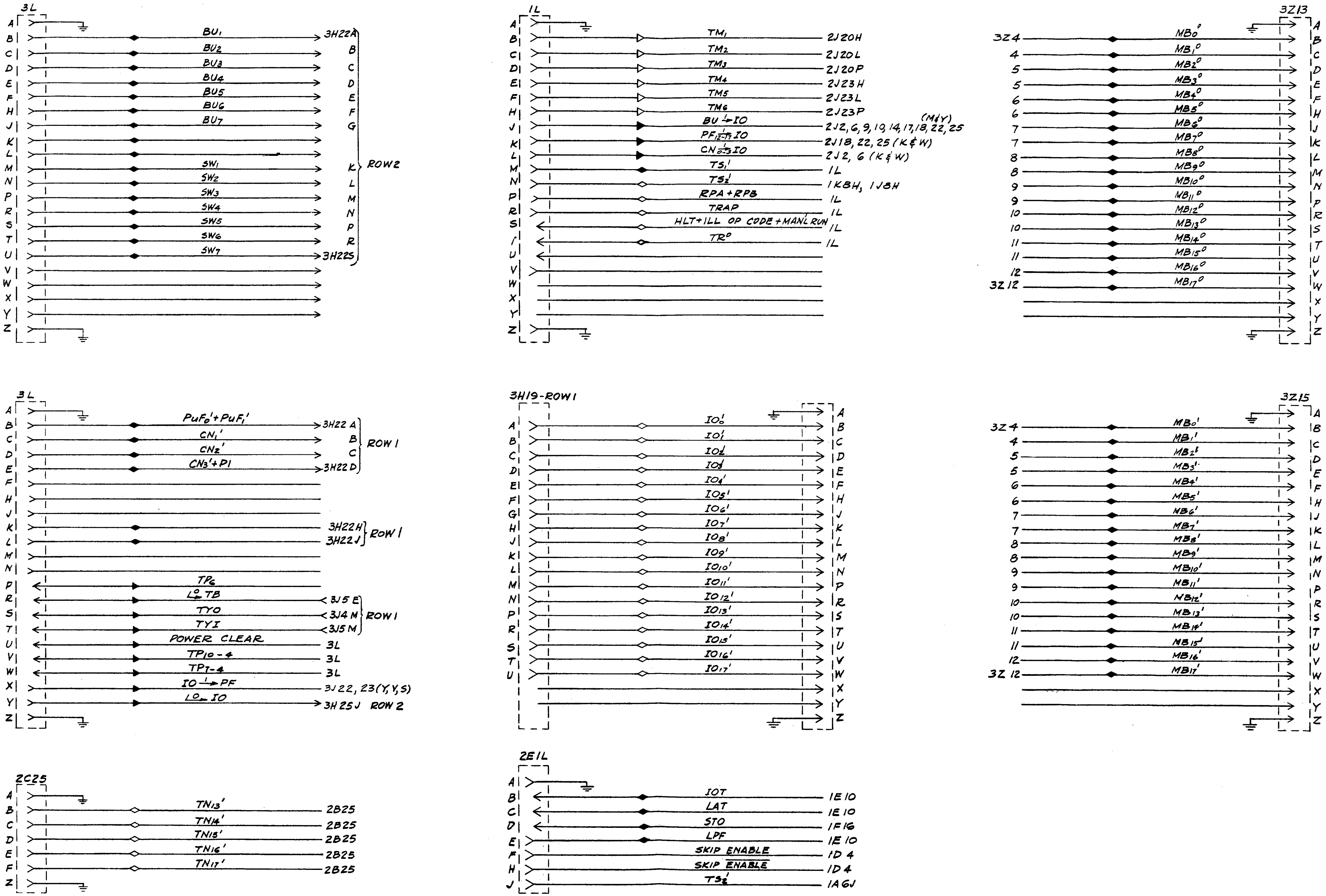
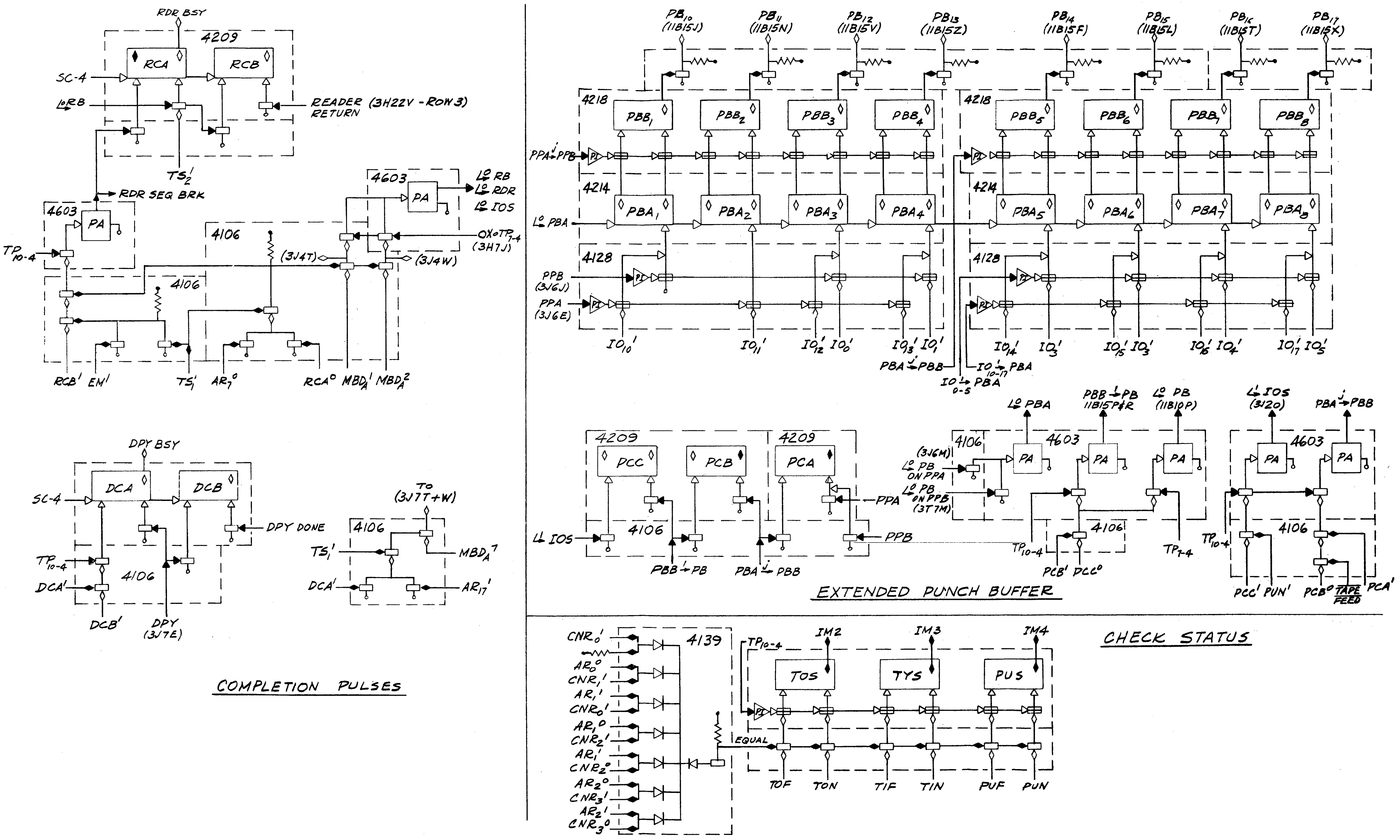

3L
A B C D E F H J K L M N P R S T U V W X Y Z
BU1 BU2 BU3 BU4 BU5 BU6 BU7
SW1 SW2 SW3 SW4 SW5 SW6 SW7
3H22A B C D E F G K L M N P R 3H22S
ROW2
1L
TM1 — 2J20H
TM2 — 2J20L
TM3 — 2J20P
TM4 — 2J23H
TM5 — 2J23L
TM6 — 2J23P
BU 1→IO — 2J2, 6, 9, 10, 14, 17, 18, 22, 25 (M&Y)
PF12-17 1→IO — 2J18, 22, 25 (K&W)
CN0-3 1→IO — 2J2, 6 (K&W)
TS1' — 1L
TS2' — 1K8H, 1J8H
RPA+RPB — 1L
TRAP — 1L
HLT+ILL OP CODE+MAN'L RUN — 1L
TR0 — 1L
3Z13
3Z4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 3Z12
MB0⁰ MB1⁰ MB2⁰ MB3⁰ MB4⁰ MB5⁰ MB6⁰ MB7⁰ MB8⁰ MB9⁰ MB10⁰ MB11⁰ MB12⁰ MB13⁰ MB14⁰ MB15⁰ MB16⁰ MB17⁰
3L
PuF0'+PuF1' — 3H22 A
CN1' — B
CN2' — C
CN3'+PI — 3H22 D
ROW 1
3H22H
3H22J
ROW 1
TP6
L0 TB — 3J5 E
TYO — 3J4 M
TYI — 3J5 M
ROW 1
POWER CLEAR — 3L
TP10-4 — 3L
TP7-4 — 3L
IO 1→PF — 3J22, 23(Y, V, S)
L0→IO — 3H25J ROW 2
3H19-ROW1
IO0' IO1' IO2' IO3' IO4' IO5' IO6' IO7' IO8' IO9' IO10' IO11' IO12' IO13' IO14' IO15' IO16' IO17'
3Z15
MB0' MB1' MB2' MB3' MB4' MB5' MB6' MB7' MB8' MB9' MB10' MB11' MB12' MB13' MB14' MB15' MB16' MB17'
2C25
TN13' — 2B25
TN14' — 2B25
TN15' — 2B25
TN16' — 2B25
TN17' — 2B25
2E1L
IOT — 1E10
LAT — 1E10
STO — 1F16
LPF — 1E10
SKIP ENABLE — 1D4
$\overline{\text{SKIP ENABLE}}$ — 1D4
TS2' — 1A6J

FIG. 5.11 - INTERCONNECTING WIRING



COMPLETION PULSES

EXTENDED PUNCH BUFFER

CHECK STATUS

FIG. 5.12 - MODIFICATIONS TO PDP-1, INPUT - OUTPUT

List A

Code	Organization	No. of copies
AF 5	AFMTC (AFMTC Tech Library - MU-135 Patrick AFB, Fla.	1
AF 18	AUL Maxwell AFB, Ala.	1
AF 43	ASD (ASAPRD-Dist) Wright-Patterson AFB, Ohio	1
AF 124	RADC (RAYLD) Griffiss AFB, New York Attn: Documents Library	1
AF 139	AF Missile Development Center (MDGRT) Holloman AFB, New Mexico	1
AF 318	ARL (Technical Library) Building 450 Wright-Patterson AFB, Ohio	1
Ar 5	Commanding General USASRDL Ft. Monmouth, N.J. Attn: Tech. Doc. Ctr. SIGRA/SL-ADT	1
Ar 9	Department of the Army Office of the Chief Signal Officier Washington 25, D.C. SIGRD-4a-2	1
Ar 50	Commanding Officer Attn: ORDTL-012 Diamond Ordance Fuze Laboratories Washington 25, D.C.	1
Ar 67	Commanding General U.S. Army Ordnance Missile Command Redstone Arsenal, Alabama Attn: Technical Library	1

Code	Organization	No. of copies
G 2	ASTIA (TIPAA) Arlington Hall Station Arlington 12, Virginia	10
G 68	National Aeronautics and Space Agency 1520 H. Street, N.W. Washington 25, D.C. Attn: Library	1
G 109	Director Langley Research Center National Aeronautics and Space Administration Langley Field, Virginia	1
M 6	AFCRL, OAR (CRIPA - Stop 39) L.G. Hanscom Field Bedford, Massachusetts	10
N 9	Chief, Bureau of Naval Weapons Department of the Navy Washington 25, D.C. Attn: DLI-31	2
N 29	Director (Code 2027) U.S. Naval Research Laboratory Washington 25, D.C.	2
I 292	Director, USAF Project RAND The Rand Corporation 1700 Main Street, Santa Monica, Cal. Thru: A.F. Liaison Office	1
AF 253	Technical Information Office European Office, Aerospace Research Shell Building, 47 Cantersteen Brussels, Belgium	1
Ar 107	U.S. Army Aviation Human Research Unit U.S. Continental Army Command P.O. Box 428, Fort Rucker, Alabama Attn: Maj. Arne H. Eliasson	1

Code	Organization	No. of copies
G 8	Library Boulder Laboratories National Bureau of Standards Boulder, Colorado	2
M 63	Institute of the Aerospace Sciences, Inc. 2 East 64th Street New York 21, New York Attn: Librarian	1
N 73	Office of Naval Research Branch Office, London Navy 100, Box 39 F.P.O. New York, N.Y.	10
U 32	Massachusetts Institute of Technology Research Laboratory of Electronics Building 26, Room 327 Cambridge 39, Massachusetts Attn: John H. Hewitt	1
U 431	Alderman Library University of Virginia Charlottesville, Virginia	1

List A-H

Code	Organization	No. of copies
AF 179	AFOSR (Harold Wooster) Director, Research Information Office Washington 25, D.C.	1
AF 237	WADD (WCLJR) Attn: Mr. R.M. Lemmen Wright-Patterson AFB, Ohio	1
AF 238	RADC (RCWID) Attn: Capt. B.J. Long Griffiss AFB, N.Y.	1
AF 239	Atomic Energy Commission (John R. Pasta) Division of Research Washington 25, D.C.	1
AF 255	496L SPO (Space Track) (ESSIS, Stop 2) 424 Trapelo Road Waltham 54, Mass.	1
G 72	Roger Moulton National Security Agency Ft. George G. Meade, Maryland	1
G 73	National Science Foundation Engineering Sciences Program 1951 Constitution Avenue Washington 25, D.C.	1
M 80	ESD (ESRDD, Maj. W. Harris) L.G. Hanscom Field Bedford, Mass.	1
N 140	Office of Naval Research Information Systems Branch Attn: Code 437, Marshall C. Yovits Washington 25, D.C.	1
	Remaining copies to: AFCRL, OAR (CRRB, Dr. H. Zschirnt) L.G. Hanscom Field, Bedford, Mass.	12

UNCLASSIFIED

AD
Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts
A TIME SHARING SYSTEM FOR THE PDP-1
COMPUTER by John E. Yates. June 1962. 102p. incl.
illus. (Contract AF-19(604)-6654, M. I. T. Project
DSR 8494, Scientific Report No. 5 AFCRL-62-519
Report ESL-R-140
Unclassified report

A system for time sharing a PDP-1 digital computer with seven typewriters, two paper tape punches, two paper tape readers and two CRT displays is described. The additional hardware required for the system and the modifications required to a basic PDP-1 are described and a program is presented to handle the monitor or "executive" functions of the system. A system using two typewriters, one punch, one reader and one display based on this design is currently being installed at M. I. T.

UNCLASSIFIED

UNCLASSIFIED

AD
Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts
A TIME SHARING SYSTEM FOR THE PDP-1
COMPUTER by John E. Yates. June 1962. 102p. incl.
illus. (Contract AF-19(604)-6654, M. I. T. Project
DSR 8494, Scientific Report No. 5 AFCRL-62-519
Report ESL-R-140
Unclassified report

A system for time sharing a PDP-1 digital computer with seven typewriters, two paper tape punches, two paper tape readers and two CRT displays is described. The additional hardware required for the system and the modifications required to a basic PDP-1 are described and a program is presented to handle the monitor or "executive" functions of the system. A system using two typewriters, one punch, one reader and one display based on this design is currently being installed at M. I. T.

UNCLASSIFIED

UNCLASSIFIED

AD
Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts
A TIME SHARING SYSTEM FOR THE PDP-1
COMPUTER by John E. Yates. June 1962. 102p. incl.
illus. (Contract AF-19(604)-6654, M. I. T. Project
DSR 8494, Scientific Report No. 5 AFCRL-62-519
Report ESL-R-140
Unclassified report

A system for time sharing a PDP-1 digital computer with seven typewriters, two paper tape punches, two paper tape readers and two CRT displays is described. The additional hardware required for the system and the modifications required to a basic PDP-1 are described and a program is presented to handle the monitor or "executive" functions of the system. A system using two typewriters, one punch, one reader and one display based on this design is currently being installed at M. I. T.

UNCLASSIFIED

UNCLASSIFIED

AD
Electronic Systems Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts
A TIME SHARING SYSTEM FOR THE PDP-1
COMPUTER by John E. Yates. June 1962. 102p. incl.
illus. (Contract AF-19(604)-6654, M. I. T. Project
DSR 8494, Scientific Report No. 5 AFCRL-62-519
Report ESL-R-140
Unclassified report

A system for time sharing a PDP-1 digital computer with seven typewriters, two paper tape punches, two paper tape readers and two CRT displays is described. The additional hardware required for the system and the modifications required to a basic PDP-1 are described and a program is presented to handle the monitor or "executive" functions of the system. A system using two typewriters, one punch, one reader and one display based on this design is currently being installed at M. I. T.

UNCLASSIFIED

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SCIENCE AND ARTS

1861

MENS ET MANUS