PROFILE OF THE PURDUE MACE OPERATING SYSTEM

V. Y. SHEN

(10/70) LO PMACE

TABLE OF CONTENTS

I. Introduction

This report describes the basic functions performed by the Purdue MACE operating system. Topics covered are the system status after dead start; the loading, scheduling, execution, and termination of simple jobs. The approach of tracing a test job through the system is used. The reader is expected to be familiar with the hardware configuration and characteristics of the CDC 6500 system as described in the CDC 6400/6500/6600 Computer Systems Reference Manual (Pub. No. 6010000). Selected system program listings may prove to be helpful at times but are not necessary.

II. The Status of the System After Dead Start

"Dead start" is the term used by CDC to denote the initial loading process of programs and tables belonging to the operating system* and the computer will be ready to process users' jobs when dead start is completed. A small program set by toggle switches on the system dead start panel is read into PP0, which starts execution immediately. It starts the reading process of the system tape from a tape unit, which contains the resident programs and tables of all processors and all the library programs. When the dead start process is completed, all processors are executing their individual resident programs or "idle loops". The low order area of the main core memory (about 30000_8 words) is initiated to contain tables, flags, and routines, collectively known as the central memory resident (CMR). System's library programs are loaded into CMR, the extended core storage (ECS), and the disk according to their frequencies of usage.

The CMR is divided into several regions as shown in Figure 1. The idle loop is one of the routines in the next to last region of CMR. We shall only discuss two regions at this time and bring up the others in the later sections when it becomes necessary.

---

*There are other phases of dead start which will be described in a later section.

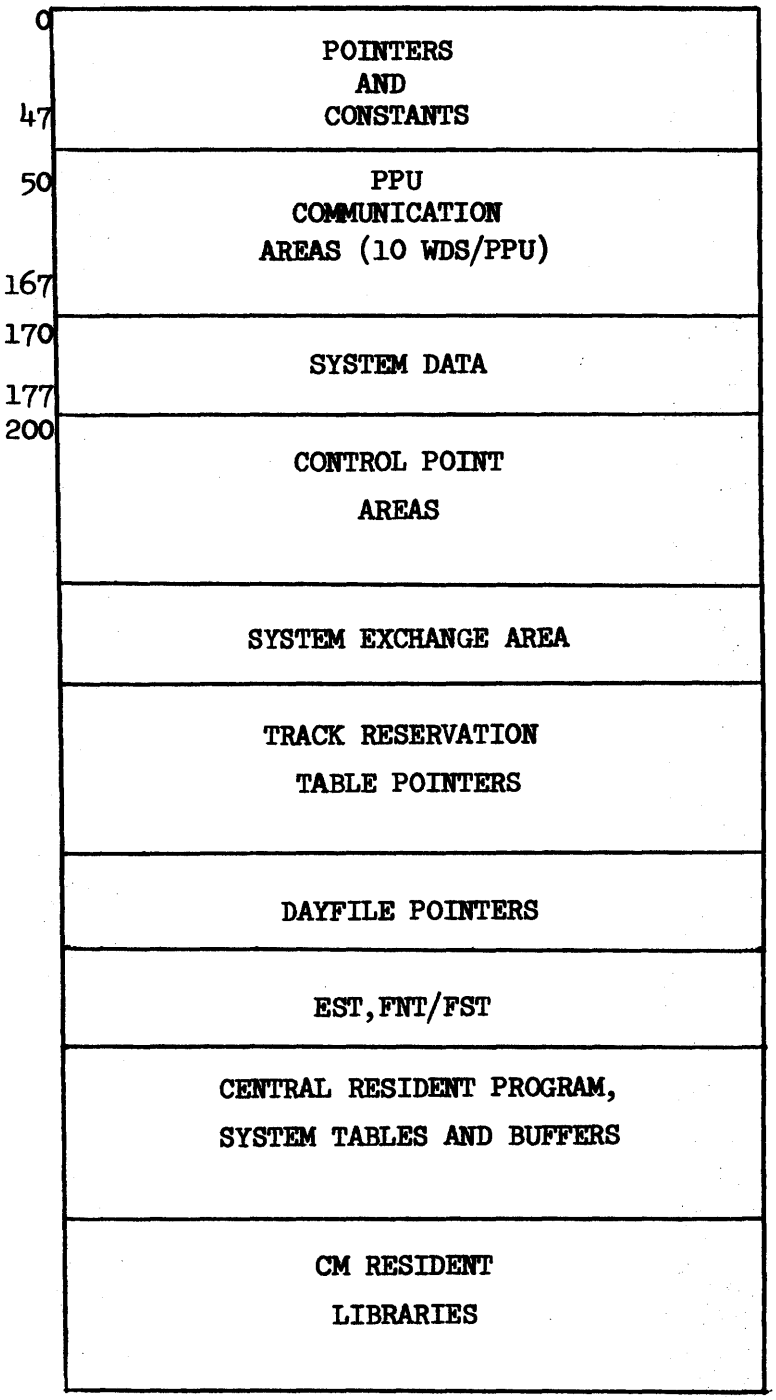


Figure 1. The central memory resident area

The region designated as PPU communication areas contains 10 blocks of 8 words (10_8 words) each. There is a block corresponding to each PPU. Except for PP0, the communication area of each PPU has the format shown in Figure 2. A PPU uses its communication area to exchange information with the system monitor. PP0 is dedicated to the monitor program MTR which uses the communication area differently from the other PPU's. This will not be discussed at this time.

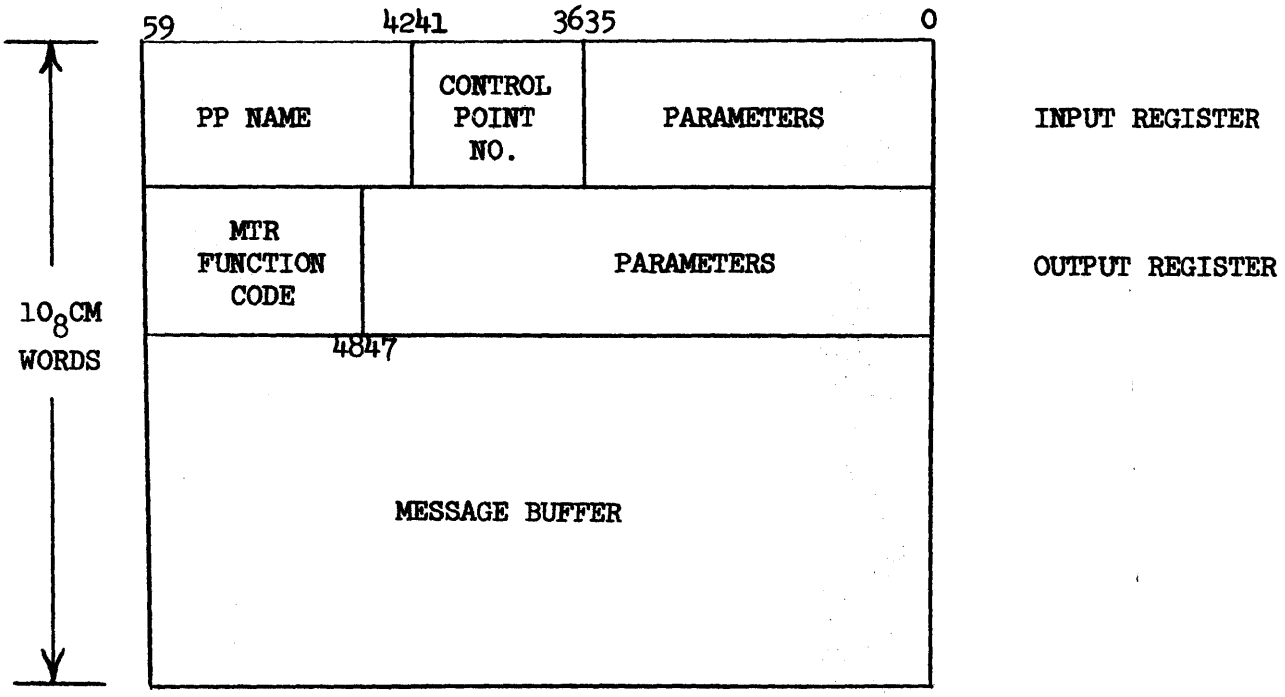


Figure 2. A PPU communication area

An idling PPU is constantly (every 128 microseconds) checking the input register of its communication area. When the PP monitor MTR writes a three character name in the leftmost 18 bits of the word, the PPU starts the loading and execution processes of the named PP program, using the parameter information in the low order bits of the word if appropriate. During the execution of the PP resident or a PP program some help from the system monitor may be needed (for example, an equipment request). The request can be passed through the output register of the communication area. A code for the requested MTR function is written in the leftmost byte (12 bits) of the word, and any parameters for the function will be written in the remainder of the word and also the message buffer if necessary.

The system monitor is divided into two parts. One is a PP program MTR which resides in PP0 permanently. It is in overall control of the computer system. Its functions include the allocation of the resources of the computer (i. e. central memory, channels, equipments, CPU's, and the other PPU's). The second part is a set of routines in CMR. It performs certain executive functions which can best be done by a CPU. Examples that use these functions will be presented in a later section.

Part of MTR's loop consists of checking the output registers in the PPU communication areas. If a request is recognized, MTR branches to the part that performs the requested function. Sometimes other PPU's may be needed to help in completing the function. When

MTR finishes, it clears the output register of the requesting PPU and continues on its loop. When the PPU finishes executing the current PP program, it puts a completion code in the output register. This, in fact, is another MTR function request. Upon recognizing this request, MTR checks the recall status (to be described later) of the central program which is using this PPU and clears both the input and output registers. The PPU then returns to its idling loop.

PP1 through PP8 are called pool PPU's. They act independently and are allocated to the user's central program by MTR. Their major function is to act as an interface between the central program and the peripheral equipments. PP9 is dedicated to the dynamic system display program (DSD). Its function is to provide communication between the system and the operator. It reserves Channel 10 to maintain the information displayed on the console screens and to monitor the inputs from the console keyboard. DSD will interpret and take appropriate action for the commands typed in by the operator.

The region designated as control point areas in CMR contains a number of job control blocks. An active job in the system must be attached to a control point, which has a block of storage reserved as a control point area to contain all the pertinent information about the job. The Purdue MACE system may have up to 26 control points, the exact number is set at the dead start time.

A control point area is a block of 128 (200_8) words. The first 16 words are referred to as the exchange package area (Figure 3).

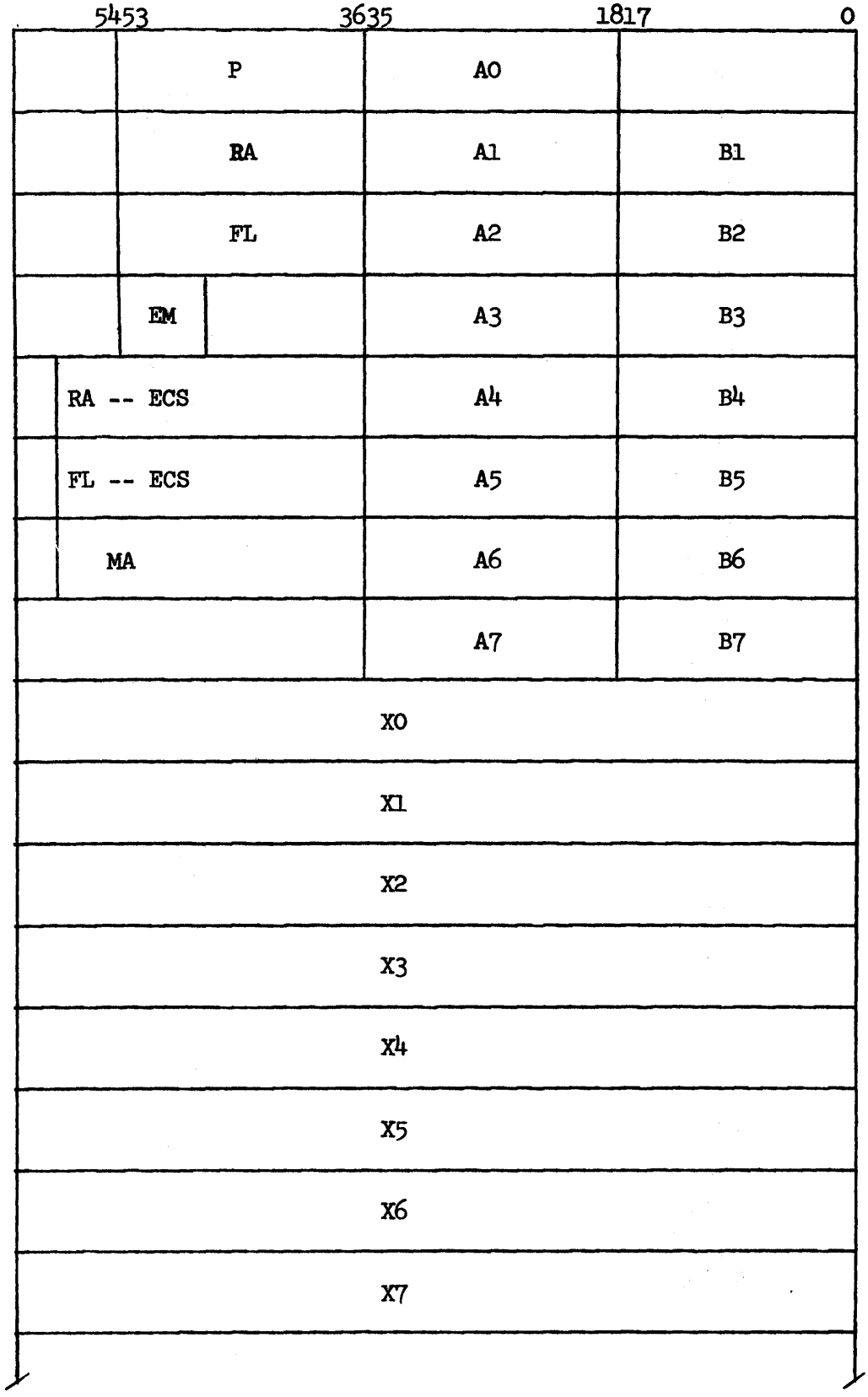


Figure 3. The exchange package area of a control point area

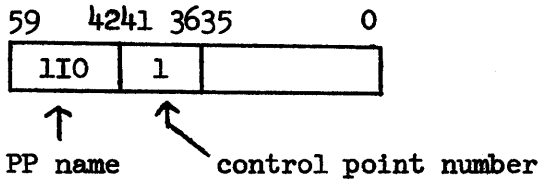
When a job is waiting to be processed by a CPU, the exchange package area contains the contents of various registers to be placed in the corresponding registers of the CPU before execution begins. When the job has a CPU assigned to it and is in execution, the exchange package area contains a system exchange package which is to be used to return the CPU to the idle loop if the job is interrupted for any reason. The remainder of the control point area contains information about the job such as status, priority, time limit, equipment assignments, etc. Their use will be discussed in later sections. A 72 word buffer is also set up for the job control statements at the end of the control point area.

III. Loading Jobs to the System

Assume we have a number of jobs on cards to be processed by the system after dead start. Normally the operator would type the AUTO command at the console keyboard to prepare the system for full multiprogramming mode. For instructional purposes, we shall simplify the procedure and only type in

1.BATCHIO.

As mentioned in the previous section, part of DSD's loop is to monitor the input from the console keyboard. It interprets and displays each character as it is typed in. DSD is ready to load a command overlay containing the input-output package BIO when the operator types in the last period. The loading process is started when the carriage return is typed by the operator, and DSD branches to BIO when the loading is completed. BIO posts a request to MTR for a pool PPU to execute a PP program called 1IO. It does this by first assembling a word in the following format:



It then writes this word into the first word of the message buffer in the PP9 communication area. It next writes the MTR function code to request a PPU in the output register. PP9 then enters a waiting loop which scans the output register constantly. If MTR does not respond within a fixed time period, a "SYSTEM BUSY" message will be displayed

and PP9 returns to its main loop.

At this time MTR should recognize that PP9 has an outstanding request almost immediately since it is not doing anything for anyone else. MTR then picks an available pool PPU, copies the first word of the message buffer of PP9 into the input register of this PPU, makes a record of the fact that this PPU is assigned to control point 1, keeps the information about the PPU time used by the job, clears the output register of PP9 and returns to its main loop. Upon recognizing that its request for a PPU has been granted, PP9 also returns to its main loop of refreshing the console displays and monitoring further keyboard inputs.

Let us assume that PP1 is chosen by the monitor and the request to perform 1IO for control point 1 is written in its input register. The resident program of PP1 initiates the loading process of 1IO, which seems to deserve a somewhat detailed description here.

All system PP programs reside in the system peripheral library. Physically a program may reside in the main core, ECS, or on a disk. Directories are therefore kept in CMR for easy retrieval of these programs. All PP programs have three-character names; there are certain conventions which are followed with respect to the first character when assigning names. Primary overlay (sometimes referred to as a PP transient) names either begin with an alphabetic character or the digit one (1). They are loaded starting from location 1100_8 of a PPU. Alphabetic overlays may be called from a running program, a control

card (in some cases), or from within the system, i. e. from a PPU. Secondary overlay names usually begin with the digit which is closest to the 1000_8 word block at which the overlay is to be loaded; for example, 2TS is loaded at 2040_8. No conventions have been established for the second and third characters, except theoretically they should have some mnemonic value.

Although the original design philosphy of the computer system calls for monitor functions to be performed by a PPU, in practice it is convenient to have the CPU itself perform some of the monitor functions. For example, a peripheral library search can be done much faster by a CPU since the library directory is kept in CMR. Other functions such as the data transfer between the central memory and ECS, and those requiring frequent references to CMR, should also be done by a CPU. Therefore a set of CPU monitor functions are defined and are handled differently from the MTR functions.

A CPU has two modes of operations which are indicated by a hardware flip-flop. It is said to be in "problem mode" when it is executing a user's program at a control point or certain system functions at a pseudo control point*. It is in "monitor mode" when it is executing the central monitor programs in CMR. A PPU may request central monitor service by first putting the corresponding function code into its output register. MTR will ignore this request since it knows this is not part of its functions. Instead of waiting for the output register to be cleared, the PPU issues a monitor exchange jump instruction (MXN) to a

---

*Control point n + 1, which is reserved for this purpose.

CPU. If the CPU is in problem mode, it is interrupted and set to monitor mode. The current contents of various registers are exchanged with the PPU exchange package, which is a predefined block in the system exchange area of CMR (see Figure 1). The CPU will proceed to decode the request in the output register and branch to the appropriate routine. When the CPU completes the function, it returns to the interrupted job and problem mode via a central exchange jump (XJ). The leftmost byte of the output register is cleared to indicate the completion. If the CPU is already in monitor mode, the MXN function will be ignored. In this case the PPU enters a waiting loop and issues MXN again later.

Without further digression, we shall resume the loading process of 1IO. PP1 first requests the monitor to search for the 1IO package in the PP library. This is a CPU monitor function and a predetermined CPU is switched to perform it. The output register contains the function code and the name 1IO as the parameter, right justified in the word.

The leftmost 24 bits of location 2 (PLDP) in CMR contains an address which points at the first word of the peripheral library directory (PLD). The PLD is a block of storage in the next to last region of CMR (Figure 1). Figure 4 shows the three types of entries in PLD, which cover all the system PP programs. The entries are ordered by the names of packages.

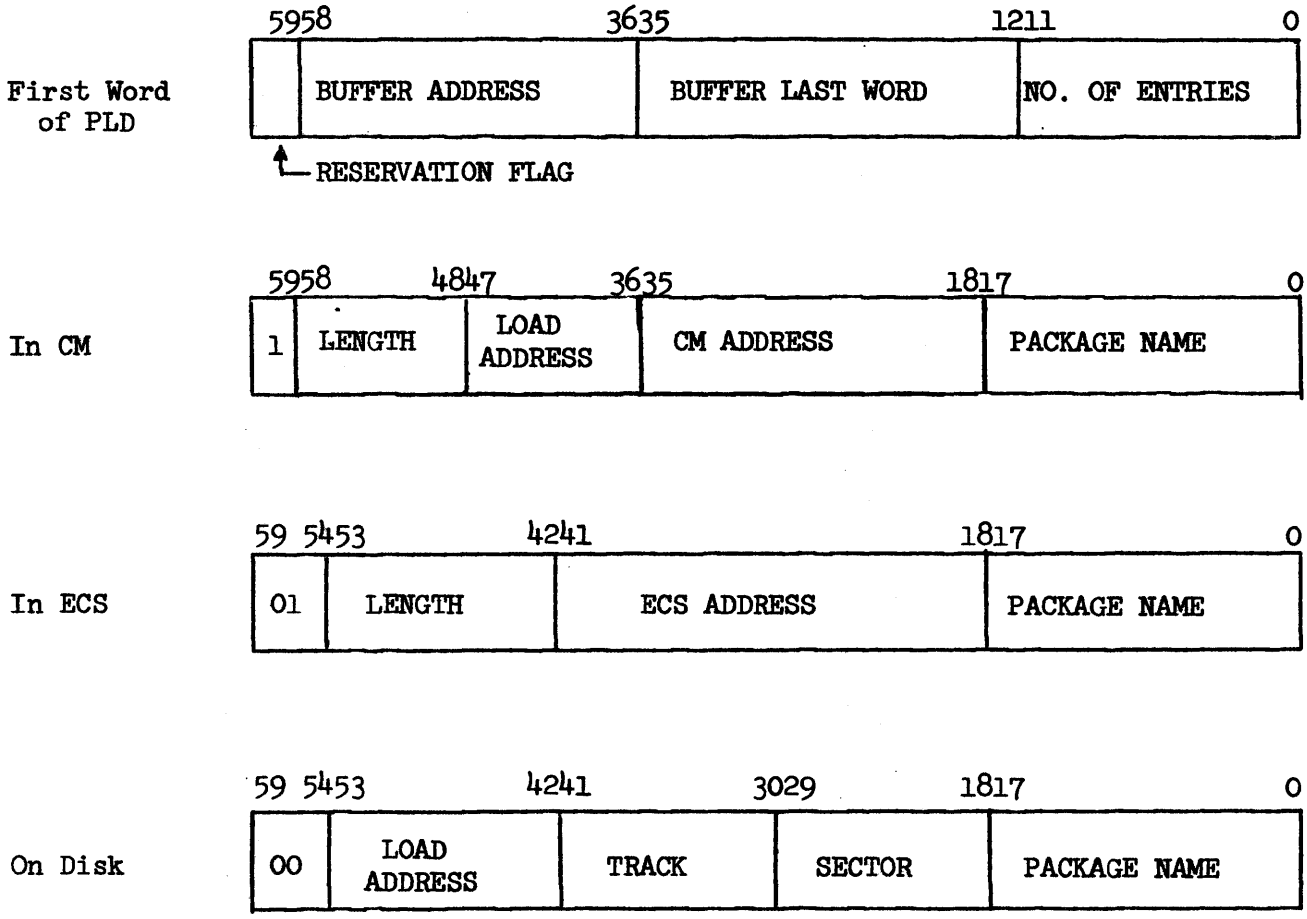


Figure 4. Entries of the peripheral library directory (PLD)

The CPU compares the name of the requested PP program package with the package names in PLD using a binary search scheme. It passes the result back to the PPU through the output register using the right most four bytes. If the package is stored in the central memory, the length, load address, and the location in the central memory are written in the output register. If the package is stored in ECS, it is read into a

segment of the buffer in the central menory (defined by the two addresses in the first word of PLD) and the information is passed to the PPU just as if the package is in the central memory. If the package is stored on a system disk, the load address and the location are passed to the PPU. The PLD can be reserved for modification purposes by setting the sign bit of its first word to 1. Any PPU that requests a search when PLD is reserved is asked to reissue the request at a later time.

If the requested PP program is already located in the central memory, the PPU loads it by simply reading it into the location specified by the load address* and branching to the entry point. Let us assume that 1IO resides on disk. Programs, alphanumeric data files and binary data files, regardless of content, are stored on the disks in a common structure. The 808 disk file used by the system has a physical structure shown in Figure 5. There are two independent units in each cabinet. A physical track on disk is divided into 100 sectors, each contains 322 12-bit bytes. A "track" as interpreted by the system programs is actually a half physical track consisting of either the odd or the even numbered sectors of a physical track. The first two bytes of a sector are called the control bytes and cannot be used to store data. They contain linking

---

*Certain PP programs can be loaded anywhere in a PPU and the load address is supplied by the PPU. They are called location-free programs.

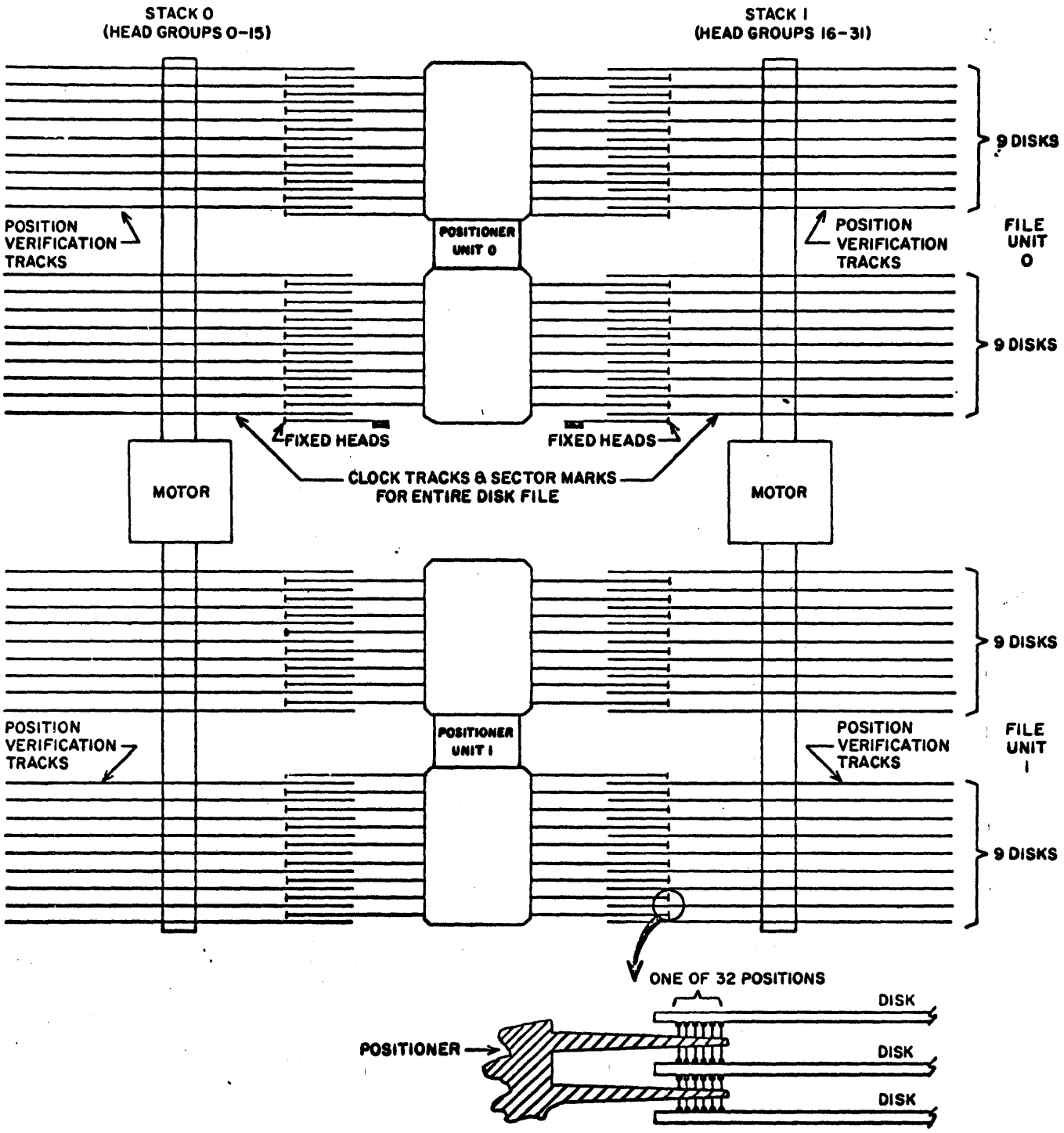


Figure 5. 808 Disk File Disks and Positioners

information of the stored data*. The remaining 320 bytes are the data bytes.

---

*Specifically, if the first control byte is not zero, it contains the sector number of the next block of data (the track number if Bit 11 is set); the second control byte contains the number of central memory words in the data bytes. If the sector is not filled, i. e., the second byte contains a number less than 100_8, it indicates the end of a logical record. If the first control byte is zero, it indicates the end of a logical file. In this case the second control byte contains the sector or track number of the next file; it is zero if the end of information is reached. A logical record or file may start at any sector of a track; but the first file of a set of files (information) must start at the first sector of a track.

A logical record of a file such as 1IO is stored on the system disk starting from the sector of a track as given in the peripheral library directory (Figure 4). In order to read the package 1IO from the system disk, PP1 first issues an MTR request for the system library. MTR maintains several tables in the central memory and PP0 to keep track of the status of all equipments and channels so as to avoid conflicts of use by different PPU's. The equipment status table (EST) resides in a region of CMR as shown in Figure 1. It has an entry for each equipment of the system ordered by equipment number. The format for a mass storage equipment entry (i. e. a disk or disk pack) is shown in Figure 6.

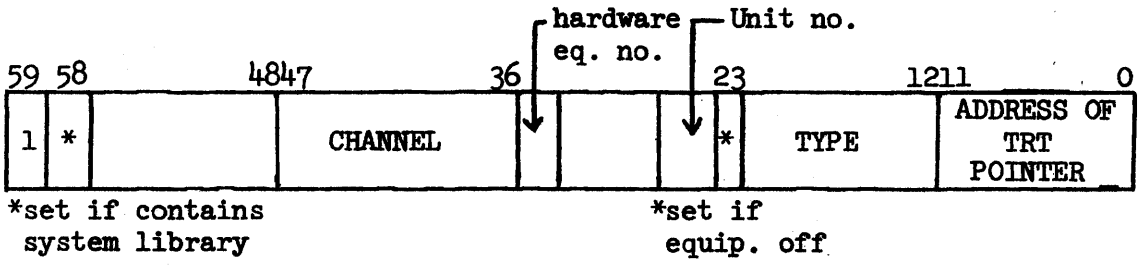


Figure 6. An entry in the equipment status table

MTR constructs a table from EST in PP0 during dead start time. It shows the devices containing the system library (normally only Disk 0) and the channel assigned to them. PP0 also has a channel status table with an entry for each channel. If a channel is used by a PPU the address of the PPU output register is written in the entry corresponding to the channel. There is also a copy of this table in the central memory located in the region referred to as system data in Figure 1.

Upon recognizing the request from PP1, MTR finds the equipment containing the system library (Disk 0) and reserves the channel assigned to it (Channel 2) for the PPU by writing "61" (which is the output register address) into the channel status table for that channel. It updates the table in CMR, returns the equipment and channel numbers to PP1's message buffer, and clears the output register. In case the channel is already in use by some other PPU, MTR will simply ignore the request by PP1 and return to its main loop. It will attempt to perform this function later when it is time to process PP1's request again. This is the same attitude MTR has toward all the unfulfilled MTR requests.

Starting from location 600_8, all pool PPU's have an overlay area for input/output drivers of mass storage equipments. The entry points for the drivers are the same to do reading, writing, and positioning of disk heads if necessary. The overlays reside in the central memory and can be loaded by the resident PPU loader very quickly. Before any I/O attempt, a PPU loads the appropriate driver if it is not already in the PPU.

After receiving the equipment and the channel, PP1 branches to the appropriate driver and reads in the package of 1IO sector by sector. The control bytes of each sector are checked each time to determine the location of the next sector. The reading process terminates after the reading of an incompletely filled sector. The channel is then released through another MTR request, which clears the entry in the channel status table. PP1 then branches to the entry point of 1IO.

The first thing 1IO does is to load and execute an overlay named 2IO. It uses the same loading procedure as the one used by the PPU resident program to load 1IO itself. 2IO, when entered, sets the job name "BATCHIO" in the control point area and issues a request to MTR for 100_8 words of central memory storage for control point 1. MTR keeps a table of unassigned storage blocks and decides what to do with each storage request. The central memory management algorithm is quite involved and will be discussed in a later section. We assume that the request for 100_8 words is granted without too much trouble since there is nothing else going on in the system. The block of storage assigned to a control point is called the "field length" of the control point. The reference address (RA) and the field length (FL) are written into the status word of the control point area, which is the twentieth (octal) word (Figure 7).

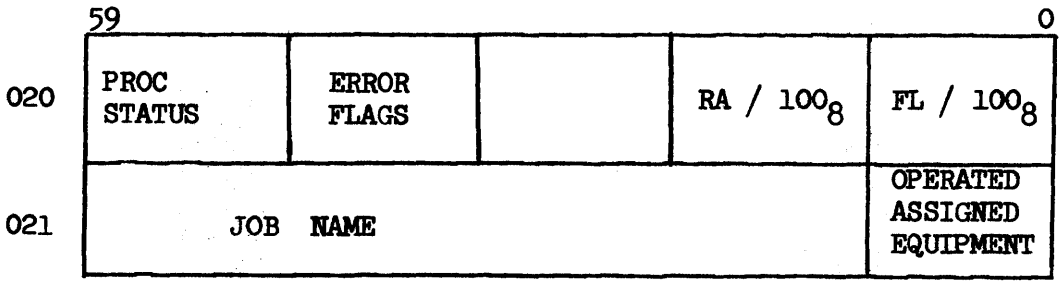


Figure 7. The status and job name words of a control point area

The job name will be displayed when DSD executes the overlay that displays the control point areas. Another function of 2IO is to construct a table of available equipments from the equipment status table. In addition to the mass storage equipment entry shown in Figure 6, there are two more entry formats (Figure 8) in the EST.

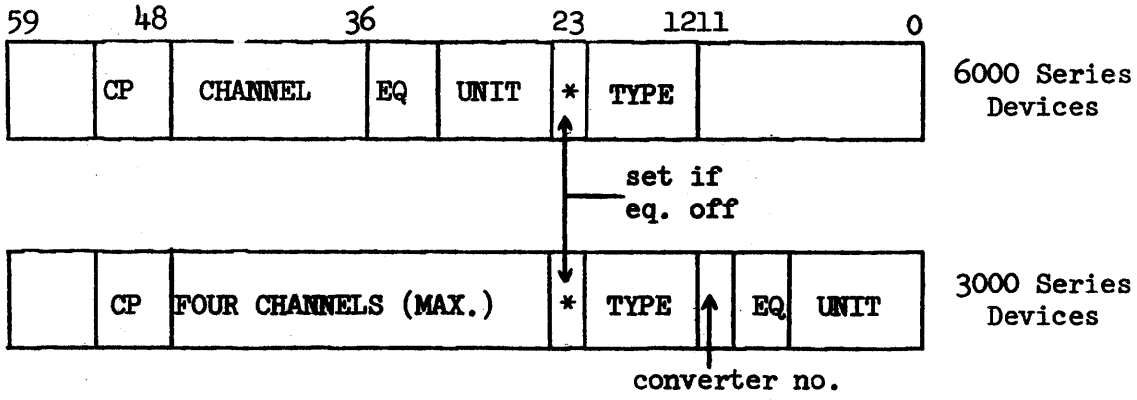


Figure 8. Entries in the equipment status table.

2IO scans down EST, looking for entries with type LP, CP, or CR. These are the line printers, card punches, and card readers, whose operations are controlled by 1IO. The table of available equipments thus contains an entry for each of such equipment in terms of its location in EST and type. 2IO returns to 1IO when the table is completed and saved in the field length of control point 1.

1IO scans the table of available equipments and starts the input or output process for an equipment that is ready to be served. It maintains a pointer on the table, and picks up the first equipment that

is not assigned to any control point (CP field = 0) and not turned off for any reason. It issues an MTR function request to reserve that equipment, which puts the number 1 in the CP field. It then issues another MTR request to reserve a channel for the equipment and uses it to sense the status of the equipment. If the equipment is a card reader and no card has been put in the hopper, 1IO will consider it not ready, and will issue monitor requests to drop the channel and the equipment. It then proceeds to check the next equipment in the available equipment table. If no equipment is ready at this time, 1IO is put on recall. The current input register is saved in word 25_8 of the control point area (RLPW) and PP1 is returned to the pool of available PPU's. Part of MTR's main loop is to check the activity at each control point. When MTR notices control point 1 has a PP program on recall, it finds a pool PPU (if there is one available) and copies word 25_8 of the control point into the input register of the chosen PPU. The PPU then starts the loading process for the PP program and MTR proceeds on its normal business. In the case of 1IO, the steps of loading and execution are repeated as described with the exception that 2IO need not be loaded again since we have already constructed the table of available equipments and saved in the field length of control point 1. This short cut is made possible by setting bit 41 of the input register.

Assume we have a number of job decks sitting on a card reader. The first one is a FORTRAN job deck which is set up as the following:

```
MS175
99999,CENTER,CM50000,P5,L500,T50.
RUN(S)
LGO.
eor card (7/8/9)
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
                .
                .
                .
      READ (5,51) list
                .
                .
      WRITE (6,61) list
                .
                .
                .
      END
eor card (7/8/9)
      Data cards
eof card (6/7/8/9)
```

When the "READY" button on the card reader is pushed, the first card is immediately read into the buffer of the card reader. This also signifies the computer that the card reader is ready. 1IO asks MTR for 1100_8 words of buffer storage, and puts itself in recall status.

It saves the contents of the input register in word 25_8 of the control point area but, instead of returning the PPU to the pool of available PPU's, it changes the input register directly to load the driver package 1CD. This eliminates the monitor steps of dropping and requesting a PPU.

Before we read in the deck, we have to take some time to explain the files in the system. Most information in the system exists in the form of files. They are stored on disks, tapes, etc. and are identified in the central memory by a 7 character name and a control point number. (Files not associated with active jobs are assigned to control point 0, which is a pseudo control point). The system handles the files through the file name table/file status table (FNT/FST), which is a block of storage in CMR (Figure 1). A file has an entry of two consecutive words in FNT/FST. The first word is the FNT entry whose format is shown in Figure 9.

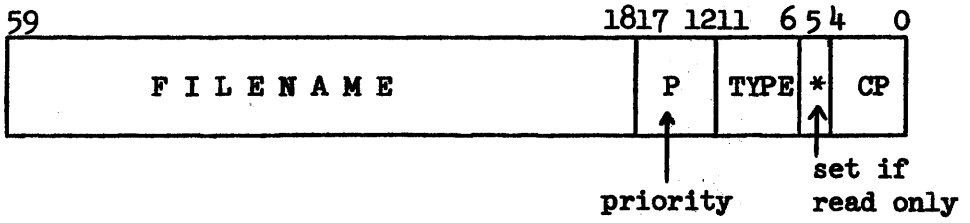


Figure 9. Entry format of the file name table

There are many types of files and they have different FST entry formats. For example, the system library programs are made a read-only common

file (Type 2). Common files have unique names and can be used by more than one job. They are not saved across dead starts and the permanent files (Type 7) are used to store more permanent information. A job may create local files (Type 3), which are destroyed when the job terminates. The FST entry for files stored on a mass storage device has format shown in Figure 10.

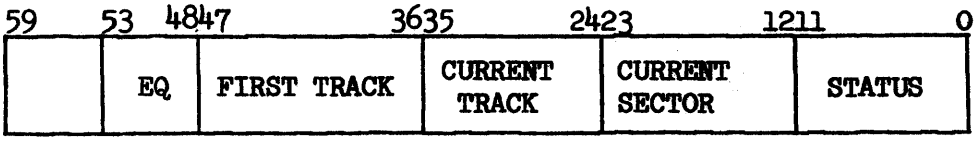


Figure 10. Entry format of the file status table for a file stored on a mass storage device.

Other types of files such as input, output, and rollout will be discussed later.

When a job wants to reference a particular file, it searches the FNT entries of the FNT/FST table starting from the beginning. If the named file is found not assigned to any control point (CP=0), the two word entry is saved in the PPU. It then makes an MTR request for Channel 15. If the request is granted, the PPU reads the two word entry and compares with that saved previously. If they are the same the PPU changes the CP field to the current control point number and releases the channel. The above steps are needed to prevent conflicts in assigning files. Since all PPU's work independently, it is possible that two or more PPU's want the same file at the same time. Channel 15

is set up as a pseudo channel which must be reserved to assign FNT entries. Since MTR processes PPU requests one by one, only one PPU can get to the file. The same concept is used to limit the access to FST (Channel 14) and ECS (Channel 16).

1CD is responsible for converting job decks into input files (or output files if there is an error) to be processed by the system later. It is the combined driver of all the readers, printers, and punches. Up to eight of these devices may be driven at any one time. It has a small monitor program within the package (the equipment manager) to divide the attention of the PPU fairly among the active equipments. Although we have only one card reader active at the time, the reader is reminded that in the process described below, a large number of branches are made back to the equipment manager just to make sure every request is handled in a short period of time.

1CD reserves Channel 12 at all times. If there is no equipment active, it releases both the channel and the PPU. When the equipment manager recognizes a card reader request, it branches to the card reader driver. The first column of each card is sensed before it is read into a buffer in the PPU. The first four cards in our test deck do not have special punches in the first column and are read in one by one. Each character is translated into 6000 series display code and the trailing blanks of a card are deleted. The card image is then written into the buffer of control point 1 in the central memory. A card image is terminated by one or more zero bytes up to a word boundary in the central memory. When the first end-of-record card is reached (a 7-8-9 multiple

punch in column 1) or the buffer is completely filled, the reading process is interrupted. The PPU makes an MTR request for another PPU to execute the PP program TJC, which translates the job card.

Normally a pool PPU is not allowed to make a request for another PPU to perform part of its task without returning itself to the pool. This rule is set in order to avoid dead-lock situations such as all PPU's are busy waiting for the service of another PPU. 1CD is exempt from this restriction because in the normal multiprogramming mode, it is usually kept very busy serving the up to 6 buffers at the BATCHIO control point. As mentioned before, it would go away promptly when there is nothing to do.

The buffer(s) at the BATCHIO control point is a circular buffer. It has a five word block of central memory in the field length of the control point which describes the buffer status at all times. The five word block is called a file environment table (FET)*, which is shown in Figure 11.

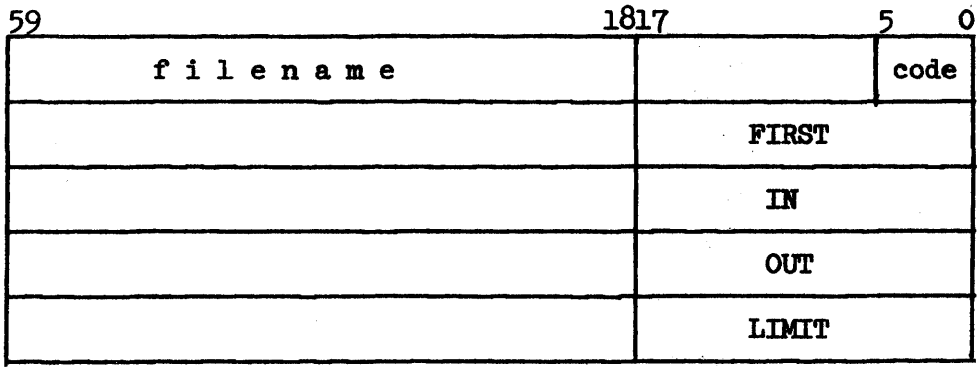


Figure 11. A file environment table

*This is the simplest form of an FET. An FET may be longer than five words to store other information about the buffer.

A buffer is usually set up for read/write processes between the central memory and the disk. The FIRST pointer in the FET points at the beginning word of the circular buffer whereas the LIMIT pointer points at the word immediately after the last word. When the buffer is used for the first time, both the IN and the OUT pointers are the same as the FIRST pointer. In a reading process, a PPU fills the buffer starting from the position pointed by IN. IN is incremented as the buffer is being filled. A CPU can take the information to the running program by emptying the buffer starting from the word pointed by OUT. OUT is also incremented as the buffer is being emptied but in no case should it be incremented past IN. The buffer is circular as when IN reaches the LIMIT, it is immediately decremented to start from the FIRST again with the restriction that it should never be incremented past OUT. Therefore the active record in a circular buffer lies from OUT to IN - 1, and the buffer space available lies from IN to OUT - 1. Both pointers will be turned around the end when necessary. The buffer capacity is LIMIT - FIRST - 1, and it is considered empty when IN = OUT. The same logic applies to a writing process. This type of buffer structure allows the efficient use of buffer space as well as the overlapping of the processors.

When TJC is called by 1CD, the input register of the PPU contains the address of the FET as the argument. The code field in the FET is set to reflect that this call is from the BATCHIO job. The first card in the job deck contains a five character job name (MS175 in our example). TJC creates a seven character name for the job by asking MTR for a two digit job sequence number. Word 22_8 of the central memory (JSNL) contains a job counter. MTR reads this word, increments it, passes it back to TJC which then concatenates the number to the user's job name. This process will distinguish jobs submitted with the same job name card except in

the case when they are entered exactly a multiple of 100 jobs apart. TJC finds the system accounting file in the FNT, marks the FST entry busy and reads the record for the given account number from the disk. It uses the accounting information to verify the parameters of the job card. It also computes the "queue priority" of the job using all parameters. The value normally lies between 1000_8 and 4000_8, and is higher if the job requires less system facilities. The queue priority is used to schedule the job during normal execution and will be described in the next section. All the bookkeeping information are saved in a block equivalent to six central memory words, to be transferred to the buffer later.

TJC loads a location-free PP overlay named OBF which begins a file by allocating an entry in the FNT/FST. OBF first searches the FNT for the seven character job name. If the name is found, OBF returns and TJC asks the monitor for a new sequence number. This procedure enables the system to distinguish jobs submitted with the same job name card exactly a multiple of 100 jobs apart. OBF normally returns with the address of the allocated FNT entry.

TJC assigns an initial queue priority to the job with the value dependent on the job category. A job may be a local batch job (as our test job), interactive job, PROCSY* job, or export/import job. This information is passed to TJC using the code field in the FET. The

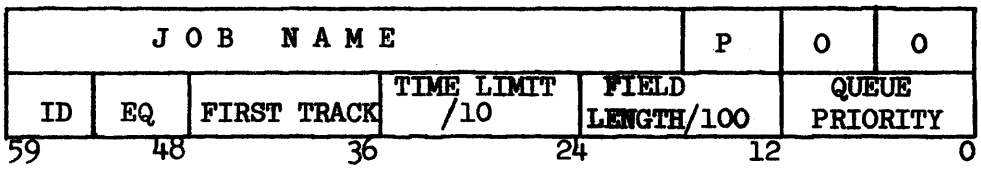
---

*Purdue Remote Online Computing System

value of this initial priority lies between 6000_8 and 6600_8. Its use will be described in the next section.

TJC returns all the information it has obtained by translating the job card through several unused fields in the FET and the first six words of the central memory buffer. (By convention, the job name card image starts at the 7th word when TJC is called.) The six words contain the accounting information as mentioned before, or error messages in case of job card error. The last bit of the code field in the FET is set to 1 to indicate completion and the PPU is returned to the pool.

The equipment manager in 1CD constantly checks the completion bit of the FET. When it notices that TJC has finished, 1CD asks MTR again for a PPU to execute the PP service program 1PS. 1PS loads the appropriate mass storage driver (in this case 2WD) which dumps the buffer on disk and updates the FST entry. When 1PS finishes and drops the PPU, 1CD proceeds to read in the remaining cards of the input deck. It reads up to the end-of-record card or until the buffer is filled as described before. 1PS is then called again to write the buffer on disk. When the end-of-file card is reached, 1CD changes the file to be of type input (Type 0), and the FNT/FST entry is shown in Figure 12. The job is said to have joined the "input queue".



P: Priority on job card

ID: Identification of the I/O terminal used by the job

EQ: Equipment number of the disk used

QUEUE PRIORITY: The initial queue priority of the job

Figure 12. FNT/FST format for a job in the input queue

If there is a job card error, only the first six words of the buffer is written on disk. The remaining cards of the job deck are flushed. The job file is made of type output (Type 1) in the FNT/FST when the end-of-file card (a 6-7-8-9 multiple punch in column 1) is reached. The job is said to have joined the "output queue", waiting to be printed (in this case with a very high queue priority).

After the set up of the FNT/FST entry, 1CD issues an MTR request for the job scheduler (1SJ). The scheduler selects the jobs for execution and will be described in the next section. 1CD resets the buffer parameters and returns to the card reader driver, which reads the following job decks in the same way.

The reader is reminded that 1IO was put in recall status when 1CD was first loaded. MTR assigns another PPU to execute 1IO later which resumes the scanning of the table of available equipments.

If there is another equipment requesting service, 1IO checks the buffers in the field length of control point 1. A new buffer will be set up (up to a maximum of six buffers) if necessary. The driver request is set as a buffer parameter and if 1CD is still active at a PPU, 1IO releases its PPU and goes on recall again.

The equipment manager of 1CD monitors the status of each buffer area. The driver requests are answered in a round-robin fashion, with a variable time slice determined by individual driver procedures. This design allows the efficient use of the PPU and the channel.

IV. Job Scheduling

The operator may start the execution of jobs by typing in

n.NEXT.

at the console, where n is any control point number that is not yet assigned. Multiprogramming mode can be achieved by entering NEXT at all control points. Usually the system is started for full multiprogramming mode by typing in

AUTO.

This command has the effect of assigning BATCHIO to control point 1, and NEXT to all other control points.

When DSD recognizes the n.NEXT command, it branches to a package called NXT. NXT changes the control status of the control point area (word 60) to make it available for scheduling and issues an MTR request for the job scheduler. DSD then resumes its normal acitvities. Since the previous section is also terminated by the request for the job scheduler, we shall join the discussions by explaining the priority scheme used.

There are maximum and default priorities assigned to each account number. The PP program TJC (described in the previous section) uses the job category and the priority specified on the job card of the program, but in no case greater than the maximum allowed, to compute the initial queue priority of the job. This is stored in the FNT/FST when the job is in the input queue (Figure 12). It is intentionally made very high (6005_8 for our example) in order to support the policy

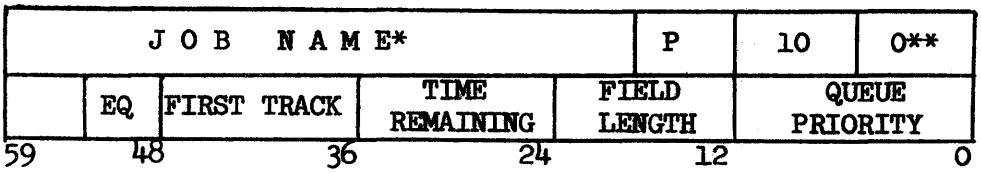
that every job is to be given a time slice in the central memory shortly after its arrival. As described earlier, TJC also computes the queue priority of the job using all parameters supplied on the job card. This is going to be the queue priority of the job when the "first pass" time slice (currently 25 seconds of combined CPU and PPU time) is expended. This priority changes from time to time to reflect the job status until the job terminates.

The job scheduler is a PP program named 1SJ which assigns jobs to control points according to their priorities and field lengths. It also calls a PP overlay 2SP at regular intervals to update the priorities of the waiting jobs on disk. We shall discuss part of its functions by following our test job.

Suppose MTR picks up the 1SJ request from PP9, caused by a 2.NEXT. command. It writes a call for 1SJ in the input register of an available pool PPU and sets up a flag so that as long as 1SJ is active in one PPU, requests for 1SJ will be answered directly by clearing the output register of the requesting PPU. This precaution is necessary because 1SJ is called very frequently from independent sources as we have already seen in our example, both DSD and 1CD request 1SJ at about the same time. Like other PP programs, 1SJ branches to an initialization routine right after it is loaded. It then checks the control point areas for the status of each control point. It constructs a table of control points ordered by ascending priority. At this time, only control point 1 is occupied (by BATCHIO). It makes

any other available control points "NEXT", which is displayed by DSD in due course. Since there is no program to be "rolled out", 1SJ searches the FNT/FST table for files of Type 0 (input) or Type 10 (rollout, to be explained later). They are jobs in queues waiting to be processed. If there are more than one job waiting, 1SJ chooses the job with the highest priority which will fit in the available memory (the central memory blocks assigned to jobs of lower proiorities are also considered available in this case). Suppose our test job is chosen, 1SJ proceeds to initiate it at a control point. This includes issuing MTR requests for the amount of storage required by the job (50000_8 words), and changing the appropriate fields in the control point area for the job such as the job name word (word 21_8). 1SJ then asks for another PPU to load and execute the program 1AJ (Advance Job Status). We pause to discuss some of the features of the system at this time but the eager reader may still follow the test job by proceeding directly to the next section.

Waiting jobs in the system belong to two types of queues. The input queue is constructed by 1IO and contains jobs that have just entered the system through input devices. It is described in the previous section. When a running job requires operator attention, such as a tape mount request, it gives up its control point and the central memory while waiting. The job is said to be rolled out and enters the rollout queue. A running job may also be rolled out if there are jobs with higher priority waiting for the storage. The FNT/FST entry of a job in the rollout queue is shown in Figure 13.



*ROLOUT if the file is assigned to a control point (rollout/rollin in progress).

**The control point number in the above situation.

Figure 13. FNT/FST format for a job in the rollout queue

The rollout file of a job contains all information needed to restart the job. The FNT/FST entry of the rollout file replaces the original FNT/FST entry of the job, which was created by TJC. The file contains all the FNT/FST entries for other files associated with the job, the control point area, the dayfile buffer, the contents of its field length, and the PPU delay stack entries. The description of this process will be left to a later section.

It is the responsibility of the scheduler to roll jobs in and out and to ensure the efficient use of the central memory. Usually the storage blocks occupied by jobs are scattered in the central memory following the CMR. An example is shown in Figure 14. Field lengths of control points are ordered according to the control point number. Since jobs may terminate or be rolled out at different times, there may be gaps between field lengths in the central memory. Ideally the memory space available is the memory not currently being used. This implies

that all the gaps between field lengths be treated as a contiguous block. This is made possible by the use of hardware relocation registers, which enables the system to move blocks of storage around to obtain the needed space easily. The storage management facility can be invoked by an MTR request which is described in Section VII.

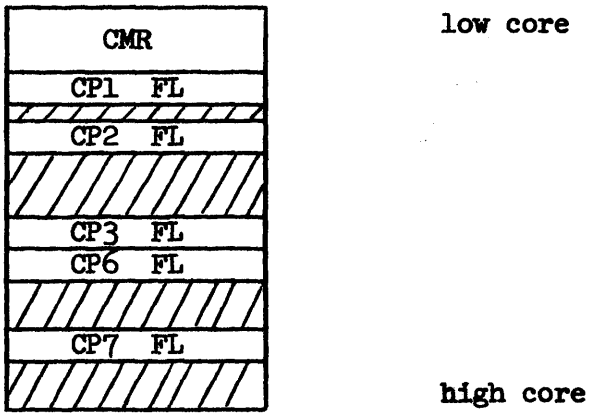


Figure 14. A central memory layout example

As mentioned a little while ago, 1SJ constructs a table of control points ordered by ascending queue priority . The jobs at control points with queue priorities lower than a fixed value (currently 100_8) will be unconditionally rolled out of the central memory. At times the system may permit every job to stay at a control point for at least a real time period which is determined by the console operator. The scheduler considers a job to have a very high queue priority within that time period so that it will not be inadvertently rolled out. 1SJ searches the FNT/FST table for jobs waiting in the input and rollout queues. The highest priority job which will fit in the available memory (with the necessary

rolling out of lower priority jobs implied) is chosen. 1SJ then rolls out the smallest number of lower priority jobs which gives the system enough free memory space and/or a free control point. It sets the queue priority of the job's FST entry to zero so that it can be skipped by future scheduling passes. 1SJ then issues an MTR request to obtain a contiguous block of storage for the job. It asks the monitor for another PPU to execute the PP program 1AJ (Advance Job) or 1RI (Roll In), which loads the job. The entire procedure is repeated as long as there is memory available and jobs waiting. For reasons such as a change of job status, or a PPU is not available, etc., the scheduling is aborted. 1SJ puts itself in recall status and will start the scheduling process anew next time.

There is another constraint on the scheduling process. The system defines five job categories using disjoint ranges of the queue priority. They are the interactive jobs, PROCSY jobs, export/import jobs, local batch jobs, and the jobs in the rollout queue. There is a limit on the number of active jobs for each job category. A proper job mix can be maintained this way. The limits may be temporarily lifted for the efficient utilization of memory if there are jobs waiting and memory available but no jobs can be scheduled.

The scheduler 1SJ is called whenever the state of the system changes or every four seconds, which is the "recall period". MTR maintains in PP0 a table called the PP delayed stack. A PP program may enter the "delayed recall" state by saving its input register and the

time period in the PP delayed stack. MTR checks the stack once in the main loop and then initiates the loading process of the PP program at a PPU if the delay time has expired.

1SJ is responsible to call the overlay 2SP (Set Priorities) periodically (every 10 seconds). 2SP increments the queue priorities for the waiting jobs in the FNT/FST. It also checks the accumulated CPU and PPU running times of jobs at control points who are still in their "first passes". The initial high priority of a job will be changed to the queue priority computed by TJC if the first pass time slice is expended.

V. Execution

The scheduler has picked our test job for execution. The queue priority of the job's FST entry is set to zero which causes the scheduler to ignore the job file during future passes. Suppose control point 2 is assigned. The new job name (MS17500) is displayed at control point 2 instead of the original name (NEXT) when DSD updates the console displays in its next loop. MTR loads the PP program 1AJ for the control point and the scheduler returns to the loop to schedule the other jobs that followed MS175 in the input queue. The scheduler will return its PPU to the pool when there is no job that can be scheduled.

1AJ is a PP program that advances the status of a job at a control point. It has overlays to process control cards and handle execution errors. When it is first called for a job, 1AJ issues an MTR request to load the overlay 2BJ (Begin Job). 2BJ reads the real time clock and initializes the various timer fields in the control point area. It also sets up word 67_8 (CSPW) of the control point area, which contains the control statement pointers (Figure 15).

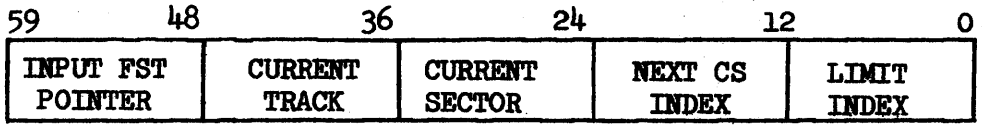


Figure 15. The control statement pointer word of a control point area.

The leftmost byte of CSPW contains the pointer to the FST entry of the job file. The current track and sector are also taken from the FST entry, and the two indices point at the beginning of the control statement buffer in the control point area (words 70_8 to 177_8). This is a circular

buffer filled and emptied by the routine RNS (Read Next Statement) in the overlay 2TS (Translate Control Statement). If the buffer is empty when RNS is called, it reads in a sector of words in the job file. The current track and sector pointers are updated so that the next read will start at the correct position on disk. Since a sector in the system contains 320 data bytes, or 64 central memory words, the reader may wonder why the buffer has a length of 72 central memory words. We recall that the input cards are packed in the job file in a way that all trailing blanks are deleted. It is therefore possible for a control statement to reside across sector boundaries. The extra 8 words (an 80-column card image) in the buffer is needed to handle such a situation.

2BJ loads the overlay 2TS and branches to the routine RNS. RNS performs the steps needed to read the disk, which include loading the appropriate driver if necessary, reserving the right channel and dropping it when data have been transferred to a buffer area in the PPU. The data are subsequently transferred to the buffer in the control point area and the indices in CSPW properly updated. The reader is reminded that TJC constructs six words of accounting information from the job card and they are put at the beginning of the job file. 2BJ uses the information to initialize certain accounting fields in the control point area and also saves the six words as they are starting from word 41_8 (ACTW) of the control point area. The next control statement index (Figure 15) is incremented so that it points at the job card of the file.

After the record of control statements is taken care of, the FNT/FST entry is assigned to the control point and the name is changed to "INPUT". The FST entry is changed to the format shown in Figure 10, with the track and sector number pointing at the beginning of the next record (the FORTRAN PROGRAM card in this case). This is accomplished by reading and skipping the data of the job file until the first end-of-record card is reached. The job card is written in the job's dayfile and 2TS starts processing the first control statement.

The system provides a dayfile for each job being processed. It contains the history of the job in the system. Information such as the time each control card is processed, error messages, and special requests of system facilities are recorded. The accounting information is put at the end when the job terminates. The system also keeps a dayfile which is a complete record of everything that happened in the system, including the dayfiles of individual jobs. The dayfiles are handled through a table of dayfile pointers in CMR (Figure 1). The table contains a two word entry for each control point and the system dayfile. An entry contains information about the dayfile buffer in the central memory and the current location on disk. It is similar to an FNT/FST pair. A job at a control point may write dayfile messages in the buffer which is then transferred to the disk. The dayfile buffer and pointers are copied to the rollout file when the job gives up the control point, and the entire dayfile is copied to the output file when the job terminates.

The first control statement of our test job is RUN(S). 2TS recognizes that RUN is the name of a program in the system library on disk. This is accomplished by scanning the library directories in CMR. 2TS prepares the PPU to read the program by requesting the system disk (Disk 0) and the channel (Channel 2), and sets the mass storage driver as we have seen in several occasions so far. The program is read into a buffer in the PPU sector by sector. The first sector contains the load address of the program and the program format. The program file is transferred to the field length of the control point starting at the load address. The channel is released when the loading is completed.

The user may specify different options of the FORTRAN compiler. The options are passed to the compiler program as arguments enclosed in parentheses following the program name. For example, our test job specifies the "S" option, which means to compile with source list. There may be more than one argument for a program. 2TS stores them, in display code, one for each word starting from RA + 2 of the field length. The compiler program is in absolute binary code with table of constants at the lower end of the field length and the first executable instruction at location RA + 100_8. 2TS clears the exchange package area of control point 2 and sets the field for the P register to 100_8. 2TS issues an MTR request for a CPU, sends the control statement to the dayfile, and returns the PPU to the pool.

Since there is no other activity except BATCHIO going on the system,

MTR finds an available CPU very easily and asigns it to control point 2 using an exchange jump. The exchange package area of control point 2 now contains the idle exchange package and the CPU starts executing the instruction at location RA + 100_8.

We shall not go into the details of the FORTRAN compiler. Briefly the program creates several files local to the control point. Two of the local files are given the special names "LGO" and "OUTPUT". It reads the source statements from the local INPUT file, lists them in the OUTPUT file, and translates them into relocatable binary form which is then stored in the LGO file. It loads several overlays in the process and saves temporary results on local scratch files. The program terminates by storing "END" in location RA + 1.

The first word of a user field length (location RA) is reserved for use by the hardware in case of an arithmetic error. If an arithmetic error occurs the type of error and the address at which the error occurred is stored in location RA. Location RA + 1 is used as a communication area between central programs and MTR. The leftmost 18 bits of the word is used to store a three character message in display code. There are four special messages which are explained as the following.

1. ABT: To abort the control point.
2. END: To end the central program by taking the CPU away from the control point.
3. RCL: To place the CPU on recall.
4. TIM: To find current time.

A central program may also call PP programs by putting the PP name in RA + 1. Any PP program whose first character is a letter may be called

in this way.

MTR checks the central programs who are currently having the CPU's during each pass of its loop. Any request put in location RA + 1 will be processed. MTR also checks for arithmetic errors which are indicated by a zero P register. When MTR recognizes the END message in RA + 1 of control point 2, it searches all other control points for the one with the highest CPU priority. An exchange jump is issued which puts the CPU to the idle loop and stores the exchange package of control point 2 back to its exchange package area. Another exchange jump will let the CPU execute the highest priority job. The CPU also updates the accumulated times for control point 2 during the idle loop.

In addition to responding to the special requests from the PPU's and CPU's, MTR checks the activities of control points regularly. If it checks the activity of a control point at one time, then it checks the next control point after a fixed time period (currently 65 milliseconds). The PP recall register in the control point area is checked and if set, the PP program is called again.

The leftmost byte of the status word (Figure 7) is called the processor status byte. The format is shown in Figure 16.

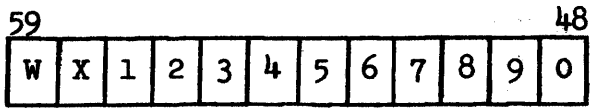


Figure 16. The processor status byte of the status word in a control point area

The program at the control point is waiting for a CPU if the W bit is set. The program is in recall status if the X bit is set. This means that the program is waiting for some PP program to finish (automatic recall) or for some fixed time period (periodic recall) before it can proceed further. In case of the automatic recall, the bit corresponding to the PPU is also set.

MTR also checks the processor status byte of the control point. If the W or X bit is set, MTR makes an attempt to start a CPU for the program. It then returns to its main loop. If there is no activity at the control point, MTR advances the status of the control point by calling the PP program 1AJ again (or 1RO if the control point is flagged for rollout).

At this time MTR notices that control point 2 has no activity. It loads 1AJ for the control point which proceeds to translate the next control statement.

The next control card of our test job is LGO. By scanning the FNT/FST table, the overlay 2TS finds a local file with that particular name. (The reader should recall that this is a file created by the FORTRAN compiler.) It rewinds the file by setting the current track equal to the first track in the FST entry (Figure 10). The status byte of the FST entry indicates that the file is of relocatable binary form, therefore cannot be loaded directly by the PPU. 2TS prepares the call to the system loader by setting loader control flags in the control point area and putting the file name (LGO) in location RA + 64_8.

It loads the absolute loader program LDR= into the field length, requests a CPU, sends the control statement to the dayfile, and returns the PPU to the pool just as in the case with the FORTRAN compiler program.

LDR= is the first entry name of the system loader package. The loader loads a file from disk, recomputes address constants if necessary, and attempts to satisfy all the external references of the user program. The reader should consult the documentation for the loader if he is interested in the details of operation. Since the loader is a central processor program, it uses the standard PP program CIO (Combined Input-Output) to communicate with files. We shall pause to discuss it briefly.

CIO is the PP program that processes input and output requests for CPU programs. Although it has the capability of accessing all the system I/O devices, CIO is used mainly for mass storage devices and magnetic tapes. In order to perform an I/O function, the central program must first prepare a buffer larger than a physical record (64 central memort words, or 320 data bytes, which is the size of a sector on disk). This is a circular buffer as described in Section III, with an FET similar to the one shown in Figure 11. In this application the FET may have more than five words. The first eight words of a long FET is shown in Figure 17.

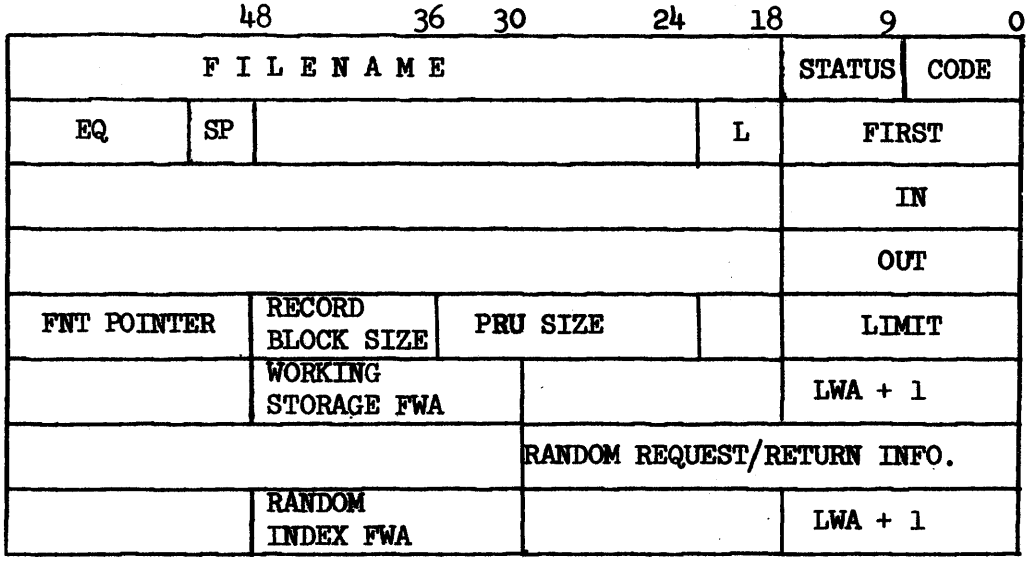
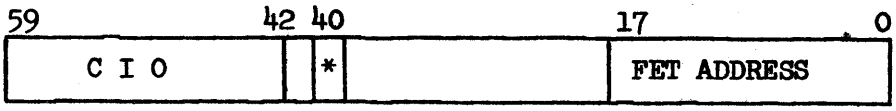


Figure 17. A file environment table used by CIO.

The code field in the first word contains a completion bit (Bit 0). The L field contains the number of words in the FET in addition to the five basic words. If it is non-zero, the special processing field (SP) tells whether it is a random access file, etc. (Random access files are files with logical records that can be directly accessed without reading or writing the entire file.) The three words following the LIMIT word are used for random access files. The programmer may append other information about the file following the eight words shown in Figure 17.

The central program enters the CIO request in location RA + 1, which is shown in Figure 18.



*Automatic recall bit

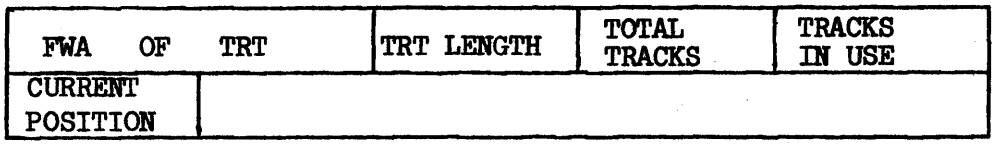
Figure 18. Format of the call to CIO.

When MTR recognizes this request, it assigns a pool PPU to the control point and loads the PP program. When CIO terminates successfully, it sets the completion bit of the FET (bit 0 of the first word) to 1 and drops out. The CPU program can therefore determine the status of the I/O process by checking the completion bit of the FET. However, in many applications, the CPU cannot proceed before the I/O function is completed. Instead of wasting time in a tight loop, the programmer may elect to put the CPU "on recall" by setting the automatic recall bit to 1 for the CIO call. In this case MTR takes away the CPU and sets the X bit and the PPU bit of the status word of the control point area (Figure 16). The output register address of the assigned PPU is also stored in the control point area (Word 22_8) so that when MTR checks the recall status of this control point later, it can readily determine whether the requested PPU function has been completed. An attempt will be made to restart a CPU if there is no PP functions to wait for.

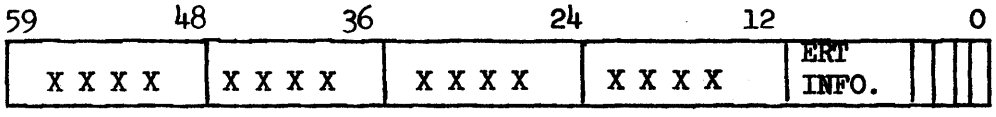
The left most byte of the LIMIT word in the FET may contain a pointer to the FNT entry of the file which is checked by CIO. If the pointer is not supplied or the file names do not match, CIO makes a

complete search of the FNT/FST table for the named file assigned to the control point. It will create an entry for the file if it is not already in the table and sets the address pointer in the FET. The buffer parameters are also checked before any action is taken.

We shall concern ourselves only with the mass storage I/O at this time. For a "read" request, CIO loads and executes the overlay 2RD (Read Disk). 2RD uses the PPU mass storage drivers to read data from the disk sector by sector. The buffer in the field length is filled in a circular fashion and the OUT pointer is checked after each sector read. 2RD usually reads up to the end-of-file mark, but it will stop and exit when the buffer is filled. It can also be instructed to stop after a sector or at an end-of-record mark. For a "write" request, CIO uses the overlay 2WD (Write Disk) to empty the circular buffer. 2WD decides the location to start writing using the track reservation table (TRT) of the disk. Figure 19 shows the formats of TRT entries.



(a)



(b)

Figure 19 (a). Track reservation table pointer format
(b). Track reservation table entry format

There is a track reservation table for each mass storage equipment. The TRT pointers reside in CMR (Figure 1) and can be referenced using EST entries (Figure 6). An entry in TRT contains information about four tracks. The right most four bits of the word are the track use bits. A track is considered in use if the corresponding track use bit is set to 1. If the leftmost bit of a track byte is 1, the remainder of the byte points at the next track of the file. If the bit is 0, the remainder is the number of sectors in use of the track, which is also the last track of the file. The ERT INFO field is used to control the access of disk tracks and will not be discussed here.

2WD computes the number of sectors to write and determines if there is enough room in the current track using the TRT. If more tracks are needed, it attempts to reserve them using an MTR request. This function is actually carried out by a CPU. MTR passes the request using its output register (Word 51_8 of CMR) and begins the central program, which is considered to be assigned to the pseudo control point n + 1. The CPU reserves the unused tracks by setting their use bits and next track pointers in the TRT. The buffer is then dumped using the mass storage drivers in the PPU. We shall return to the loader without further digression.

The loader reads in the LGO file using CIO. It proceeds to modify address references when necessary and tries to satisfy the external references. It may have to load several programs from the

system library on disk. When the object program is completely edited, the loader moves a few words of code to the high end of the current field length. The execution of this piece of code moves the object program down, thus writes over the loader. It then puts a call to the PP program RFL in location RA + 1 with the length of the object program as an argument. The CPU then branches to RA + 100_8, which starts a loader supplied loop to wait for the acceptance of the RFL request. The PP program RFL reduces the field length to the size used by the object program. The CPU branches to the first instruction of the object program when RA + 1 is cleared.

The READ statement in our test program calls CIO to read the local INPUT file and the WRITE statement calls CIO to write the local OUTPUT file. The central program is put in the automatic recall status during these operations. It may give up the CPU at other times such as monitor function requests from the PPU's. It may also be interrupted and put in wait status by a job with a higher CPU priority. It may even be rolled out and restarted later if there is a higher priority job waiting for the storage. The END request is put in location RA + 1 when the test program terminates.

VI. Job Termination

When 1AJ is called for a job and there are no more control cards to be processed, it sets the PPU input register to 1CJ and branches to the PP resident program. Thus the PPU loads the PP program 1CJ (Complete Job) and starts execution. 1CJ reads the real time clock and updates the running times for the last time. Future processing to get the job out of the system will be considered system overhead and will not be charged to the user. It issues an MTR function to release the field length of the control point, since all information is saved on files and the central program is no longer needed. It then proceeds to release the files that also are not needed.

In the case of our test job, all information is written on the OUTPUT file of the control point. (The dayfile is not considered a local file and is treated differently). Local files such as LGO and INPUT are no longer needed. The job may have used other types of files such as the common files, permanent files, etc., which should also be released. In other words, the only files to be saved are the OUTPUT file and the various punch files if they are used.

To drop a file, the PPU loads the location free overlay ODF. The simplest file drop is to drop a local file on a mass storage device. ODF issues an MTR request to drop the tracks used by the file and MTR uses a CPU monitor program to modify the TRT, as in the case of reserving tracks. The FNT/FST entry is then cleared, which completes the process.

a buffer in the field length of control point 1 and loads the PP program 1CD if the latter is not already active at some other PPU. 1IO then enters the recall status again.

When the equipment manager of 1CD recognizes a line printer request, it branches to the line printer driver. The driver issues an MTR request for another PPU to execute the service program 1PS, which loads the overlay 2LD. 2LD loads the initial print data in the buffer, which is the banner page made of the job name in three-inch composed characters. The driver interpretes the coded information in the buffer to make appropriate format controls and starts printing. 1PS is called again and again to fill the buffer in the central memory with the data on the output file using the overlay 2RD. If for any reason the printing of the output file must be aborted, 1CD uses the dayfile pointers of the FST entry to print the dayfile record immediately. This proves to be quite useful to the user sometimes. When printing is completed, 1CD resets the buffer, clears the FNT/FST entry of the file, issues an MTR request to drop the tracks, and returns to its equipment manager to handle other driver requests.

Finally, when an operator notices that a line printer has had some activity, he walks over to check. Our test job is detached from the printer and put in Box 175, waiting to be picked up by the programmer.

VII. Storage Management

In Section IV we mentioned the fact that it may be necessary to move several field lengths in memory at times in order to obtain a larger block of contiguous storage for a control point. This is usually called the relocation process. The relocation of jobs can be done very easily in this computing system since each CPU has a hardware relocation register. A central program always starts at location zero in its field length. All address references are defined relative to location zero. The reference address (RA in Figure 3) is the real hardware address of the job's location zero when it is running. This value is kept in the relocation register of a CPU and added to each address reference during execution to obtain the real address. After the program is moved only the reference address needs to be changed in order to resume execution.

The rightmost byte of word 1 (MFLL) in the central memory contains the total field length of the machine. The size of CMR is stored in the rightmost byte of word 20_8 (STSW). After dead start, MTR takes the difference of these figures and stores the available field length in the rightmost byte of word 56_8 (CMCL). MTR also maintains a table of unassigned field lengths in PP0, ordered by control point numbers. Each entry in the table is the amount of storage in the gap between the last word of the field length of this control point and the first word of the field length of the next control point. Field lengths are measured in units of 100_8 central memory words.

MTR handles all requests for storage. A request will be ignored if the available field length (location 56_8 of the central memory) is not large enough to satisfy the request. Suppose control point 6 in Figure 14 requires storage. If the gap between the field lengths of control points 6 and 7 is large enough, MTR assigns the requested increment to control point 6 and updates the records. If the gap is not enough to satisfy the request, MTR will try to move the field lengths of the lower control points (control points 3, 2, and 1, in this order) toward the low core until the needed room is obtained. If this is still not enough, it will try to move the field lengths of the higher control points (control point 7 in this case) toward the high core.

After it has determined which control point(s) to move, MTR moves the field length(s) one at a time. If the control point to be moved has active PPU's assigned, the storage move is delayed and MTR returns to its main loop. The system programmer is therefore required to put "pause for relocation" functions in PP programs which might be assigned to control points for a length of time. This will prevent the unnecessary inhibition of storage move processes. Next MTR will interrupt the central program if it is being executed. Word 20_8 of the control point area is the status word of the central program (see Figure 7). The leftmost byte is the processor status byte which is shown in Figure 16. As mentioned in Section 5, the central program is waiting for a CPU if the W bit is set. It is in recall status if the X bit is set. In either case the program is not being executed. MTR clears the W and X bits and saves the current status. If the program is currently being executed, MTR

finds the waiting job with the highest priority and issues exchange jump instructions to switch the CPU to the waiting job. The original job is put in W status and handled the same way as described above. In other words, the central program is made ineligible to be processed by a CPU until the move process is completed. At that time the original status will be restored.

Since moving a block of information in the central memory requires a lot of central memory references, it is performed by a central program. Like the track reservation function described in Section V, the program is not executed in the monitor mode. It is considered to be assigned to the pseudo control point n + 1 and therefore interruptable. MTR puts the requested function code in its own output register, which is location 51_8 of the central memory. It sets up the exchange parameters in the system exchange area and begins the central program. The process is repeated until the requested storage is obtained.