INHOUSE

The Purdue, Dual-MACE Operating System

V. Abell
Purdue University Computing Center

June, 1974

(6/74) LO DMACE

The Purdue, Dual-MACE Operating System

V. Abell
Purdue University Computing Center

June, 1974

TABLE OF CONTENTS

4. (cont.)

Introduction

The Purdue, Dual-MACE operating system, Dual MACE, is a twin machine executive for automatic control of CDC 6000 series computers. It is an extension of the Purdue Mace Operating System [1], which is, in turn, based upon the experimental CDC operating system, MACE [2]. The popular CDC KRONOS time sharing system [3] is also a MACE derivative.

The CDC computers controlled by Dual MACE are multi-processors. At Purdue the two machines, a CDC 6400 and a CDC 6500, have a total of twenty-three processors. The basic design philosophy of Dual MACE is to provide a sufficient but flexible executive that will enable a maximum of resource sharing, a minimum of system inter-dependence, and a minimum of change in external functional appearance from Purdue MACE.

To that end the inter-system coupling is extremely loose. Its basic hardware device is extended core storage, or ECS [4]. ECS is a fast access bulk core storage device with a maximum transfer rate of 100 nano-seconds per sixty bit word. It is designed to permit access to as many as four CDC 6000 systems, and contains an eighteen bit flag register for use in inter-machine reservation control.

In Dual MACE, tables describing common system resources are stored in ECS and accessed by the system monitor programs active in each machine. Various bits of the flag register, and in some cases, sequencing rules and table contents provide the necessary interlocks.

The ECS resource tables permit the sharing of system program libraries, utility disk storage space, and permanent files. In addition the two system monitor programs communicate with each other via ECS areas in order to accomplish job location, job file transfers and various other executive tasks.

Because the systems are so loosely coupled, the same system programs, from compilers through monitors are used in both machines. In a relatively small number of cases a discrete machine number is employed to describe the residence of a particular job, the sequence in which system tables are allocated, or to control the order of initial system loading.

The remainder of this document describes the action of the system program components, with special emphasis on system control and resource sharing. A more user-oriented discussion of Purdue MACE, still applicable to Dual MACE, may be found in the document PMACE [5].

Introduction

The Purdue, Dual-MACE operating system, Dual MACE, is a twin machine executive for automatic control of CDC 6000 series computers. It is an extension of the Purdue Mace Operating System [1], which is, in turn, based upon the experimental CDC operating system, MACE [2]. The popular CDC KRONOS time sharing system [3] is also a MACE derivative.

The CDC computers controlled by Dual MACE are multiprocessors. At Purdue the two machines, a CDC 6400 and a CDC 6500, have a total of twenty-three processors. The basic design philosophy of Dual MACE is to provide a sufficient but flexible executive that will enable a maximum of resource sharing, a minimum of system interdependence, and a minimum of change in external functional appearance from Purdue MACE.

To that end the inter-system coupling is extremely loose. Its basic hardware device is extended core storage, or ECS [4]. ECS is a fast access bulk core storage device with a maximum transfer rate of 100 nano-seconds per sixty bit word. It is designed to permit access to as many as four CDC 6000 systems, and contains an eighteen bit flag register for use in inter-machine reservation control.

In Dual MACE, tables describing common system resources are stored in ECS and accessed by the system monitor programs active in each machine. Various bits of the flag register, and in some cases, sequencing rules and table contents provide the necessary interlocks.

The ECS resource tables permit the sharing of system program libraries, utility disk storage space, and permanent files. In addition the two system monitor programs communicate with each other via ECS areas in order to accomplish job location, job file transfers and various other executive tasks.

Because the systems are so loosely coupled, the same system programs, from compilers through monitors are used in both machines. In a relatively small number of cases a discrete machine number is employed to describe the residence of a particular job, the sequence in which system tables are allocated, or to control the order of initial system loading.

The remainder of this document describes the action of the system program components, with special emphasis on system control and resource sharing. A more user-oriented discussion of Purdue MACE, still applicable to Dual MACE, may be found in the document PMACE [5].

System Architecture

The hardware characteristics of the CDC 6000 series systems are well known [6]. Briefly the machines of the 6000 series consist of one or two central processors (CPU's) and a number of peripheral processors (PPU's) all of which share a large, fast central memory of sixty bit words. The CPU has a minor cycle time of 100 nanoseconds; the PPU, one miscrosecond.

The PPU's each have a full complement of arithmetic, shift, logical and input-output instructions, and 4,096, twelve bit words of private memory. They share access to twelve, one megacycle, twelve bit channels. The PPU's are primarily designed to provide input-output and executive tasks; the CPU's bear the computational load.

Five PPU words of twelve bits can be contained in one, sixty bit central memory word. This is the minimum exchange amount between the PPU and central memory. A twelve bit group in a central memory is called a byte. The bytes are labelled from left to right, zero through four. Byte zero is bits 48-59; one, 36-47; two 24-35; three, 12-23; and four, 0-11.

When a central memory word contains CPU instructions, it is divided into fifteen bit groups called parcels. The parcels are numbered from left to right, zero through three. Parcel zero is bits 45-59; parcel one, 30-44; parcel two,15-29; and parcel three, 0-14.

The Purdue 6500 consists of two CPU's, ten PPU's and 98,034 words of central memory. The 6400 has one CPU, ten PPU's and 65,536 words of central memory. The ECS storage, shared by both machines, contains 503,808 sixty bit words. In addition, both systems have access to a large variety of peripheral devices, including rotating mass storage devices, magnetic tape units, plotters, paper tape equipment, and a number of terminal system, front-end processors. Figure 1 pictures the basic configuration. Note that not all devices are shared.

System executive control is performed by a number of discrete programs, most of which share the occupancy of PPU's and shared areas of central memory. However, two peripheral processors, and a moderate amount of central memory are reserved entirely for system executive programs and job control tables. The central memory required depends to a large extent on changeable parameters, but typically consists of approximately 18,000 words in each machine.

One of the two dedicated peripheral processors contains the console driver program, DSD - Dynamic System Display. DSD maintains a dynamic display of system status on the twin-screen CRT console, and accepts operator keyboard entries. The other dedicated peripheral processor contains the program MTR - peripheral processor monitor. MTR performs a wide variety of system monitoring tasks, which include CPU switching, job request processing, timing controls, system stability checking, etc.

A portion of the dedicated central memory contains an additional executive program, CPUMTR - central processor monitor. CPUMTR, in contrast to MTR, is an event driven executive, processing requests for resource and system activities upon specific stimulation by the CPU's and the PPU's.

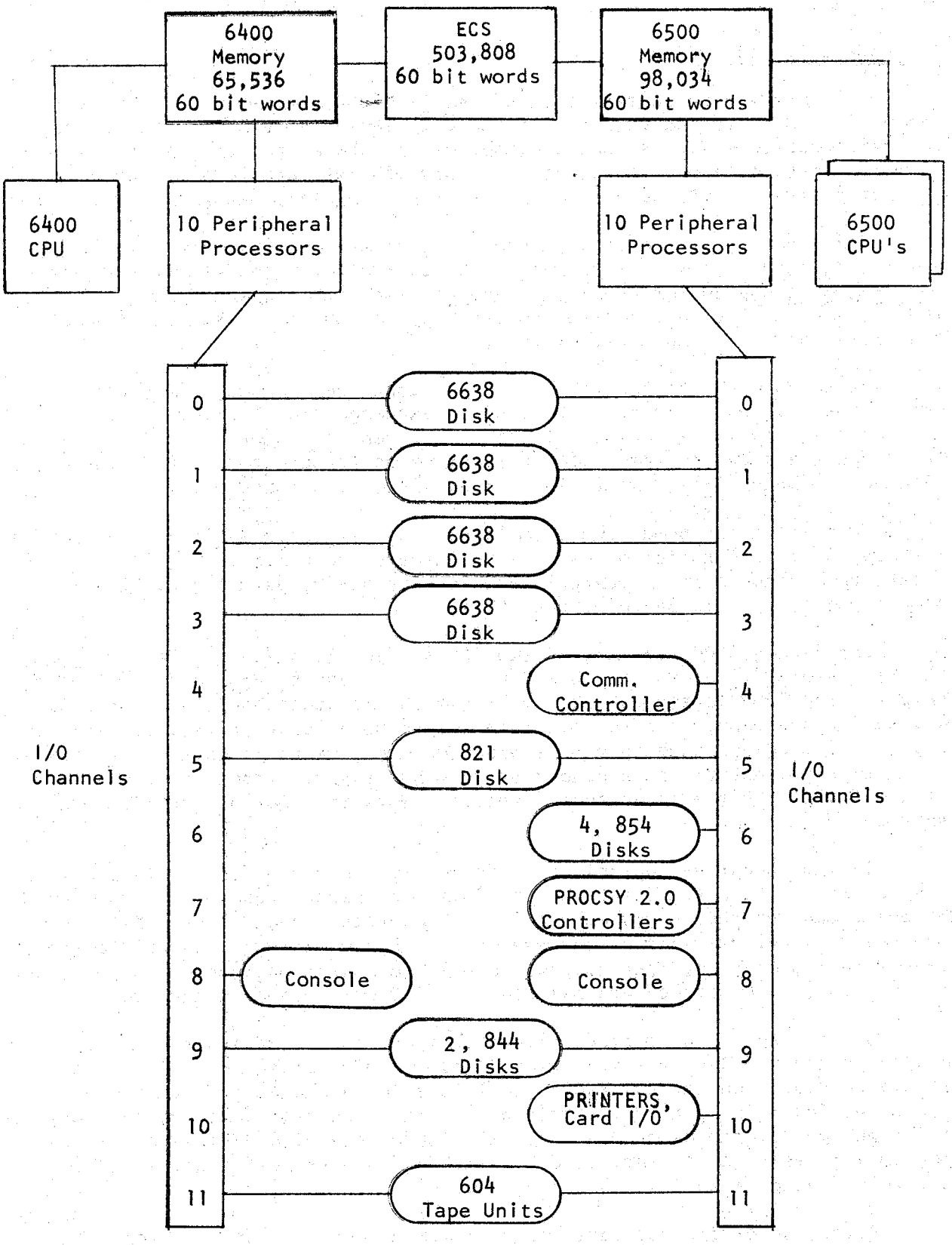

Figure 1, Purdue Hardware Configuration

Rapid access to CPUMTR is provided through a hardware exchange jump feature, available to both the CPU's (XJ) and the PPU's (MXN). Both operations effect the exchange of all CPU control registers between the CPU and a sixteen word memory area in two microseconds. Thus CPU and PPU and CPUMTR can return control to the interrupted CPU process with an exchange when the request has been processed.

The sixteen words exchanged between the CPU and memory contain a number of control and general purpose registers. These include sixteen, eighteen bit registers (A0 through A7, B0 through B7) for program execution, addressing and indexing; and eight, sixty bit registers (X0 through X7) for program execution, arithmetic and logical operations. The seven control registers include the following:

Register	Usage
P	The current (next) instruction address, relative to RA.
RA	The current program reference address. This value is automatically added to all relative address values - i.e., P, and A1 through A7.
FL	The relative address limit on field length. This value is an automatic limit on P and A1 through A7.
EM	Exit mode value - an address and arithmetic error selection code.
RAX FLX	ECS reference and limit addresses.
MA	The "monitor address" register (see Chapter 4).

The remaining system resources - PPU's, CPU time, and memory - are shared among system and user jobs through a pseudo-machine mechanism called a control point. A control point is a pivotal area, occupying 128 words of central memory, through which task execution is controlled, and to which job execution resources are allocated. The control point may be thought of as the control element of an individual computer, and the entire set of control points as a division of the hardware machine into a number of separate machines, each of which can execute an independent job.

The control point jobs, which may include such system tasks as job stream input/output (printers, card readers, etc.), or terminal subsystems, as well as user tasks (compilations, program executions, etc.) obtain resources for execution via exchange operations (CPUMTR), resource request notices (MTR) and through the efforts of a number of job and job sequencing programs which share the use of the PPU's.

These programs include ones for job scheduling, initiation, swapping,

and termination, as well as a wide variety of input-output drivers. Each PPU program, in turn, makes use of the resource access facilities of CPUMTR and MTR to perform its specific operations.

Excluding system jobs, generally all jobs which execute at control points may be interrupted, the job control information (control point area and program execution central memory) copied to a mass storage (disk) file, and re-started at a later time, possibly at another control point, or even in another machine. In the Dual MACE system this process is called auto-roll.

Auto-roll significantly enhances the performance of the system, by permitting rapid interruption of job execution in order to enable the execution of a higher priority job. Moreover, it also permits the transferral of jobs between systems by the passage of the auto-roll file pointers between machines. Since a large number of resources are shared between machines, and since system architecture and even system programs are the same, a rolled-out job can pass between machines with no visible effect on job execution.

In particular, this structure permits a job to execute on a given machine until it requests a resource not on that machine (e.g. a paper tape reader), and then to be transferred to the other machine for further execution. As another example, since the 6500 has more central memory than the 6400 (98,304 words versus 65,536), a job request for additional central memory can cause an inter-machine job transfer.

Dead-Start

The operating system programs of the Dual MACE system are initially loaded with a process called dead-start. The entire process begins with the execution of a boot-strap sequence from a hardware dead-start panel, which in turn effects the loading of sufficient additional dead-start programs from the dead-start tape to complete system loading. These dead-start program load system program libraries, recover permanent file directories and system logs, construct central memory and ECS tables, and, in selected cases, recover job files after system failure, provide post-mortem dumps, etc.

Dead-Start Panel

The dead-start panel is a set of 144 toggle switches which are used to define twelve, twelve bit PPU instructions. When the dead-start switch or button is activated, these twelve instructions are entered into the memory of peripheral processor zero in locations one through twelve. At the same time, the remaining PPU's are assigned to their respective, activated I/O channels (PPU one to channel one, PPU two to channel two, etc.) and set executing a 4096 word channel input operation to PPU word zero.

PPU zero is then allowed to begin execution at location one with the first instruction of the dead-start panel. The remaining PPU's wait for input on their respective channels. When they have received input, and after their respective data channels have been de-activated, the processors begin execution at the value loaded in PPU word zero, plus one.

System Bootstrap - BTS

The twelve instructions of the dead-start panel for Dual MACE contain a small program to rewind a magnetic tape and load the first record of the tape into PPU zero memory starting at word 7705_8. This first record contains the program BTS. It is designed to load a small amount of code into words 7705_8, through 7777_8, and a location value into word zero. At the end of the transfer of the record from the tape, the tape unit de-activates the tape channel, and PPU zero begins executing the record loaded into its memory, starting at the word address loaded in word zero, plus one.

BTS saves the entire contents of PPU zero memory, excluding those words destroyed by the loading of the BTS tape record, by writing PPU zero memory to a predefined area in central memory (50000_8 through 51462_8). The saved PPU memory can be used by the next dead-start program (DSS), to form a system, post-mortem dump. BTS then activates the tape channel and initiates the transfer of the next tape record into PPU zero memory, again starting at word 7705_8.

Dead-Start Dump - DSS

The next tape record of the Dual MACE dead-start tape contains the dead-start dump program, DSS. As its first operation, DSS stops all CPU's. Then it requests the selection of three processing options by displaying the three options and a single character selection code on the left screen of the system console. The display takes the form

```
SELECT DUMP OPTION

T - TAPE DUMP
P - PRINTER DUMP
CR - NO DUMP*
```

Tape Dump

When the tape dump option (T) of DSS is selected, DSS will write the CPU exchange packages, copies of each PPU memory, and central memory to a magnetic tape. DSS first requests the specification of the tape unit number with the display

```
SELECT UNIT (0-7)
```

Upon entry of a single digit unit number (0-7), DSS copies the CPU exchange packages and PPU zero memory from central memory to the dump tape. It then loads each remaining PPU in turn with a small dump program, activates the PPU, receives a memory copy from the PPU, writes the memory copy to the dump tape, and resets the PPU to its dead-start condition. Finally, DSS copies all of central memory to the dump tape.

Printer Dump

When the printer dump option is selected, DSS requests the printer channel and equipment numbers with the messages

```
LP CH 00
LP EQ 00
```

When these two parameters have been entered, DSS permits the selection of further dump options with the display

```
P - DUMP PPU
C - DUMP CENTRAL MEMORY
E - DUMP EXCHANGE PACKAGES
T - TAPE DUMP
S - LOAD SYSTEM
```

*CR signifies carriage return

Entry of the single character key (P, C, E, T, or S) effects the specified operation. The PPU dump operation requests the PPU number with the display

PP NO 00

The central memory operation requests the address range with the displays

CM FROM 000000
CM TO 000000

The T entry permits a return to the tape dump option; the S entry, to system loading.

System Loading

When the tape dump operation completes, when the printer dump operation is terminated with the S selection, or if no dump operation was selected, DSS continues system loading. Two options may be specified. They are requested with the display

0 - PURDUE MACE
1 - SAVER

Selecting option 0 causes DSS to position the dead-start tape to the record containing the program PDM; option 1, to SVQ. The next record of the tape is then loaded to PPU zero memory, and given control.

SAVER - SVQ

The dead-start program SVQ can be used to reconstruct central memory from a dead-start dump tape (written by DSS) after certain cases of system failure. The tables thus reconstructed are then processed further by the dead-start recovery programs. SVQ requests the entry of the dump tape unit number with the display

SELECT DUMP TAPE

and the dump file number with the display

SELECT FILE SKIP, 0 - 7

Dual MACE Option Selection - PDM

The dead-start program, PDM, permits the selection of a set of basic system dead-start options for Dual MACE. In the same fashion as DSS and SVQ, PDM displays a single character option code, and a description of the option

Entry of the single character key (0, 1, or 3) specified operation. The PPU dump operation requests the display

PP MD DD

The central memory operation requests the address range wi

CM FROM 000000
CM TO 000000

The T entry permits a return to the tape dump portion the loading.

System Loading

When the tape dump operation completes, when the pri is terminated with the 3 selection, or if no dump operatio continues system loading. Two options may be specified, with the display

0 - OUTPUT MACE
1 - SAVE2

Selecting option 0 causes DSD to position the dayfile record containing the program FDM: option 1, to SVD. The tape is then loaded to PPU zero memory, and given control

SAVE2 - SVD

The dead-start program SVD can be used to reconstruct from a dead-start dump tape (written by DSD) after certain failure. The tables thus reconstructed are then processed start recovery programs. SVD requests the entry of the dum with the display

SELECT DUMP TAPE

and the dump file reader with the display

SELECT FILE:SKIP, 0 - 7

Dual MACE Option Selection - FDM

The dead-start program, FDM, permits the selection of system dead-start options for Dual MACE. In the same fashi FDM displays a single character option code, and a descript

selected by the code. In addition, PDM displays the value selected. All options are selected from pre-defined lists, and selected in step fashion by repeated entry of the option code. The option display has the format

```
0 - "value" TYPE-INS
D - DEADSTART TYPE "value"
C - CPU 0 IS "value"
T - TAPE UNIT IS "value"
X - ECS TABLE STATUS IS "value"
S - CMRDECK ORDINAL IS "value"
ENTER RIGHT BLANK TO DEADSTART
```

TYPE - INS

The selection code is a zero.

The default value is NO.

The one optional value is blanks. When this option is selected the dead-start program SET is instructed to display the entries of the central memory resident deck (CMRDECK) syntax definitions, and to accept additions or changes to them through console keyboard entries.

DEADSTART TYPE

The selection code is D.

The values may be

```
MINOR (default value)
TAPE
DISK
MAJOR
```

These four options specify the type of dead-start processing to be performed by the remaining dead-start programs. A TAPE dead-start is a full load; a MINOR dead-start is a basic recovery; a MAJOR dead-start is a more elaborate recovery; and a DISK dead-start is an initial dead-start using a system label file. See the section on dead-start recovery.

CPU 0 IS ON or OFF

The selection code is C.

The value may be ON or OFF to reflect the status of CPU zero of a two CPU system (e.g., the 6500).

T - TAPE UNIT IS (0 - 7)

The selection code is T.

The value may be an integer from zero through seven. It defaults to the value found in bits zero through two of word three of the dead-start panel program. This option permits the use of an alternate dead-start tape.

X - ECS TABLE STATUS IS LOAD, USE OR NONE

The selection code is X.

The value may be LOAD to load the Dual MACE, ECS tables, or USE (default value) to use previously LOADed ECS tables. In general, the ECS tables are LOADed when a TAPE dead-start is performed and USEed on all subsequent dead-starts. A third option, NONE, specifies no ECS table processing.

S - CMRDECK ORDINAL IS (0 - 3)

The selection code is S.

The value may be an integer from zero through three. It specifies which central memory resident deck (CMRDECK) of syntax declarations is to be used by the dead-start program SET. The value defaults to that found in bits six and seven of dead-start panel word three. Normally it is zero for the 6400 and one for the 6500.

ENTER RIGHT BLANK TO DEADSTART

When all options have been selected, pressing the right most blank (unmarked) key on the top row of the console keyboard causes PDM to search the dead-start tape for the program SET, load it into PPU zero memory, and give execution control to it.

CENTRAL MEMORY RESIDENT CONSTRUCTION - SET

The dead-start program SET constructs (TAPE or DISK dead-start) or resets (MAJOR or MINOR dead-start) central memory resident tables. Its functions are controlled by the entries made to PDM, by syntax declarations in the CMRDECK, and by additions or changes to the CMRDECK which may be entered to SET through the console keyboard (provided that the "blank" TYPE INS option was selected via PDM). The central memory resident section (CMR) of the Dual System is pictured in figure 2. The system symbolic library (OPL) contains the text deck, PPCOM, which describes CMR in terms of assembler language (COMPASS) declarations. PPCOM is listed in appendix A.

Figure 2

Central Memory Resident - CMR

Address	Contents
0	Zeroes
1-177$_8$	Control Point zero-pointers, PPU communications areas, etc.
200$_8$-377$_8$	Control Point one 200$_8$-217$_8$ Exchange Area 220$_8$-267$_8$ Pointers 270$_8$-377$_8$ Control Statement Buffer
400$_8$-	Remaining Control Point Areas System, problem-mode exchange area Disk Reservation Table Pointers Dayfile Pointers Equipment Status Table (EST) File Name/Status Table (FNT/FST) PPU Exchange Areas Track Reservation Tables (TRT's) for disks Equipment Reservation Table (ERT) PPU Program Library Directory (PLD) System, CPU Program Library Directory (SLD) Dayfile Buffers Inter-Machine Communication Buffers CPU Monitor Code Installation Area I/O Stacks MESA Library Directory (MLD) Special TRT's Resident PPU Library (RPL)
45000$_8$ (approximately) " " "	User Program Space
177777$_8$ (6400) 277777$_8$ (6500)	

All of these areas are described in detail in Chapter 6

Central memory is preset by the dead-start program SET according to declarations found in a record of the dead-start tape, called CMRDECK. These declarations specify table sizes (e.g., the length of the file name/status table), the number of control points, the number and types of equipments with their hardware parameters (connect codes, channels, etc.). The entries of the CMRDECK may be changed or augmented by console keyboard type-ins to SET. In addition, SET will display a deck containing instructions for forming type-ins, if requested, by the entry of the "+" character.

Depending on the dead-start type option (TAPE, MINOR, MAJOR or DISK) declared to PDM, SET will either clear (zero) or leave intact the table areas of CMR. In all cases, SET reconstructs the basic pointers of control point zero.

SET performs additional tasks related to CMR construction. It determines central memory size and CPU configuration. It loads and relocates the code of CPU monitor. Finally, it loads the ECS set-up program, SXC from the dead-start tape into PPU one, waits for SXC to complete, then loads the dead-start tape loader program, STL, into PPU one, and gives it control. SET then returns PPU zero to a dead-start condition (4096 word input to location zero on channel zero).

Extended Core Storage Set-up - SXC

The dead-start program SXC is responsible for Dual System ECS set-up. It operates in one of the three modes, NONE, LOAD, or USE, declared to the program PDM. When the NONE option has been selected, SXC returns immediately to SET.

When the LOAD option has been selected, SXC constructs Dual System pointers in ECS, copies selected tables from CMR to ECS and relocates those ECS areas local to each machine. The constructed pointers reference a shared equipment status table (EST), the inter-machine communication area, shared PPU and CPU program libraries, a shared equipment reservation table (ERT) and shared track reservation tables (TRT's). SXC loads the EST, the ERT and TRT's from central memory resident. Under the USE option, SXC reads the Dual System ECS pointers, verifies their consistency with CMR, and then copies the EST, ERT and TRT tables to central memory resident. Figure 3 maps the usage of ECS.

In both modes of processing, SXC performs extensive checks on the validity and consistency of shared declarations. SXC uses the system console to display messages indicating the discovery of errors during this process. When SXC completes, it signals that fact to SET through a central memory communication cell, and returns PPU one to its original dead-start condition (4096 word input to location zero on channel one).

The system symbolic library (OPL) contains a common deck, COMSECS, which describes ECS usage in terms of assembler language (COMPASS) declarations.

Figure 3

ECS Map

Adress		Contents
0 - L		PPU Program Library
L - L+M-1		Inter-machine Space
	L - L+77_8	Pointers
	L+100_8 -	Inter-machine Communication Buffers
		Equipment Reservation Table
		Equipment Status Tables
		MESA Library Directory
		PPU Library Directory
		System CPU Program Library Directory
		Track Reservation Tables
L+M -		Machine one, PPU edit space
		Machine one, MESA swap space
		Machine two, PPU edit space
		Machine two, MESA swap space (optional)

System Tape Loading - STL

When SXC completes, and signals that fact to SET, SET loads the next record of the dead-start tape into PPU one. That record contains the system tape loader program, STL. STL not only loads the system program libraries from the dead-start tape, it also loads the remaining run-time system executives (PPU monitor, PPU resident, and the dynamic display driver, DSD), and coordinates the remainder of the dead-start process, including permanent file, dayfile (log) and recovery processing.

For the performance of all of these tasks, STL loads a number of special programs from the dead-start tape. In order, these include:

1. Control-ware (micro-program) for the 844 disk unit controller.

2. Peripheral processor resident (PPR). This is the run-time interface between PPU programs and the system.

3. The installation defined program, DSI, is loaded in PPU two. It performs permanent file and dayfile (log) processing.

4. If the MINOR, MAJOR, or DISK dead-start options were declared, the job recovery program, REC, is loaded in PPU three. If the TAPE option was declared the mass storage loader program, MSL, is loaded in PPU three.

5. The date and time program, DTE, is loaded in PPU four. DTE accepts the current date and time of day from the console keyboard, verifies their correctness, and stores them in CMR.

6. The dynamic display driver program, DSD, is loaded in PPU nine. DSD provides the run time display and console entry facilities. [8]

7. Peripheral processor monitor, MTR, is loaded in PPU zero.

STL transmits the various PPU programs to the processors on the respective data channels to which the PPU's are assigned. The unassigned PPU's (five, six, seven and eight) are loaded with code for entry to the idle loop of PPU resident.

STL then loads the first two segments of the PPU program library from the dead-start tape. The first segment contains PPU programs resident in central memory (CM), the resident PPU library (RPL). These are programs whose execution mode or frequency requires the rapid and efficient access provided by CM residence. The second segment contains programs which can reside in ECS, if available, but must reside in CM if ECS is not available.

Some programs of these two libraries are used in subsequent dead-start operations.

After these two library segments have been loaded, STL activates all PPU's by de-activating their data channels. The next processing of dead-start is performed by DSI. If a TAPE dead-start is in progress, STL starts DSI. If a MINOR, MAJOR or DISK dead-start is in progress, REC starts DSI.

Permanent File and Dayfile Processing - DSI

DSI receives a "go" signal from STL or REC in its central memory communication area. DSI performs two major tasks - the processing of permanent files and dayfiles (logs).

DSI can be directed to recover or initialize the permanent file sub-system of Dual MACE. Recovery includes accessing each device permanent file directory, as declared in the CMRDECK entries, verifying its accuracy, and copying the indicated track reservations to the CMR and ECS track reservation tables. When DSI is instructed to initialize permanent file directories, it records new directory labels, and clears their track reservation tables.

Whether recovering or initializing permanent file directories, DSI also constructs a master permanent file directory in a disk area declared via a CMRDECK entry. This directory contains all permanent file references declared in all device directories.

DSI can also be instructed to recover or initialize the system dayfiles. These dayfiles include a master dayfile (SYSTMDF) which contains all dayfile entries; an accounting information dayfile (PDUFILE); an equipment error dayfile (ERRLOG); and a statistics information dayfile (STATDF). Usually the master dayfile is activated only for purposes of performance measurement.

A CMRDECK entry declares the location (disk equipment and first track) and the recovery/initialize status for each dayfile. When recovering a dayfile, DSI spaces to the end of the dayfile, insures that all tracks are reserved in CMR and ECS, and updates the dayfile position pointers in CMR. When initializing a dayfile, DSI terminates the file with an initial end-of-information (EOI) sector, drops all reserved tracks except the first one, and sets the CMR dayfile pointers to indicate that the dayfile is empty.

Separate dayfiles are recorded on each of the two, Dual System machines. CMRDECK entries declare their separate locations. The dayfiles are not merged until processed for accounting or performance measurement.

When DSI finishes, it updates its communication area status. The next dead-start activity depends upon the dead-start type - TAPE, or recovery (MINOR, MAJOR, or DISK).

TAPE Dead-start - STL

If a TAPE dead-start has been selected, DSI returns control to STL. STL then activates the mass storage loader, MSL, through central memory communications pointers. Using these pointers, STL communicates the length and CM location of each library program to be copied to disk by MSL from central memory. With this structure, STL can overlap the reading of library records from the dead-start tape with their copying to disk by buffering them to CM for disk copying by MSL.

If the ECS tables are being LOADed, or if no ECS tables are in use, the remainder of dead-start consists of this copy operation. STL copies the remaining PPU library programs and all CPU programs to CM, and MSL copies them to disk. STL reserves disk space for the copy, in such a way that the libraries begin at the end of an area reserved for the system label on disk equipment zero. The label begins at sector zero of track zero of disk equipment zero. STL also constructs the directories.

When all programs have been copied from tape to disk, MSL writes the system label file in the area reserved for it by STL. This label file contains a copy of central memory resident, beginning with the first file name table entry and ending with the last CPU program library directory entry. The label file is referenced by the first file name table entry, named SYSTEM. Various portions of the label file are used by the recovery processor, REC, when a MINOR, MAJOR, or disk dead-start is selected.

Each machine of the Dual System has a separate label file. The label of the first machine TAPE dead-started begins at sector zero on track zero of disk equipment zero. The label of the second machine begins at the next free track on disk zero. Its location is recorded in the first machine label by MSL.

ECS Library Directory Construction - STL

At the end of the program library copy, STL copies the PPU program, CPU program and MESA program library directories to ECS, provided that the ECS LOAD mode was selected, and that the system device, disk equipment zero, has been declared a shared device. These ECS directories can be used by STL in the dead-start of the second machine of the Dual System to bypass all of the tape to disk library copy.

Using ECS Library Directories - STL

STL uses these ECS library directories when the TAPE and ECS table USE options are selected, and when the system device, disk equipment zero is a shared device. In this case, STL reads the directories from ECS and inserts their entries in central memory. The CPU program and MESA program directories are copied directly. Only selective portions of the PPU program directory are copied - those referencing ECS and disk resident programs.

MINOR, MAJOR and DISK Dead-start - REC

When a MINOR, MAJOR, or DISK dead-start has been selected, STL activates the recovery program REC. Using selected portions of the system label file and central memory, REC recovers library directories, track reservation tables, file entries, and jobs in execution. The extent of the recovery is specified by the dead-start type.

A MINOR recovery is the simplest level recovery. It assumes that all of CMR is intact and includes the recovery of the library directories from the label file, and the return of executing jobs from their assigned control points to the input queue, from which they can restart execution. A MAJOR recovery, at the next level, includes all the steps of a MINOR recovery. In addition, no assumptions are made concerning the track reservation tables. They are reloaded from the system label file, thus resetting them to their TAPE dead-start condition. Then REC reads all disk files recorded in the file name table and recovers the track linkages as recorded on the files. This includes reading files indexed within files, such as rolled-out jobs, and recovering permanent file accesses.

A DISK dead-start is the equivalent of a TAPE dead-start, using the system label file rather than the dead-start tape. In this dead-start, REC copies the entire label file to central memory, thus restoring it to its TAPE dead-start condition from the file name table through the CPU program library directory. All activity since the TAPE dead-start is thus erased.

REC locates the system label file by referencing the first label file sector written on sector zero of track zero of disk equipment zero. REC verifies the presence of the label, halting the dead-start if an error is detected. From the contents of a valid label REC can detect which label is to be used, and the location, if necessary, of the correct, second label. The label file and central memory resident are constructed in such a way that REC can use the dead-start type to calculate an increasingly larger skip address at which to begin copying the system label file to central memory.

Dead-start Type	Skip Address
DISK	none
MAJOR	First track reservation table
MINOR	PPU library directory

Final Dead-start Activities - STL

When the library loading or recovery processes have been completed, STL terminates the dead-start process by issuing several dead-start messages to the system dayfiles, and by activating a number of repetitive PPU programs.

These include

1IM - inter-machine stack processor
1RS - the system, repetitive services processor
1SM - the terminal system status monitor

Each of these programs execute for short, repeated periods, using any of the available pool PPU's (one through eight). Their recall time is preset, and the recall cycle is controlled by a PPU delay stack mechanism of CPU monitor.

Special Dead-Start - Checkpoint and Restore

A very special dead-start facility of the Dual MACE System permits the suspension of system activity in one or both machines and later, full resumption. In the intervening period, all system hardware, except the system disk devices, can be used for other processing. In particular production activity can be suspended (checkpoint) for emergency maintenance or for system development, and then resumed (restore) at a later time. The removable disk devices (e.g. 844 and 854 disk packs) and alternate CMRDECKs facilitate hardware usage during checkpoint.

The actual operations of checkpoint and restore are a blend of system and dead-start processing. Both functions are performed by the system checkpoint/restore PPU program, 1CK. 1CK is activated after dead-start via an entry at the console keyboard.

When called to checkpoint, 1CK suspends all control point activity (rollout or termination) and copies central memory resident tables and selected ECS tables to a disk file. Pointers to that disk file are stored as a special FNT/FST entry in the system label file.

A checkpointed system restoration is started with a disk dead-start. This brings the checkpoint file FNT/FST entry into central memory. When 1CK is activated by the appropriate console entry, it reads the checkpoint file and restores central memory and ECS from the contents of the file. Provided that none of the disk devices of the checkpointed system have been disturbed, system processing can resume without further delay.

Both machines of the Dual MACE System may be checkpointed and restored. The operations must be performed in the same order both times - i.e., the last machine checkpointed is the last machine restored. It is also possible to checkpoint one machine, provided that the other does not disturb the disk files of the checkpointed system. Generally, when only one machine is checkpointed, the other is first emptied of all jobs.

A checkpoint operation can also be automatically initiated by an element of the Purdue, power-down-sequencer. This hardware system is an emergency, sequenced, power-off mechanism. Among its operations is the sending of a special signal to the 6400 and 6500 CPU's. This signal raises a status

bit on the system console channel. The system display driver, DSD, activates ICK when it senses the signal. ICK in machine two ignores the signal. ICK in machine one performs a checkpoint and then sends an inter-machine function to machine two which activates ICK for a properly sequenced checkpoint. After an appropriate delay for the completion of the checkpoints, the power-down-sequencer cuts power to the 6400 and 6500 CPU's, as well as all peripheral devices, in the proper order.

bit on the system console channel. The system display driver, DSD, activates ICK when it senses the signal. ICK in machine two ignores the signal. ICK in machine one performs a checkpoint and then sends an inter-machine function to machine two which activates ICK for a properly sequenced checkpoint. After an appropriate delay for the completion of the checkpoints, the power-down-sequencer cuts power to the 6400 and 6500 CPU's, as well as all peripheral devices, in the proper order.

System Intra-Communication

After the dead-start has been completed, the PPU monitor (MTR), the CPU monitor (CPUMTR) and the PPU resident (PPR) combine to provide system executive functions. In their combined operation they communicate over a structured I/O path which uses the hardware exchange jump features of the 6000 and a set of standard communication cells.

The communication cells consist of eight words of central memory for each peripheral processor. The ten sets of eight word areas begin at location 50_8 and end at location 167_8. MTR uses location 50_8 through 57_8; DSD, words 160_8 through 167_8.

Peripheral Processor Resident - PPR

From the viewpoint of the PPU executive routine (PPR) the eight words consist of an input register (word 0); an output register (word 1); and six message buffer words (2 through 7). PPR receives task assignments in its input register, in the form of a PPU program name and associated parameters:

Bits	Contents
59-42	Program name
40-36	Control Point number
35- 0	Routine dependent parameters

PPR makes requests of the other two system executive processors, MTR and CPUMTR, through its output register:

Bits	Contents
59-57	CPUMTR Flags
56-48	Function Number
47- 0	Function dependent parameters

The Function Number indicates the specific request (memory request, equipment release, etc.). The 48 parameter bits in the output register may be augmented with the 360 bits of the six message buffer words for the transmittal of function dependent data (e.g., character format messages). The CPUMTR flags are set by CPUMTR to indicate various, non-standard responses to the function request.

In a standard function request, the first twelve bits of the output register are cleared by MTR or CPUMTR when the requested operation has been performed. The output register parameter bits and the message buffer words may be used by MTR or CPUMTR to return function response information.

PPR Idle Loop

When a pool PPU (normally PPU's one through eight are called "pool" PPU's) is activated by STL, after the execution of a small preset routine its execution control passes to a resident location called PPR. A small input register scan loop begins at that location. Its basic function is to respond to task assignments placed in its input register by CPUMTR, which makes the assignment by simply storing a word of the proper format in the input register cell.

Every 128 microseconds the PPR scan code samples the contents of bits 48-59 of its input register. (The sampling rate is a compromise between response time and memory bus load.) When those bits become non-zero, PPR begins the processing of a task assignment. This processing involves the use of other, resident routines, and entries from the set of PP memory cells in the range (0-77_8). Since these cells can be directly referenced in a twelve bit (one word) instruction, they are usually called "direct cells".

Direct Cells

The economy of reference of direct cells makes their efficient usage important to the effectiveness of peripheral processor code. In order to standardize their usage, the direct cells in each pool PP are structured into several categories: utility cells; standard variable cells; and constant cells. Moreover, coding standards have been established for naming the direct cells with two character names, and a set of names has been reserved for all three classes of direct cells.

Direct Cell Constants

There are seven direct cell constant locations. They are set at dead-start and must never be destroyed by any PPU program.

Location	Name	Contents
70	ON	1
71	TR	3
72	HN	100_8
73	TH	1000_8
75	IA	The input register word address
76	OA	The output register word address
77	MA	The message-buffer word address

Direct Cell Variables

There are two sets of direct cell variables: those loaded by PPR when it begins the processing of an assigned task; others associated with various, standard tasks (e.g., file processing, mass storage processing). The variables set by PPR include:

Location	Name	Contents
50-54	IR-IR+4	Input Register
74	CP*	The assigned control point address
55	RA	The control point RA/100_8
56	FL	The control point FL/100_8

Mass Storage Variables

When mass storage processing is taking place the following four locations contain mass storage variables

Location	Name	Contents
4	T4	The Channel
5	T5	The equipment ordinal
6	T6	Current logical track
7	T7	Current logical sector

Other, miscellaneous direct cell assignments will be mentioned in discussions of their associated tasks.

Program Loading

During the input register scan cycle of PPR, the contents of the input register word in central memory (CM), located at (IA), is transferred to direct cells 50_8 through 54_8 every 128 microseconds. PPR recognizes a non-zero value in 50_8 as a task assignment and uses the program loading facilities of PPR, including MTR/CPUMTR communication and, possibly, mass storage processing.

Program Names

As mentioned, the PPU program name occupies 18 bits of the input register word. Thus it consists of three, six-bit display code** characters. The characters of the name are selected to indicate the load address, usage, and accessability of the package.

---

*Note that all control point addresses must be less than 7601_8.
**Display code is the CDC binary to alphanumeric code.

Programs whose names begin with a character in the range (A,Z) may be accessed by central and peripheral processor routines. Programs whose names begin with a number (0,9) can be accessed only by MTR, CPUMTR, or another PPU program. The number indicates the load address characteristics of the routine. Appendix F lists all PPU programs.

Character	Load Address Characteristic
0	Location Free or Zero level - loaded and executed relative to the contents of direct cell 15_8 (LA).
1	Primary level - routine loads at 1100_8 (PPFW).
2	Secondary level - routine usually loads at 2000_8.
3	Tertiary level - routine usually loads at 3000_8.
4,5	Miscellaneous overlays.
6	Reserved for mass storage drivers and stack processor routines which load at 556_8.
7	Reserved for error processors which normally load at 7503_8.
8	Not normally used.
9	Reserved for DSD overlays which may have fixed or variable load addresses.

Whatever the program name, the process of loading it is basically one of locating the routine and loading it from its storage medium into its assigned PPU memory locations. To perform these functions, the PPR input scan loop uses the subroutine PLL.

The Peripheral Library Loader - PLL

PLL is the subroutine in PPR which locates and loads all PPU programs. Its basic input parameter is the program name, which is passed, left justified, in the 24 bits of direct cells 13_8 and 14_8 (CM+3 and CM+4). An optional parameter, for location free (Zero-level) programs is the load address which is passed in direct cell 15_8 (LA).

PLL locates the residence of a program by searching the peripheral library directory constructed by STL. For reasons of efficiency, PLL requests that CPUMTR perform the search, and transmit to PLL the program location.

In the first place, the PLD is located in CM, for common access to all PPU's. Thus the directory can be accessed more effectively by the CPU. Secondly, the directory is large, and therefore it is sorted by program name to make use of a binary search. Finally, PPU programs may reside in CM, on a mass storage device (disk) or in ECS. ECS routines must be moved to CM before a PPU can load them.

The PLD entries have the form:

Bits	Contents
17-0	The program name
59	If one, the program is in CM and: bits 18-35 = CM address bits 36-47 = length in CM words bits 48-58 = PP load address (0 if location-free)
54-59	If 00_8 the program is on the disk and: bits 18-29 = sector 30-41 = track 42-53 = load address
54-59	if 01_8, the program is in ECS, and: bits 18-41 = ECS address 42-53 = ECS word length (note: the PPU program header in ECS contains the load address in byte 2 of word 1).

Searching the PLD

PLL requests a PLD search of CPUMTR by placing a request in its output register and starting CPUMTR (see the description of the FTN routine for more detail on this process). PLL passes the program name in parameter bits 6-23 of the output register.

When CPUMTR has recognized the request it performs a binary search of the sorted directory. The search can have several results:

1. The program name cannot be found in the PLD. In this case, PLL issues an ABTM function, and returns to PPR. ABTM terminates the task, thus clearing the input register.

2. The program is in CM (bit 59 of the PLD entry = 1). CPUMTR returns the CM address, CM word length and PP load address in the output register (clearing bits 48-59, of course).

3. The program is on the disk. CPUMTR returns the track, sector and load address.

4. The program is in ECS. If CPUMTR has CM buffer space, it transfers the program to CM and returns parameters as in case (2). If buffer space is not available, CPUMTR requests a function repeat. (See the CPUMTR section for more detail on function repetition.)

The CPUMTR response parameters effect a return to PPR (case 1); a loading of the program to PP memory (cases 2 and 4) and return to the caller; or a disk load operation (case 3).

When the program location is disk, PLL makes use of the disk driver facilities of PP resident. While these facilities are described in more detail in another section, it is important to note two constraints: (1) accessing the disk driver may cause a call to PLL, thus PLL is singly re-entrant; (2) the disk driver must not require a disk driver to be loaded from the disk, thus the driver must be CM or ECS resident.

Regardless of these factors, the net effect of disk load equals those of the CM and ECS load with one important exception: since the package is transferred in disk sectors of 502_8 PP words, of which 500_8 are PP instructions (the other 2 words are disk control bytes), and since the load is terminated by a sector containing 473_8 PP words or less, the last word address of a PPU program transferred from MS cannot exceed 7765_8 ($12_8 \times 500_8 + 473_8 + 1073_8 - 1$).* After completing the load, PLL returns to the caller with the package load address in LA.

PP Task Initiation

If the call to PLL from PPR to load a program is successful, PLL updates the direct cell variables CP, RA and FL, and enters the package at (LA) + 5. The cell CP is set to 200_8 times the control point number in the input register word. RA and FL are set via a call on the resident routine PRL (see the description of PRL), which sets those two cells by reading the control point status word.

Once the program gains control it may make use of any or all of the routines already mentioned - PLL, FTN, PRL, the disk driver, as well as several other useful routines. When the package has completed its task, it notifies CPUMTR via a function call to FTN and returns to PPR.

Occasionally a program will pass control directly to a different program in the same PPU. It does so by writing a new input register value and transferring control to PPR, where the PPU load operation can resume. This method of calling a package is effective in that it shortens system overhead and permits the passage of many parameters in the direct cells which are not disturbed by the loading process (only cells $0\text{-}17_8$, CP, RA, FL and IR-IR+4 are disturbed). However, it does bypass CPUMTR control and should not be used to call packages which are to be regulated by CPUMTR, such as job scheduler, job advancement, etc. (See PPU Control.)

PPR Routines

There are eight resident routines in PPR which may be used by PP packages: FTN, PRL, RCH, DCH, DFM, EXR, PLL and SMS. In addition there are three routines of constant address available for access in whichever disk driver might be in use: POS, WDS, RDS. Entry to all routines is effected via the PPU, RJM instruction.

---

*1073_8 is the normal PPU program load address.

Function Request - FTN

The most commonly used routine in PPR is FTN, the routine by which function requests are transmitted to MTR and CPUMTR. The applicable parameters are passed in direct cells 11_8-14_8 (CM+1 - CM+4); the function number in the PPU A register. Additional parameters may be passed through the CM locations of the PPU message buffer ((MA) - (MA) + 5). Function names and codes are listed in appendices A,B, and C.

FTN completes the output register word in (CM - CM+4) by storing A in CM, and then transfers the word to the central memory communication area of the PPU. If the function requested is for MTR processing, FTN samples the output register word every 128 micro-seconds, and returns to the caller when bits 48-59 become zero. Note, then, that a function number equal to zero is a null operation.

If the function is to be processed by CPUMTR, FTN sets B0 of the PPU's own exchange area (whose address is pre-set at dead-start) non-zero and exchanges a CPU via an MXN instruction until B0 becomes zero.

FTN then waits for bits 48-59 of the output register to clear. Occasionally, CPUMTR instead of clearing the bits, will set a repeat request in bit 2^{58}. The scan code senses this request, uses bit 2^{57} as a CPU number, resets the output register word, repeats the MXN, and resumes the output register scan.

The FTN routine makes use of the constant cell, ON, and by virtue of the hardware operation of the CRM instruction, destroys the direct cell 0 (TO). When FTN exits, the A register is zero, 10_8 (CM) is zero, and 11_8-14_8 (CM+1 - CM+4) contain the MTR or CPUMTR reply values.

Pause for Relocation - PRL

The PRL routine of PPR is a storage move wait interface for the peripheral processor. Any program which enters this routine must be prepared to wait an indefinite length of time for a possible central memory storage move. As such, then, it is poor practice for a program to enter PRL with some system resource reserved (e.g. a channel, or equipment).

PRL uses the direct cells CM-CM+4. At exit they contain the status word (STSW) of the control point to which the PPU is assigned. The direct cells RA and FL are updated and the A register is set to RA.

PRL first reads the control point status word to CM-CM+4. This word contains a move-storage bit (bit 34). If this bit is clear, PRL sets FL and RA and returns.

If the move bit is set, PRL issues a PRLM function, via FTN. This function notifies PPU monitor that the PPU can accept a storage move. When FTN returns, the move is complete. PRL re-reads STSW and re-checks the move bit, looping through the PRLM-read cycle until the move bit clears.

Reserve Channel - RCH

RCH reserves a system channel by issuing an RCHM function request via FTN. On entry the A register must contain the channel number, which is stored in CM+1 before entry to FTN.

Drop Channel - DCH

DCH releases a reserved channel by issuing a DCHM function request via FTN. On entry the channel number must be in the A register, and it is stored in CM+1 before entry to FTN.

Some dual interface disk drivers may modify code in PPU resident to link the DCH subroutine to themselves in order to release a unit reservation. The DCH subroutine is restored after the unit has been released. It is then important that any routine which uses the disk drivers drop the channel reservation through DCH.

Issue Dayfile Message - DFM

The DFM routine of PPR is used for the transmittal of messages to the system, job or special dayfiles. The PPU can, of course, transfer messages directly to the control point console lines (MS1W and MS2W) without monitor assistance with the central write, PPU instructions.

On entry to DFM, bits 0-11 of the A register must contain the address of the message characters in PPU memory, and bits 12-17 must contain the message routing code for CPUMTR. DFM stores the routing code in CM+1, and uses the message address to transfer the message to the PPU-CM message buffer to a limit of four words or to a zero byte terminator, whichever comes first.

If the first word of the message is zero, it signifies an indirect message. In that case, the PPU must have already stored the message in central memory.

The address supplied to DFM references a CM control word (five PPU words) of the form:

59-48	Zero
47-30	CM message address
29- 0	Unused

DFM merely transmits the control word to its message buffer. Then DFM calls the PPR subroutine DFD.

DFD is responsible for issuing the DFMM function to CPUMTR via the FTN routine, and for processing the CPUMTR response. Normally, CPUMTR will return a "message processed" response, in which case DFD returns to DFM, and DFM exits if the message has been fully processed or re-cycles for messages longer than four words.

However, when CPUMTR senses that the CM dayfile buffer being referenced is full, it returns a "dump-buffer" response. When DFD recognizes this response, it transfers a portion of the PPU to CM (the length and CM address are specified in the response), and executes the program 1DD, via the routine EXR.

The program 1DD transfers the dayfile buffer from CM to the disk file containing previous entries for the dayfile. When 1DD has completed, it restores the dumped PPU memory, in which it executes, from the CM dump area, and returns to DFD. DFD then exits to DFM, and DFM returns to its caller.

Note that the dump process, while it involves dumping PPU memory, loading 1DD, and re-loading PPU memory, is essentially transparent to the DFM caller. The main caution to the DFM user is the same one that applies to the PRL caller - calls should not be made when system resources are reserved.

Whatever the final DFM path, the direct cells T0, T1, T2 and CM - CM+4 are destroyed. All other direct cells are preserved, even when a 1DD operation occurs.

Execute Routine - EXR

The routine, EXR, which is used by DFD to execute 1DD, is a general purpose, routine execution interface in PPR. The user sets the A register to the three character program name, sets LA if the routine is location free, and executes EXR.

EXR loads the routine via PLL, and passes control to the loaded routine. By transferring its exit address to the exit instruction of the loaded routine, EXR also makes it possible for the loaded routine to use EXR itself.

Peripheral Library Loader - PLL

PLL has already been described in the section dealing with the initial loading of a PP task from the PPR idle loop. As has been noted, PLL is also used by EXR, and may also be used by routines which wish to load another program and execute it repeatedly.

Set-up Mass Storage Driver - SMS

SMS is the disk driver preparation interface of PPR. It uses the standard set of disk direct cells, T4 and T5, already described, in loading the driver.

On entry, only T5 need be set (to the equipment ordinal). From the equipment status table (EST) entry for that ordinal, SMS determines the driver type. It then checks the current resident driver identification at location MSD against the requested driver. If the two match, SMS sets T4 to the channel to which the equipment is connected, executes the driver preset routine, and returns to the caller.

If the required driver is not resident, SMS forms the driver program name by prefacing the EST equipment type with a 6, and executes PLL to load the driver. Note again: the driver cannot be disk resident, otherwise the conflicting situation will occur in which one disk driver may be needed to load a different one to the same locations.

When PLL returns to SMS, the driver identification is re-matched with EST entry for the equipment, T4 is set to the channel, SMS executes the driver channel preset routine, and returns to the caller.

Mass Storage Processing

Once the proper driver has been loaded in the disk driver area (cells 556_8 through 1071_8) of PPR, a PPU program can reference four disk driver labels for disk file processing. The cells SLM and SLM+1 contain the sector limits for the inner and outer zones of the device (only the 6603 has differing limits for inner/outer zones). The three labels, POS, WDS, and RDS are RJM entry points for positioning, writing, and reading a sector of a disk file. Entry to all three labels presumes that the channel referenced in T4 has been reserved (by a call to RCH, or a positive response, CCHM call to FTN).

Positioning - POS

After the disc channel is reserved the positioner of the device must be moved to the proper spot. While on some devices (821, 844 or 854) the act of positioning is simultaneous with reading/writing, on others (808) it is a

separate operation. Thus, for uniformity and device independence, all disk file processors use POS at the beginning of file processing, and every time the track number (in T6) changes.

Write Sector - WDS

The entry WDS is used for writing a single sector on a disk device at logical track T6, logical sector T7. The sector buffer address in PPU memory is supplied in the A register.

Read Sector - RDS

The entry RDS is used for reading a single sector from a disk device at logical track T6, logical sector T7. The buffer address in PPU memory is supplied in the A register.

Error Processing

Whenever an error is detected during the transfer of a sector to or from a disk device, the driver makes use of a 7-level error overlay to process the results of the error condition. Depending on the equipment, one or more 7-level overlays may be required to process the error. For some devices (the 821 or the 844) a 7-level overlay may also be required to preset the driver channel instructions.

Normally, for the first three errors the error processing overlay issues an error message to console line 2 of the control point and to the error log dayfile (via DFM) and returns to the driver. When the fourth error occurs, the error overlay will set the equipment error bit (bit 12 of word 26_8 (SNSW) of the control point) and will wait operator action.

If the operator types "GO", the error overlay will return to the driver, instructing it to issue the error message to the job dayfile (again via DFM). If the operator types "DROP" the error overlay will issue a DPPM function via FTN and return to the PPR idle loop. No message will be issued if the original calling program is IDD, in order to avoid dayfile lock-up.

The preceding discussion should make it clear that the usage conditions af DFM, PRL, etc. apply to disk processing. In fact, they apply to all processing as a general rule: reserve only the resources needed for as short a time as possible, and avoid parallel resource requests (including calls to PPR resident routines.)

Addressing

While the direct cells T4 through T7 provide the primary addressing control of disk file processing, two PP data words (bytes) in the disk sector itself provide the secondary level of file address linkage.

The first byte of the sector (control byte one, or CB1) contains the normal, forward file linkage address. Usually, this is simply the number of the next sector in the file. When the next sector is in a new track, bit 11 of CB1 will be set and bits 0-10 will be the track address (the sector number is zero). Thus CB1 is normally non-zero.

The second byte of the sector, CB2, is the length of the sector in CM words. Normally it is 100_8 for a full sector of 100_8 CM words (500_8 PP words). If it is less than 100_8 it indicates an end-of-record (EOR) condition. When both CB2 and CB1 are zero, they signal an end-of-information (EOI) condition.

While CB1 is normally non-zero it may be set zero to indicate an end-of-file (EOF) condition. In that case, CB2 must be non-zero, containing the forward file linkage normally found in CB1.

Thus, when a PP package is using the disk driver to read a file, it uses the CB1 and CB2 contents to update the contents of T6 and T7 as the file is processed. Some general rules are:

1. If CB1 = CB2 = 0 there are no more sectors in the file (EOI)
2. CB2 < 100_8 signals EOR.
3. If CB1 is non-zero and less than 4000_8, it is the next sector address to be stored in T7. If CB1 is greater than 4000_8, then it is the next track to be stored in T6; T7 is to be set to zero; and POS must be entered before reading the next sector.
4. If CB1 is zero, (EOF), CB2 is to be used in place of CB1 in rule 3, above.

Inter-machine Disk Conflicts

Several of the system disks (808, 821, 844) can be accessed by both machines. The reservation conflicts are resolved by routines of the drivers themselves, and, as has been mentioned, a unit reservation may be released through special linkages between the drop-channel subroutine (DCH) and the disk driver.

The contention logic of each dual-interface disk driver is essentially the same. The driver attempts to connect and reserve the unit for a moderate duration, usually less than a hundred milliseconds. If the unit cannot be reserved within that itme, the driver executes a 7-level overlay. That overlay persists until the unit is reserved, dropping the channel (DCH), pausing for relocation (PRL) and sending unit reservation delay messages at appropriate intervals. The messages take the form

"xxyy RESERVED."

where xx = equipment type code

yy = equipment number

The dual inter-face types are:

DB = 808 (6638)

DG/RD = 854 pack

DH = 821

DJ = 844 pack

The Peripheral Processor Monitor - MTR

A second system executive processor loaded at dead-start is PPU monitor (MTR). MTR is loaded in PPU-zero and remains there for the entire duration of system execution. Its title of monitor is particularly appropriate, since, in comparison to CPUMTR, its main task is to continually review the status of the system, and issue a appropriate stimuli to CPUMTR, where necessary.

In addition to this monitoring activity, MTR, also processes a small number of PP output register function requests, which, in general require long-term processing. These include storage control (2 functions), system step control (2 functions), a time/date function, and an inter-machine communication function.

Function Processing

In a main scan loop, MTR reads each of the PP output registers in turn, looking for MTR type functions and CPUMTR step-mode status. When an MTR function is recognized, the appropriate resident processor is invoked. MTR function codes are listed in Appendix B.

Step Control

Step control is an important system debugging aid. When it is set, the system is said to be "stepped", and each pool PPU function request, either to CPUMTR or MTR is delayed or stepped until specifically advanced by a keyboard, space-bar entry. Thus the testing of a system PPU routine can be advanced from one function request to the next, while the effect of its operation is monitored in central memory.

When a CPUMTR stepped function is recognized, MTR attempts to advance the processing of the function, subject to the step advance control provided by DSD. (Note: a CPUMTR, stepped function is signified by a value of 1 in bit 59 of the PPU output register word.)

Both step advance and step set are invoked through DSD commands, and by MTR functions (STPM, SMSM). When step mode is set, MTR sets control point zero word 44_8 (MSCL) non-zero to signal the step to CPUMTR. CPUMTR, in turn, recognizes the step mode in this word, and sets the step mode, output register bit in response to function requests from processors halted by the step mode. A step advance function from DSD, stimulated by a space-bar entry from the console, causes MTR to locate a processor eligible for advance, set the processor address in MSCL, and re-start the processor via an MXN instruction at the proper PP exchange address. CPUMTR then responds by processing the function, and clearing the processor address in MSCL.

When MTR recognizes a stepped CPUMTR function, and step mode is not set, it uses the same process to advance the stepped processor (e.g., after the system has been unstepped).

Storage Control

Two MTR functions are reserved for storage control: RSTM, request storage; and PRLM, pause for relocation. Both are issued via the PPR routine, FTN.

When a PPU issues a request for storage, three conditions can exist:

1. The storage is not available and MTR issues a response to that effect.
2. The storage is available and ready for use. MTR assigns it and issues a response to that effect.
3. The storage is available, but must be re-grouped or relocated.

In both cases two and three, MTR makes calls to CPUMTR to change the contents of the memory control cells of the control point, (RA and FL in STSW, RA and FL in the exchange package). In addition, case three requires MTR to calculate and execute a storage move.

A storage move can be a fairly long-term operation, depending upon the amount of storage that must be grouped to satisfy the request, and the extent to which it is scattered between control points. From a table MTR can determine the inter control point free space amounts, which, in case two provide sufficient data to satisfy the storage request.

When MTR has located the required storage, it begins relocating the memory of each affected control point, one control point at a time. The process steps are:

1. Set the control point move flag in STSW (via a CPUMTR function call).
2. Wait for all processors assigned to the control point to issue a PRLM function via a call to PRL.
3. Issue the move storage function request to CPUMTR.
4. Update the FL and RA for the moved control point and clear the move flag (again, via a CPUMTR call).

At the end of this four step cycle, MTR clears the output registers of the PPU's which were held on the PRLM function in PRL (PRL updates RA and FL and exits) and advances to the next control point to be moved. Eventually when the entire move is complete, MTR will signal that fact to the original, requesting PPU in an output register response. Note that the requesting PPU must update the RA and FL direct cells via a separate call to PRL.

System Monitoring

In addition to the function processing, MTR maintains a number of other periodic activities. These include the following:

1. Checking Absolute Address Zero
2. Central Processor Monitoring
3. PPU Delay Stack Monitoring
4. Control Point Monitoring
5. Central Memory Testing
6. CPU Program Timer interrupt Control
7. I/O Stack Monitoring
8. Inter-machine Function Transfer
9. Top-line Error Stop Monitoring
10. Clock and Date Control

Checking Absolute Address Zero

A large number of processes, including the operation of the CPU exit mode hardware, depend on the fact that the cell at CM address zero (often called absolute address zero or AAZ) contains zero. To prevent rapid system degeneration when a processor erroneously writes a non-zero value in AAZ, MTR frequently checks AAZ.

When MTR detects a non-zero AAZ, it clears AAZ, steps the system (by setting MSCL), puts "ERR ZRO" in the topline message word (TLML), writes the non-zero AAZ in word 57, and clears word 56. As long as byte 4 of word 56 is zero, the system is effectively stopped. When byte 4 of 56 becomes non-zero, MTR unsteps the system and permits system activity to resume.

Central Processor Monitoring

MTR monitors the status of the CPU's to provide several important system control operations: (1) CPU switching; (2) processor time and status control, and (3) exit mode control.

The switching of a CPU from job to job is essentially the task of CPUMTR. MTR assists by clocking the interval of usage, and depending on the setting of the CPU slice time value in CSTI, stimulating CPUMTR to perform the switch (and, incidentally, an elapsed time check). Moreover, when two CPU's are available, a CPU, processing in CPUMTR, which detects the need to switch the alternate CPU, must request that the switching be stimulated by an external processor (in this case, MTR).

MTR also maintains a check of processor status. This includes a check for a zero P register (a sign of an error exit), non-zero values in bits 48-59 of RA+1 of the memory associated with the control point to which the CPU is assigned (an indication of a CPU call for PPU assistance, for example), and CPU recall request statuses (RCL and CCR calls).

PPU Delay Stack Monitoring

An important and powerful processing option available to PPU programs is the ability of the routine to relinquish the assigned processor, and resume processing after a fixed time interval (as specified by the PPU program) has elapsed. This process is called "delayed recall" and is effected by the maintenance of a delay stack by CPUMTR. The delay stack contains the recall, input register (program name, control point number, and parameters) and the time of recall.

Each time CPUMTR accesses the delay stack it determines the most proximate recall time and stores the value in word 43_8 of control point zero (PRTL). MTR checks the value in this word against the real time clock, and whenever the recall time is reached, stimulates CPUMTR to examine the delay stack, to assign the recalled task to a processor, and to update PRTL accordingly.

Control Point Monitoring

MTR maintains a check on the status of all control points to which active jobs are assigned. In particular, MTR stimulates CPU recall and job advancement. CPU recall is stimulated on the fixed clock cycle determined by the CPU slice time (CSTI), or on the basis of parameters associated with the recall function (CCR or RCL) located in word RA+1 of the memory assigned to the control point.

MTR stimulates job advancement in cooperation with CPUMTR. Usually, the events which lead to job advancement stimulate CPUMTR directly. In some cases, however, CPUMTR may be unable to advance the job at the time of stimulus - e.g. a PPU is required and none are available, or a rollout is requested and some processor is active. Thus MTR assists CPUMTR in job advancement by re-issuing the advancement until it can be effected.

Central Memory Testing

Once every 500 milliseconds, MTR attempts to test a section of memory. A section is a group of 100000_8 central memory words. CPU monitor checks section zero (0 through 77777_8). In the 6500, MTR tests section one (100000_8 through 177777_8) on one 500 millisecond cycle, and section two (200000_8 through 277777_8) on another. In the 6400, only section one is tested.

The test is performed only if MTR can locate unused memory in the section. Since both the 6400 and 6500 memories are interleaved, only eight words need be tested. Since memory is allocated in 100_8 word groups, only one group need be available for testing.

The memory test consists of the writing, reading and comparing of three values, eight words at a time. The values are all zeroes, all ones, and alternating ones and zeroes ($5252\ 5252\ 5252\ 5252\ 5252_8$). If a comparison fails, MTR steps the system, places the message "-ERR MEM s" (s = section number) in CM word TLML, and the failing CM address in CM word 55_8.

CPU Program Timer Interrupt Control

CPU programs may request time interval interrupts by making appropriate function calls to the PPU program CPM (control point manager). The call results in the storing of an interrupt time in word TICW (47_8) of the CPU program control point area. The word is adjusted appropriately to compensate for the time during which the program may be rolled out.

MTR scans the timer interrupt words of active control points. When it detects a control point whose interrupt time has arrived, it sends a function request to CPUMTR indicating that fact. CPUMTR performs all of the additional tasks necessary to perform the interrupt, in particular the switching of CPU control to the interrupt address defined by the CPU program.

I/O Stack Monitoring

Several I/O stack processors are used in the Dual System, in particular to control the activities of multiple I/O devices on single channels. These include processing for the CDC 604 tape units and the terminal system (PROCSY) front-end buffer machines. These processors are controlled by I/O stacks located in central memory resident. Each control point is assigned a stack.

Occasionally a stack processor suspends an operation which it cannot complete immediately (e.g. the unloading of a tape which is being rewound) in order to continue with another operation. MTR scans the I/O stacks for such suspended operations, and periodically activates the stack processor for the completion of the operation.

Inter-machine Function Transfer

Function requests are transferred between machines of the Dual System using ECS and CM buffers. Once a second, MTR activates CPUMTR for the transfer of new requests and replies to ECS, and the reading of replies and requests from ECS.

Top-line Error Stops

A single central memory word in control point zero, TLML, (location 15_8) is used for storage of critical system message information. MTR scans bits 48-59 of this word for the display code characters "*E". When MTR locates this value, it steps the system, and changes bits 54-59 of word TLML to a "-" (minus sign). Thus a PPU program which detects a system error condition can signal the system operator, step the system, and receive a reply that the system is stepped, by appropriate use of TLML. When the system is unstepped, MTR clears (zeroes) TLML.

Clock and Date Control

MTR maintains two clocks - real time or millisecond and time-of-day. It also maintains the system date. An MTR function is available (ADTM) for altering the time-of-day and date values, as well as a bias value on the real time clock, which, when added to the real time clock, gives the time from midnight in milliseconds.

MTR derives all clocking from a hardware channel clock (channel 14_8) which has a period of 4.096 milliseconds. Thus MTR routines are coded in such a way that no process prevents MTR from sampling the clock channel within 4.096 milliseconds. The channel clock is used to maintain a real-time or millisecond clock, which MTR stores in CM word RTCL (170_8). The clock value is multiplied by four (left shifted two places) in order to enable some PPU programs to use

bits 12-23 (word position three or byte three of five PPU words) as a second clock - milliseconds divided by 1,024.

Once every one thousand milliseconds MTR advances a twenty four hour time of day clock, stored in CM at control point zero word TIML (30_8). It has the display code form

hh.mm.ss.

where hh = hours
mm = minutes
ss = seconds

The date is maintained in similar fashion, and is stored in CM control point zero word DTEL (31_8) in the form

mm/dd/yy.

where mm = month number
dd = day
yy = year

Central Processor Monitor - CPUMTR

A third element of the system executive which is loaded at deadstart is CPUMTR. Since CPUMTR is stored in relocatable CPU code on the dead-start tape, it can be stored in any area of central memory resident through the relocation facilities of SET.

After SET has relocated and loaded CPUMTR, has halted the CPU's and determined their configuration, it starts all CPU's through a preset routine of CPUMTR. This preset routine, which overlays the PPU dump area used for IDD dayfile dump, establishes the various constants used by CPUMTR, and presets the exchange areas used by the PPU's to reference CPUMTR facilities.

CPUMTR Access

All access to CPUMTR is based upon the use of the exchange instruction, and thus, of course, exchange areas. Since access calls to CPUMTR come from both the PPU's and from CPU's processing user jobs at control points, for convenience and flexibility each source of access has a separate exchange area reserved for it. Thus there are ten PPU exchange areas, an exchange area in each user control point, and a special exchange area for system usage.

PPU Access

When a PPU routine issues a function request which requires CPUMTR action, (appendix C) the FTN subroutine accesses CPUMTR via a monitor exchange instruction (MXN) addressed to the exchange area reserved for the PPU and designated for a particular CPU. Several aspects of the MXN instruction and the exchange operation in general play an important role in regulating the access to CPUMTR.

In the first place, the exchange operation effects a complete swap of the 16 words in the exchange area with the operating registers of the CPU. That is, the registers of the CPU are stored in the exchange area and loaded with the contents of the area, before the registers are stored, without destroying either set of information.

Secondly, the MXN instruction is part of a 6000 hardware option called "monitor mode". In this option the MXN instruction and a corresponding CPU instruction, the XJ, are added to the existing exchange logic, together with a monitor mode flip-flop for each CPU. The MXN sets the monitor mode flip-flop. The XJ clears it, when set, and sets it, when clear. When the monitor mode flip-flop in either CPU is set, an MXN instruction has no effect.

In particular, once the FTN routine has set B0 in the exchange area non-zero, and has issued an MXN, it can detect the hardware response to the MXN by checking B0. If B0 is still non-zero then the exchange did not take place (of course, the non-zero B0 set by FTN is ignored by the exchange hardware).

When FTN sets B0 non-zero it also must set the P register (the location at which the execution of CPUMTR will begin) and A0. The P value is set in the exchange area by the CPUMTR preset routine and stored in PPU memory by the preset routine of PPR. A0 is set at each exchange to the CPU number being designated by the MXN instruction.

Other registers are set by CPUMTR preset, and are preserved for the duration of execution. These include:

```
 RA = 0
 FL = The machine FL
RAX = 0
FLX = The machine FLX (bit 23 = 1 for flag register access)
 B1 = The PP output register address
 B7 = 1
```

CPU Access

A central processor program can also access the facilities of CPUMTR. Again, the operation involves the exchange operation and the monitor mode option. In the case of the CPU however, the XJ instruction is the stimulus.

When the XJ instruction is issued by a CPU, the exchange area targeted de-

pends on the setting of the monitor mode flip-flop. The CPU number is, of course, determined by the processor in which the instruction is executed.

If the monitor mode flip-flop is set, the exchange area address will be determined by the effective address calculated from the B_j and K portions of the instruction.

This is the normal case for exits from a PPU function call, and the effective address is the PPU exchange area. The result of the exit is to return the CPU to the processing which the MXN operation interrupted. If, however, the monitor mode flip-flop is clear when the XJ instruction is issued, (the case for a control point problem access request), then the exchange area address is taken from a special register in the exchange package, called MA (the monitor address register). An XJ issued by a CPU whose monitor mode flip-flop is clear obeys the following rules:

1. The exchange area address is taken from MA.
2. When the exchange completes the monitor mode flip-flop will be set.
3. The exchange is delayed until the monitor mode flip-flop of the alternate CPU is clear.

Because of the influence of the MA register on an XJ exchange operation, its assignment, and the contents of the 16 word area to which it points are critical to the proper processing of the XJ instruction issued when the monitor mode flip-flop is clear. (Often the state of the CPU with the monitor mode flip-flop clear is called "problem mode"; when set, "monitor mode".)

Both the MA value and the exchange area are established by CPUMTR whenever it assigns a CPU to a control point, or to a system, "problem mode" task. The basic assumption is that such an assignment is always made by a CPU, processing CPUMTR code in monitor mode. Thus some PPU issued an MXN which triggered the process.

The MA of a control point to which CPUMTR is assigning the CPU is always set to the address of the exchange area in the control point area (in fact, the first word of the control point area). CPUMTR starts the CPU for the control point by issuing an XJ instruction whose effective address is the exchange area of the control point. The net effects of this operation are:

1. The registers of CPUMTR are swapped with those of the control point.
2. The CPU leaves monitor mode and enters problem mode.
3. The MA register in the running, problem mode CPU contains the control point exchange area address.

Consider, now, the effect of an XJ issued by the problem mode process. Since the CPU is in problem mode, the exchange address is taken from MA. Thus the problem mode registers are stored in the control point exchange area, the CPUMTR registers are placed in the CPU, and CPUMTR processing resumes, in monitor mode, with the instruction following the XJ instruction with which CPUMTR previously

assigned the CPU to the control point. (This is true, since the P register stored in the exchange area represents the P register after the RNI cycle which is part of the processing of the 60 bit, XJ instruction.)
With control returned to CPUMTR, and the problem mode exchange package safely stored in the control point area, CPUMTR can process the XJ access request and return to the caller with a single XJ, and can, optionally, choose to assign the CPU to another problem mode control point.

The function request itself is fully described in the 60 bits of the XJ instruction. In fact, since only the 9 bit, XJ operation code is used for a problem mode XJ, there are 51 bits available for passing parameters.

CPUMTR locates the XJ instruction at (P-1) of the control point exchange package area. The bits of this instruction word have the following assignment:

Bits	Contents
59-51	013_8 (the XJ operation code)
50-48	B_2
47-30	K_2
29-21	Function Code (appendix D)
20-18	B_1
17- 0	K_1

The parameters $(B_1)+K_1$ form one 18 bit parameter, called Parameter 1; $(B_2)+K_2$ form Parameter 2. The function code (Appendix D) serves the same purpose as the PPU function code: it defines the access request.

PPU Function Processing

When a PPU issues an MXN via the routine FTN accessing CPUMTR, the P register stored in the exchange area directs the CPU to the special entry point of CPUMTR, labelled MXN. This entry point pre-processes all function calls. In effect, this involves the following steps:

1. Checking the status of the CPU which was interrupted (on/off);
2. Checking step mode status;
3. Isolating the function code (from the output register word) and the control point assignment (from the input register word) of the caller, and entering the processor.
4. Entering statistics measurement data, if that mode is enabled.

CPU Availability

The FTN routine of PPR targets the MXN to a particular CPU and exchange area. When PPR executes its preset routine at dead-start, the CPU target is set to the first available CPU, or CPU 0 if neither is available. FTN continues to use that CPU, subject to the status check at each MXN entry time; and miscellaneous repeat requests issued by specific function processors.

The status check made at MXN entry time is simple: if the CPU in use is logically "on", processing can continue; if "off", the output register, CPUMTR flags are set to request a repeat on the alternate CPU. Thus a CPU can be turned "off" and the FTN routine of each PPR will automatically switch to the alternate CPU. Note that for the above reasons, CPUMTR will not permit both CPU's to be "off" at the same time.

Checking Step Mode

The process of stepping CPUMTR, or permitting the processing of functions only by an entry of the console keyboard space bar, is controlled by the CM word, MSCL. If the contents of MSCL are zero, no step mode is set, and MXN processing can continue. If they are non-zero, they have the following significance:

Bits	Contents
59-42	The control point in step (0=all)
41-24	The function in step (0=all)
23-18	Non-zero
17- 0	The PP address being advanced

Step mode processing follows the rules:

1. Neither MTR nor DSD functions can ever be stepped.
2. If the PP address in MSCL matches the address in B1, bits 0-17 of MSCL are cleared and MXN processing continues.
3. If all functions and all control points are stepped, the CPUMTR flag in the output register (bit 2^{59}) is set and CPUMTR exits from MXN.
4. If the PP is not assigned to the stepped control point, and/or not requesting the stepped function process, MXN processing continues, otherwise the step flag is set as in (3), above.

Function Processing

The function number in the PPU output register selects the processor to be invoked. Currently, bits 48-53 constitute the function number, thus permitting 63 functions (0 is not meaningful).

The control point address of the caller's assignment is selected from bits 36-40 of the input register. MXN enters the proper processor through a jump to a function jump vector, indexed by the function number. At entry time the following register contents are meaningful:

Register	Contents
A0	CPU Number
B1	PP output register address
B2	0
B3	Control Point Area address
B5	The function number
B7	1
X0	The function number
X1	The output register, left shifted circularly 12 bits
X2	A 12 bit mask, left justified
X3	A 42 bit mask, left justified
X4	A 48 bit mask, left justified

Processor Exits

Whatever the specific actions of a function processor may be, there are only five basic paths through which it may exit. All but one of them (the CPU switching exit) eventually result in the execution of an XJ instruction to switch the CPU from monitor mode back to the interrupted problem mode process.

The No Operation Exit

The NOP exit is used to process those function requests for which no processor is defined. It simply consists of the calculation of the PPU exchange address, and an exit to the return exchange instruction of MXN.

The exchange address is calculated by the formula

$$X_a = 2 * P_a + X_k$$

where

X_a = exchange address

P_a = PP output register address (B1)

X_k = The address of the exchange area for PPU-0 minus 2 * PPU-0 output register address.

Note that this calculation presumes that the PPU exchange areas are contiguous, and arranged in ascending order of processor number.

The Normal Exit

In the normal case, the function processor performs a simple task, forms a response, and returns control to the interrupted problem mode task. This normal exit routine is used for such processing.

At entry (X6) must be the output register reply. Typically bits 48-59 will be zero, but this exit can be used to transmit other replies, as in the wait and repeat exit cases. The contents of X6 are stored in the output register word, the exchange address is calculated by the standard formula already stated, and the return XJ instruction is issued.

If the statistics mode is enabled, the statistics sampling routine is also executed. Using data stored at entry time this routine provides a summary of the access request.

Note that the following register contents will appear in the PPU exchange area:

B1 = output address
B3 = exchange address
B7 = 1
RA = 0
FL = Machine FL
RAX = 0
FLX = Machine FLX

The Wait Exit

Occasionally, a function processor cannot complete its task until some other event occurs. A good example is the processing of a channel request when the channel is reserved. Rather than simply request that the PPU repeat the function access, thus incurring a large amount of processing overhead, selected function processors will set a wait response (bit 57) in the CPUMTR flag bits.

Such an action presumes that some other stimulus will cause a reexamination of the waiting request. Thus, to extend the previous example, the RCHM processor sets the wait response; the DCHM processor checks for it.

After the wait response has been formed in X6 (the low 57 bits of the original request, plus the 1 flag), the wait exit processor passes control through the normal exit back to the interrupted problem mode process.

Repeat-Request Exit

The operation of the repeat request exit is similar to that of the wait exit and the MXN entry code when it detects that the CPU is "off" X6 is set to the low 57 bits of the request plus the repeat code (bit 56), plus the repeat CPU number (0 or 1) in bit 55.

This exit is normally used by those function processors which deal with CPU status. For example, a request to change the exit mode register for a control point when issued to CPU-0 will cause a repeat-request reply for CPU-1, if CPU-1 is assigned to the control point.

CPU Switching Exit

Some function processors by nature cannot return CPU control to the interrupted processor (e.g., the drop CPU function processor). In those cases this exit is used.

This exit effects the copying of the interrputed exchange package contents to the proper control point exchange area, the restoration of the PP exchange area, and the selection of a new problem mode usage for the CPU. Prior to starting the new, problem mode task, the PPU output register is cleared.

Problem Mode Function Processing

Another case in which the CPU is ordinarily not returned to the interrupted task involves the processing of function requests in problem, rather than monitor mode.

Such capability is particularly desirable for function processing of long or indeterminate duration; for example, storage relocation. Since the function request processing begins in monitor mode, CPUMTR contains a transition facility by which the processing is completed in problem mode.

This facility makes use of an extra control point called the system control point, reserved solely for CPUMTR usage and numbered N+1. It contains only enough space for the exchange area and the CPU scheduling words (status, priority, time, etc.).

When CPUMTR receives a problem mode request, it first checks the status of the system control point. If it is busy, a wait exit response is issued, and control returns to the interrupted process, which may, in fact, have been the busy problem mode function processor.

If the system control point is free, it is set busy, the system control point status is set to 'W' (waiting) the interrupted package is copied to its proper area, the PP package is restored, and CPUMTR re-schedules the CPU. Normally the CPU priority of the system control point is such that the CPU will be scheduled to the system CP, but this is not an absolute requirement,

since the 'W' status guarantees that the CPU will be assigned to the system control point as soon as it becomes available.

The scheduling of the CPU to the system control point is done exactly in the same manner as that for all other control points: MA is set to the control point exchange area; and XJ is issued to that address. Thus, when the problem mode function processing is complete, and the proper response has been stored in the requesting PPU output register, control returns to CPUMTR via an XJ operation.

In fact, control returns to the XJ function processor. However, the special address of the system control point blocks normal processing. Instead, the PP output registers are checked for functions waiting for problem mode processing. Should one be found, the CPU is re-scheduled to process the function at the system control point. Otherwise, the system control point is set inactive, and another problem mode process is selected.

A problem mode processor may also return to CPU monitor to process some interlock operation in monitor mode and then regain control in problem mode. The problem mode processor simply modifies the return instruction counter (P) in the system control point exchange area, stores parameters in the X registers of the exchange area and issues an XJ. When the monitor mode operation completes, it returns to problem mode by executing the XJ instruction which first transferred control. This leaves the system exchange area (particularly the P register) set for the return of control when the problem mode processor has finished.

CPU Function Processing

As already noted, problem mode CPU processes may make access requests to CPUMTR. The entry location is controlled by the setting of MA and the exchange area to which MA points.

When an XJ access operation is issued, the function request is decoded from the contents of the XJ instruction itself. Processing action is very similar to that for MXN calls; the function code serves as an index to a jump vector. On entry to the function processor the following register conventions apply:

Register	Contents
B1	Control Point FL
B2	-1
B3	The control point exchange area address
B5	Parameter address one
B6	Parameter address two
B7	1
X0	0
X2	The control point RA
X3	The control point FL
X4	The contents of the XJ word

Illegal function numbers cause a control point, CPU abort (ABT in RA+1, for example). Function processors may make use of the parameters of the exchange word, may change values in the exchange area, etc.

There are two basic methods of exit: back to the caller at the instruction following the XJ; or to CPUMTR for re-assignment of the CPU. In the first case, reply conditions, etc., depend upon the processor.

When an XJ function processor returns to CPUMTR, the negative value in B2 signals a return from an XJ call. CPUMTR may choose to assign the CPU to another job, assign a PPU to the requesting process, and may even return to the process via an XJ. In any case, the fact that MA is set to the control point exchange area address means that the problem mode exchange package area need not be moved, and thus no special processing is necessary to switch to a new task, or return to the one initiating the XJ.

CPU Control

A large amount of code in CPUMTR is devoted to the control of the CPU's themselves. Some of it has already been described in the discussions of CPU assignment and XJ function processing.

The remaining code deals essentially with processes peripheral to the assignment of a CPU to a particular process. These can be divided into: (1) process selection and (2) process checking.

Control Point, CPU Status

The selection of a program for CPU assignment depends directly upon the CPU status in the control point status word, STSW (21_8), of the control point of the program. The CPU status occupies bits 54-59 and can assume the following values:

Value	Status	Meaning
40_8	W	The control point is waiting for a CPU
20_8	X	The control point is on recall by direction of the CPU.
10_8	I	The control point is waiting for an I/O operation or PPU call to finish
4	R	The control point is waiting for an I-type processor to finish and the processor is in delayed PPU recall

X Status

The X, R, and I statuses merit some additional explanation. A processor reaches X status through a voluntary surrender of the CPU. It can do so by placing "CCR" or "RCL" in bits 42-59 of RA+1, or by issuing an XJ call.

The "CCR" is a general "check CPU recall" request. Bits 0-17 of the call reference a function word of the form:

```
59    = count function if set, test function if clear
58    = bit test value
57-30 = unused
29-18 = bit number
17- 0 = word address or count
```

Depending on the option requested, CPUMTR will return CPU control after "count" CPU slices or when the addressed bit reaches the specified value.

The "RCL" call may request that CPU control be returned after bit zero of the word addressed in bits 0-17 is set. If the address is zero, the "RCL" request suspends CPU activity for one CPU slice.

A CPU slice is twenty milliseconds in the Dual System. This value is stored in CM control point zero word CSTI (46_8) and may be altered during system execution. PPU monitor regulates the CPU sliced activities. In addition, MTR checks the "CCR" and "RCL" conditions before stimulating CPUMTR to reassign the CPU.

In rare cases, CPUMTR may set "X" status for a program when a request for PPU access cannot be satisfied. This case rarely occurs, since CPUMTR will enter as many as four PPU delay stack requests before delaying the CPU program with "X" status. This type of "X" status is processed in the same fashion as an "RCL" call with no address - the delay is at least one CPU slice.

I and R Status

When the CPU requests the assignment of a PPU to perform an I/O or control operation it may surrender the CPU until the operation is completed by setting bit 40 of the call (RA+1 or the call word passed to the XJ processor). The processor is assigned, the control point is given I status, and the address of the assigned PPU is placed in RA+1, right justified.

When the assigned PPU terminates, a match of its address and RA+1 causes a change in the CPU status from I to W, in effect, restarting the CPU. Of course, RA+1 is also cleared.

Should the assigned PPU be unable to complete its task, and instead, enter delayed recall, a match of the PP address and RA+1 causes the CPU status to change from I to R. RA+1 is set to the delay stack address of the recalled processor. When the processor is recalled, the stack address, the R status, and RA+1 combine to reset the status to I.

CPUMTR may also set R status when it must place an I status request in the PPU delay stack because no pool PPU is available. This has the same net effect on the CPU program as a transition from I to R status.

This transition from I to R and back again is necessary for the proper control of CPU assignment when the job is rolled-out (swapped). In that case, the PP causing R status is noted, and at roll-in time, the processor register is replaced in RA+1, and the CPU is assigned. Thus the CPU status of a job at roll-in time is simply W or none at all.

CPU Selection - W Status

When CPUMTR searches for a program to which to assign a CPU it considers only those jobs in W status, whose control points do not have move status set (bit 34 of STSW. The selection is based on a CPU priority value in bits 0-17 of word JCIW of the control point area. The highest priority program, excluding the one to which the CPU is assigned, is selected.

Each CPU priority class (zero through three) is represented by a linked-list queue. Thus the CPUMTR search for a program can be restricted to the non-empty queues of priority not less than that of the program using the CPU. New requests enter these queues at their head rather than tail. This tends to give better service to programs which require the CPU in short bursts, typically I/O bound jobs.

Should more than one control point have W status, and provided that an alternate CPU is available, the selection of a new job may also activate the path from CPUMTR to MTR to the alternate CPU for its own selection of a process for assignment.

When a new process is selected, and/or an old one terminated, various clock and time accumulation cells are updated. These reflect all of the values necessary to regulate and charge CPU time usage, as well as those by which MTR monitors CPU switching.

CPU Checking

Before CPUMTR returns or assigns initial control to a program, a number of checks are made on the program status, including: (1) P register status; (2) RA+1 status; (3) Program Stop status; and (4) time accumulation status.

The P register in the exchange area must be within the range established by FL. Moreover, it must not be zero, since zero is used by the exit mode hardware to signal an error exit. The occurrence of either case causes the control point error status to be set. This in turn triggers job advancement.

If the P register is in range, but the operation code (upper 6 bits) of the

next instruction is zero, the control point error flag is set to denote a program stop error. The control point is then eligible for job advancement.

Should the upper 18 bits of RA+1 contain a non-zero value, CPUMTR recognizes a call for CPUMTR or PPU action. CPUMTR action calls include:

RA+1, bits 42-59	Action
ABT	End the CPU and set the CPU abort error flag
CCR	General, CPU check request
CIO	Intercept stack processed I/O calls. All others are passed to the PPU program CIO
END	End the CPU
RCL	Recall the CPU (see the X status discussion)
MSG	Process a message
TIM	Return a time value

Other values beginning with an alphabetic character, and containing only alphanumerics are considered to be PPU calls. Remaining values constitute errors, and result in the setting of the "PP CALL ERROR" flag.

Finally, before assigning the CPU to a program, CPUMTR checks the accumulated CPU time of the control point against the job card estimate. Provided the accumulated time does not exceed the estimate, the CPU is assigned. Otherwise, the "TIME LIMIT" error flag is set, and the job becomes eligible for advancement.

Idle CPU

When there is no activity for the CPU - there are no control points with "W" status - CPUMTR sends the CPU to a small, memory test program. After a short memory test is performed, the CPU is stopped.

The memory test is similar to that performed by MTR. Here, only section zero is tested, again with the zeroes, ones and alternate ones-zeroes tests. The detection of an error causes a Top-line stop - CPUMTR stores "*ERR MEM c" in TLML (c=CPU number).

When the memory test completes, the CPU is stopped at CM location two. The upper six bits of this location is zero (a program stop code, PS). Thus when the CPU is idle the P register value displayed by DSD will be two.

Job Advancement

Normally a job becomes eligible for advancement when the error flag is set at the control point. This depends to a certain extent upon the action of the processors assigned to the control point. The CPU is always removed, and CPU status is cleared. Most PPU routines drop when they sense the error flag in

STSW (usually after a call to PRL). All delay stack entries are purged.

The three preceding conditions actually constitute the factors which affect job advancement: the job must have no processor status active or pending, including I/O stack entries. When the conditions are met, CPUMTR advances the job by invoking the processor indexed by bits 30-35 of word JCIW of the control point. Usually the processor is 1AJ, for primary level, "Advance Job". In special cases it may be:

Processor	Action
1CJ, 1RB	Terminate or re-start job
1RO	Rollout Job
1RI	Rollin Job

The action of rolling out a job constitutes one exception to the general job advancement rule. If rollout status is pending, 1RO will be called as soon as all PPU activity ceases.

PPU Control

PPU activity at the control point and in the system is controlled by CPUMTR exclusively. As already noted, a pool processor is assigned by placing a request in its input register, and released through one of several different function calls.

When a processor is assigned to a control point the status word (STSW) of the control point is changed to reflect the assignment: a count of active PPU's in bits 48-53 is incremented. When a PPU enters recall, a separate count in STSW (bits 24-29) is incremented; when the package is re-assigned, the count is decremented.

Using the CPU, PPU and delay status fields of STSW, and the header word of the control point I/O stack, CPUMTR or MTR can make a rapid check of job status. When job advancement is necessary, a fourth field in STSW, bits 30-33, are set to the index of the processor assigned (1AJ, 1RO, 1CJ, 1RI). Thus CPUMTR can strictly control and recognize the exact advancement status of the job.

An additional layer of PP control can be added to that of job advancement. It consists of the control of the number of PPU packages of a given name active. Thus, for example, CPUMTR can be instructed to limit the number of pool PPU's assigned to 1AJ, 1RO, 1CJ, etc. In practice only 1RO and 1RI are regulated (2 copies each).

A number of other processors, however, are limited to single, active copies. These are called "unary" processors, and include those routines whose proper operations does not permit multiple, simultaneous copies to be in execution. The job scheduler, 1SJ, is a good example.

Unary processors form a special class, whose members are regulated by CPUMTR, whatever their status. Thus, for example, CPUMTR records the whereabouts of ISJ, whether in the delay stack, in a PPU, or never active. Several PPU functions are devoted to the special or general invoking of unary processors. Several other functions use the status record of unary processors to affect their processing (job scheduler disabling at a control point, or I/O stack processing are examples.)

In regulating job advancement, processor count, unary processors, the delay stack, and in assigning PPU's in response to PPU function requests, CPUMTR depends upon the eight pool processors. In the pool, a processor is available if its input register word is zero; not available, if non-zero.

CPUMTR keeps track of available PPU's by maintaining a cell whose contents point to one of the available PPU's. When a PPU is assigned the cell is changed to zero if no other PPU is available, or to the address of another available PPU. Correspondingly, when a PPU is released, the cell is updated, if zero.

The process of releasing a PPU is also interwoven with processor count and unary processor control, in a manner corresponding to the actions of assigning a PPU. This includes updating the PP count in the control point status word, the processor count, and the unary processor status.

Statistics Mode

When CPUMTR is assembled, it is possible to specify the insertion of performance measurement code. This code is assembled in an inactive state. It is activated when a properly authorized CPU program at a control point makes an appropriate function call to the PPU program CPM. That call activates statistics mode by replacing key instruction sequences with entry calls to CPUMTR statistics routines, and by linking the memory of the CPU program to the CPUMTR statistic routine with buffer pointers.

With statistics mode thus activated CPUMTR will transmit variable length records to the buffer of the sampling CPU program. These records describe PPU function requests, CPU function requests, and CPU assignments, as well as the CPUMTR responses to the various requests. The data supplied to the CPU program can be reduced, filtered or analyzed as it is copied to the program buffer, or it can be transferred to a file for later, detailed analysis.

This method of statistics gathering has two important advantages: (1) the space and time dedicated to statistics sampling is very small when no sampling is in progress; (2) the sampling program can measure its own impact on system performance, since it receives statistics records indicating the system resources it uses.

Intra-communication Example

As an example of the use of the system intra-communications facilities, consider the sequence of operations required by the following example:

A CPU program at control point two wishes to use the hypothetical PPU program WIS to record one sector of data on file "LFN". When CPU zero is assigned to the program, the program stores the following call in RA + 1 (or makes an XJ request for PPU assignment):

59-42 = "WIS"
41-40 = 2
39-18 = 0
17-0 = 100_8

The CPU then waits for RA+1 to become zero.

While checking CPU statuses, MTR notes the presence of the RA+1 call and starts CPUMTR, using CPU zero, at the PPU entry point MXN using an exchange jump. MTR requests the processing of a check-CPU function (CCPM).

In the processing of the CCPM function CPUMTR detects the RA+1 call. Since the PPU name starts with "W", it is a legal call. CPUMTR then locates an available PPU (assume PPU two, input address 70_8), and places the RA+1 value in the PPU input register with bits 36-41 set to a two (the control point number). Since bit 40 of the call was set, CPUMTR sets the "I" status bit in the status word and 71_8 in RA+1 of control point two.

The "I" status mode of the call requires suspension of CPU activity at control point two. So CPUMTR copies the interruped exchange package from the exchange area of PPU zero (MTR) to control point two, resets the exchange area of PPU zero, and clears the output register for PPU zero, thus freeing it for further processing. Assuming no other waiting programs can be found, CPUMTR sends the CPU to the problem mode memory test and program stop.

Meanwhile the idle loop of PPR in PPU two has copied CM word 70_8 to PPU words $50\text{-}54_8$. Upon detecting the non-zero, "WI" characters in 50_8, PPR stores 50_8 in 13_8, 51_8 in 14_8 and calls the PLL subroutine. PLL makes a call to the FTN subroutine with the A register set to the search PPU library function (SPLM). FTN stores the function code in 10_8, copies $10\text{-}14_8$ to CM word 70_8, sets the P, A0, B0 word in its exchange area, and executes an MXN function on CPU zero.

As soon as CPU zero enters the problem mode memory test, and possibly before it has completed the test, the MXN instruction of PPU two interrupts CPU zero and sends it, in monitor mode, to the PPU function processor. The SPLM function is decoded, the program WIS is located in the library directory, and its location is stored in CM word 71_8 (bits 48-59 $\neq$ 0). CPUMTR returns CPU zero to problem mode in the memory test with an XJ.

The FTN routine senses the change in 48-59 of CM 71_8, and returns with CM

71_8 in PPU words $10\text{-}14_8$. Assuming that WIS is resident in central memory, PLL reads the package to PPU memory using the load address (1073_8), CM word length and CM address supplied by the CPUMTR response to the SPLM call, found in $11\text{-}14_8$. PLL returns to the idle loop with LA (15_8) = 1073_8.

The idle loop sets CP (74_8) = bits 0-5 of 51_8 (the control point number),2, times 200_8, or 400_8, and calls PRL. PRL reads the control point status word at CM 421_8 to PPU words $10\text{-}14_8$. Bit 34 of the word (bit 10 of PPU word 12_8) is zero, so PRL stores (14_8) in FL (56_8), (13_8) in RA (55_8) and returns to PPR ((A) = (RA)). PPR then enters WIS at 1100_8 (LA of 1073_8 + 5) skipping over the header of WIS.

WIS uses $53\text{-}54_8$ as a parameter block address. Adding RA*100_8 to bits 0-17 of $53\text{-}54_8$ it reads one CM word (containing "LFN") to PPU words $40\text{-}44_8$ (FN - FN+4). The WIS calling convention is that the first parameter block word is a file name, and the next 100_8 words are the sector data.

WIS copies $40\text{-}44_8$ to CM word 72_8 the first message buffer word, sets A to the function SFBM (set file busy) and calls FTN. FTN passes the function to CPUMTR as already described. The SFBM function causes CPUMTR to search the file name table for the file "LFN", assigned to control point two. Assuming that the file exists and is not busy (bit zero of the FST word is set zero) CPUMTR sets bit zero equal to zero and responds in the PPU two output register with the file status (FST) word address of the "LFN" entry and a flag indicating that it has been set busy for the PPU. FTN returns to WIS.

WIS notes this response, stores the file address in 57_8 (FA), and reads the FST word to $20\text{-}24_8$ (FS - FS+4). Noting from the FST parameters that the file is assigned to equipment 01_8, track 4401_8 but is positioned at the last sector of that track, WIS requests a new track to be linked to 4401_8 by placing 01 in 11_8, 4401_8 in 14_8, setting A to RTKM (request track) and calling FTN. FTN passes the request to CPUMTR. This reserves space in advance for the end-of-information (EOI) sector of "LFN".

CPUMTR decodes the function. Since RTKM is a "problem mode" function, CPU zero is reassigned from its idle state to control point N+1, and control is given to the RTKM processor via an XJ. Once the track is reserved (e.g., 4402_8) control returns to CPUMTR in monitor mode, the response is transmitted to PPU two, and CPU zero returns to the idle state. FTN returns to WIS.

WIS stores the track number, 4402_8, in PPU word 7000_8. It sets 6 (T6) to 4401_8 and 7 (T7) to the last sector (60_8). It stores the equipment number 1 in 5 (T5) and calls SMS. SMS notes that equipment one is a DB device (808) and that the DB disk driver (6DB) is not loaded. It loads 6DB with a PLL call, executes the 6DB preset routine, sets 4 (T4) = 1 (the channel of DB01) and returns to WIS.

WIS calls RCH to reserve channel one (A = (T4)). RCH calls FTN, A = RCHM (request channel), and 11_8 = 1. FTN calls CPUMTR, which assigns the channel. FTN returns to RCH; RCH returns to WIS. WIS calls POS in 6DB. POS connects and reserves DB01, links the DCH subroutine to its drop channel processor (DCP), and positions to track (T6).

WIS reads the sector data from CM, starting at $53\text{-}54_8$ (0-17) plus RA times 100_8 to PPU memory, starting at 7002_8. The 100_8 CM words occupy 500_8 PPU words, $7002\text{-}7501_8$. The next track value (4402_8) stored in 7000_8 is control byte one. It indicates that track 4402_8, sector zero is the next sector of the file. Control byte two, 7001_8, is set to 100_8 for a full sector word count. WIS sets A to 7000_8 and calls WDS, the disk write routine of 6DB. When the sector has been recorded, WDS returns to WIS.

WIS now records the EOI sector. It sets the EOI control bytes (zeros in 7000_8 and 7001_8). It places the new track (4402_8) in T6, zero in T7 and calls POS. When POS returns WIS sets A to 7000_8 and calls WDS. WDS records the EOI sector and returns.

WIS drops the channel reservation by setting A = (T4) and calling DCH. DCH, in turn, enters DCP in 6DB. The unit reservation is cleared, DCH is unlinked from 6DB, and the channel reservation is dropped via a function DCHM call to FTN. FTN returns to DCP; DCP to DCH; DCH to WIS.

WIS sets bit zero of the FST word to one (bit 0 of 24_8) and writes the FST word to CM at (FA). This sets "LFN" non-busy. WIS now releases the PPU with a DPPM (drop PPU) call to FTN and returns to the PPU idle loop at PPR.

The DPPM processor in CPUMTR clears the PPU two input register word (70_8). It notes the "I" status of control point two, to which PPU two was assigned, and that RA+1 of control point two contains 71_8 (the output register address of PPU two.) This signals CPUMTR that the CPU is to reassigned to control point two. CPUMTR clears RA+1, sets the CPU status to "W" and enters control point two in its appropriate CPU queue, as determined by the CPU priority found in bits 0-17 of word 22_8 (JCIW) of control point two.

CPUMTR searches the CPU queues, locates the control point two request and selects it for assignment. This requires that the PPU exchange area of PPU two be reset. No exchange area copy is required, since PPU two interrupted an idle CPU with the DPPM call. After the exchange area of PPU two has been reset, the PPU two output register is cleared, freeing PPU two for further processing (i.e., FTN exits to WIS, and WIS returns to PPR).

The CPU is assigned to control point two ("W" status is cleared). This is effected with an XJ instruction directed at the exchange area of control point two. The CPU resumes processing in the RA+1, clear loop, exits from that loop, and the processing of the example is complete.

Machine inter-communication

The two machines of the Dual System have no direct inter-connection. They are connected, however, through a number of dual access devices, including extended core storage (ECS) and several different types of disk devices. The two machines depend on the contention reservation logic of the disk controllers for regulation of access to those devices. ECS accesses are regulated using the interlock features of a common ECS reservation control register, the flag register. In some cases, ECS access is controlled by the reservation of separate areas for each machine. Finally, the read-only access to the PPU programs stored in ECS needs no regulation.

The processing of disk contentions is performed by each disk driver program individually. Controller reservation waits are thus transparent to the user of the driver. In most cases, once a driver has reserved a controller, the reservation is held until the calling PPU program drops the reservation of the local machine, software channel access.

ECS Flag Register Usage

The ECS flag register is used to control access to the disk track reservation and equipment reservation tables stored in ECS. It is also used to control access to the system permanent file directories. The ECS flag register is an eighteen bit register whose bits are managed with a CPU instruction. That instruction has several options, one of which is especially useful in reservation control.

That option is called selective test and set. Using a mask supplied by the CPU, a bit by bit comparison is made between the mask and the flag register. If every one bit in the mask is zero in the flag register, those bits are set to one in the flag register and the instruction exits successfully. If one or more tests fail, the instruction performs a failure exit.

In the Dual System, each ECS controlled resource is assigned a flag register bit. A machine of the system reserves a resource with a selective test and set operation on the flag register using the appropriate single bit mask assigned to the resource. The machines loop on this test until it succeeds. Once a resource is reserved, the machine uses it for a brief duration, and then clears the reservation with a second flag register operation, a selective clear.

Flag Register Bit Assignments

Twelve flag register bits are reserved for access to shared disks or pseudo-disks. With the current Dual System disk assignments the disk flag register bits are:

Bit	Octal Value	Device	
0	1	6638, channel zero	(DB00)
1	2	6638, channel one	(DB01)
2	4	6638, channel two	(DB02)
3	10	6638, channel three	(DB03)
4	20	844 unit zero	(DJ04)
5	40	844 unit one	(DJ05)
6	100	821 disk	(DH67)
7	200	Permanent file pseudo-disk	(DU74)

Special Access Flag Register Bit Assignments

Two other bits are currently assigned for the regulation of inter-system access. The first is bit 12 (10000_8), assigned to the regulation of the equipment reservation table. This table regulates permanent file and assignable device accesses (excluding magnetic tapes). The second is bit 13 (20000_8), assigned to the regulation of access to permanent file directories. This bit is used only when new entries are being added to the directories, in order to prevent both machines from creating a duplicate entry, or from attempting to use the same open slot for a new entry.

ECS Track Reservation Table Usage

The assigned flag register bits regulate access to the disk track reservation tables (TRT's) in ECS. A number of other factors related to the usage of the disk TRT's affect system performance. These include the ECS access method, the maintenance of a central memory copy, and integrity checking.

Access to the ECS TRT's is complicated by the fact that all TRT table processing is performed in problem mode at the system control point (N+1). Since the FTN routine of PPU resident can interrupt these problem mode processes, and since an interrupted ECS transfer must be totally restarted, the transfer length between ECS and central memory is restricted to the largest, uninterruptible length, eight words.

In order to make maximum use of the data transferred in eight TRT words, the track reservation processor searches each entire eight word block end around, starting with the block containing the last previously reserved track, if known. This not only minimizes ECS access and maximizes usage of the data transferred, but, by virtue of the general, sequential relationship between logical track and physical disk position, it also tends to cluster tracks of the same file in the same disk area, thus reducing the amount of intra-file positioning.

Each eight word ECS transfer goes to or from a central memory TRT image. Thus the presence or absence of ECS access is not critical to the basic processing of the TRT. It does, however, eventually lead to a patchwork situation, in which the central memory TRT's of each machine match the ECS TRT in only selected areas.

Tracks in use by the files of a machine are correctly matched by CMR and ECS TRT entries. In general, since the ECS TRT is copied to CM eight words at a time, not all of the tracks in use by files of the other machine are fully represented by TRT entries. This situation causes no problem when tracks are reserved or dropped, since the ECS TRT is referenced in those operations.

It does cause a problem, however, when a file is moved from one machine to another. The first problem occurs during the reading of the file EOI sector. When the disk read PPU program, 2RD, reads an EOI sector it matches the EOI track and sector with the CMR, TRT linkage data. An error stop is issued if the two values do not match. A second problem occurs if the file is to be processed randomly, since the TRT linkage is the basic device of random file processing.

For these reasons, whenever a file is transferred from one machine to the other (only the file pointers are transferred, not the file itself), the PPU program performing the transfer issues the CPUMTR function RTCM (request TRT change) which copies the entire TRT from ECS to central memory. The same function is used whenever a new permanent file access is granted (via the equipment reservation table) in order to insure that the full, TRT linkage chain is present in central memory.

The RTCM function also generates a track usage count. This count is stored in ECS, and in the central memory TRT pointer. The count is updated by the track reserve and drop processors of each machine.

The track reserve and drop processors perform a number of integrity checks on the TRT's during each access. These include the obvious checks - dropping unreserved tracks or system library tracks, attempting to link to unreserved or linked tracks, etc. In addition, the processors make a consistency check of each TRT word using two four bit fields - the free and reserved bit fields.

Each bit of the free field is assigned to one of the four tracks represented by the TRT word. If the bit is set, the track is free; clear, reserved. Each bit of the reserved field is assigned to the same, corresponding tracks. The value usage is inverted. If a bit is set the track is reserved; clear, the track is free. By insuring that both bits have proper values, the TRT processors can detect TRT destruction at a very early stage, including the most common type of destruction - the erroneous clearing of TRT words to zero.

When the TRT processors detect TRT content errors, they make use of the Top-line error stops to inform the system operator of the error, and to halt system activity before further damage results. In general, one of the recovery dead-starts (MINOR or MAJOR) may be performed to restore the system after a TRT error stop.

Inter-machine Function Processing

There are two ECS areas reserved for inter-machine communication. From one machine one area is used for transmitting function requests and receiving replies (the "OUT" blocks), the other area is used for receiving function requests and transmitting replies (the "IN" blocks). The other machine uses the areas in reverse order: it receives requests and transmits replies in the first area ("IN" blocks); it transmits requests and receives replies in the second ("OUT" blocks).

Both machines transfer data to and from the ECS inter-communication areas from corresponding central memory buffers. The actual transfer of requests or replies and reception of replies or requests is controlled by the contents of the central memory buffers in each machine. The processing of the buffers is performed once a second by CPUMTR at the request of MTR.

An individual function request block in the inter-communication buffers is eight words long and has the following format:

Word(s)	Bit(s)	Contents
1		Block header word
	59	Set if the block is active
	58	Set if a reply is waiting to be transmitted
	57	Set if processing is in progress
	56-48	Assignment status
	47-36	Address of assigned job
	17- 0	Function or reply
2-8		Function parameters or reply data

The "IN" and "OUT" blocks of each machine are cleared at dead-start. When a PPU program wishes to make an inter-machine request, it first reserves an "OUT" block by making a RIFM (reserve inter-communication function block) function call to PPU monitor. MTR locates a free block (bits 48-59 of the block header word = zero), stores the PPU output register address in 48-56 of the block header and returns the block address to the requesting PPU. The PPU fills in words two through seven with the function parameter data. It then updates the block header with the job address (file address or terminal number) and a non-zero function code.

When CPUMTR is activated for block processing by MTR via an MTRM (MTR services) function call, it scans the "OUT" blocks looking for a block whose active status (bit 59) is clear, and whose function field (0-17) is non-zero. Such a block represents a function to be transmitted. CPUMTR sets the active bit and writes the function to ECS.

Block processing also includes a scan of the "IN" blocks. Consider the action of CPUMTR in the second machine in processing an "OUT" block activated by the first machine (received as an "IN" block in machine two). In scanning the blocks, CPUMTR checks the active flag, the reply ready (bit 58) and the function fields. If the active flag is clear, CPUMTR reads the block contents from ECS, looking for a new request. If the active flag is set, CPUMTR skips the block until the reply ready flag is set. The request of such a block is still being processed. Finally, if the block is active and the reply ready flag is set, CPUMTR sets the function field zero, clears the reply ready flag, and writes the block to ECS.

The actual task of processing the function requests received in "IN" blocks and routing replies received in "OUT" blocks is one of the activities performed by the PPU program IRS, the system repetitive services processor. IRS is assigned to a PPU once every 250 milliseconds. Using timing data stored in central memory, IRS performs a wide variety of repetitive processes, whose cycles range from once every 250 milliseconds, to once every eight seconds.

Once every 250 milliseconds IRS scans the "IN" function blocks. From the contents of the "IN" block header words it can detect new requests. IRS performs some requested functions, and passes other functions to the PPU program ICP, inter-communications processor. When IRS passes a function to ICP it sets the processing active flag (bit 57) in order to avoid repetitive calls to ICP. The ICP program is activated at control point zero by an RPJM (request PPU job) function call from IRS to CPUMTR. Parameters in the ICP input register identify the block to be processed. When the function has been processed, IRS or ICP writes the reply data in words two through seven of the block, and sets the reply ready flag. On its next "IN" block scan, CPUMTR writes the reply block to ECS.

IRS also scans the "OUT" blocks every 250 milliseconds. An active block whose function field is zero signifies that a reply has been received. IRS uses the job address to locate the CPU program which requested the function. A rolled-out caller is readied for rollin by raising its queue priority, and a terminal caller is scheduled for execution by the setting of an interrupt flag in the terminal control block (TCB).

A function request is initiated by a CPU program with a call to MTR (via RA+1) or CPUMTR (via an XJ) for the use of the PPU program ICP. ICP serves double duty: processing "IN" blocks at the request of IRS when used at control point zero; forming "OUT" blocks and receiving replies when used at control points one through N.

The CPU program caller transmits to ICP the address of an eight word function block of the following form:

Word	Bit(s)	Contents
1		Block header
	59-36	Unused
	35-18	Reply code
	17- 0	ICP function code
2-8		Parameter/reply data

The ICP function code in bits 0-17 of the header word is an even-numbered code, local to ICP. ICP sets the field odd when processing is complete. If an error is detected during processing, the field is set odd and negative (bit 17 is set) and an error explanation code is set in bits 18-35.

When ICP has validated the function code and the authorization of the CPU program to make the request, it reserves an "OUT" function block with a RIFM function call. It copies the parameter words to the assigned function block, and completes the block header with job address and function code.

The action of ICP while waiting for the function block depends on the caller. If the caller was a terminal program, ICP interrupts the terminal processor (MESA) and releases the PPU. If the caller was a normal CPU program job, ICP lowers the queue priority of the job so that it will be rolled out, and then enters PPU delayed recall.

When IRS senses that the reply has been received it schedules the terminal caller for execution by setting an interrupt flag; the normal caller by raising its queue priority. The terminal caller reactivates ICP by requesting its reassignment to a PPU. When the normal caller rolls in, ICP re-enters a PPU automatically from the PPU delay stack. In either case, ICP senses the reply and copies the reply data to the caller memory area. ICP releases the function block by clearing the header word, sets the CPU program function block header word odd (bit zero = one) and terminates.

Function Overhead

The preceding discussion has indicated that a number of programs are involved in the processing of a single function. The sequence is basically the following:

1. ICP forms a request in machine one.
2. MTR activates CPUMTR in machine one and the function is written to ECS.
3. MTR activates CPUMTR in machine two and the function is read from ECS.
4. IRS in machine two recognizes the request and effects its processing, possibly involving ICP in machine two.
5. MTR activates CPUMTR in machine two and the reply is written to ECS.
6. MTR activates CPUMTR in machine one and the reply is read from ECS.
7. IRS in machine one recognizes the reply and informs the original caller.
8. ICP in machine one is activated for the caller and completes the processing of the reply.

Since four MTR/CPUMTR cycles (two in each machine), two IRS cycles (one in each machine), a possible roll-in and miscellaneous processing by ICP are required, a normal request/reply cycle can take more than four seconds to complete. This cycle can be speeded up, but to do so would involve a high cost in terms of MTR/CPUMTR and ECS/CM traffic. Generally, the density of data transferred is high enough and the frequency of transfers is low enough that the four second cycle represents a good compromise between cost and speed.

Inter-machine Function Codes

There are currently seven inter-machine functions which are processed by IRS or ICP.

Function 1

Checkpoint System

This function is issued by machine one when the power-down-sequencer hardware is activated. It causes an auto-checkpoint of machine two.

Function 2

Search For Job

The requested machine searches for all occurrences of a specified job name in the file name table or at executing control points.

Function 3

Read Central Memory

Seven words of the central memory of the requested machine are transferred.

Function 4

Request Job File

The FNT and FST words of the first job file (input or rollout) which can be transferred, and whose job parameters match those of the request, are transferred. The FNT/FST entry is deleted in the requested machine.

Function 5

Request Print File

A print file FNT/FST words are transferred as in function 4.

Function 6

Request Punch File

A punch file FNT/FST words are transferred as in function 4.

Function 7

Request CMR File

Selected portions of CM are written to a file and the file pointers are transferred.

File Transfer

Three inter-machine functions effect the transfer of queued (assigned to control point zero) files. Actual file transfer consists of the transfer of the two file pointer words, the file name table word (FNT) and the file status table word (FST). Together these two words fully describe the type, residence and position of the file.

When a file is considered for transfer the two function processor PPU programs, IRS and ICP, examine a number of file parameters in addition to those which may be specified in the file transfer request function block words. These include assignment to control point zero, file residence and file association. Files assigned to control point zero are queued files awaiting processing (execution, printing, etc.).

File residence is restricted to shared disks. These disks are ones having dual hardware access, and for which an ECS TRT was defined at deadstart. In the case of a rollout file, all the files in use by the job, indexed in the first record of the rollout file, must also be resident on shared disks.

The status of the rolled-out files, indexed in the rollout file, is marked in the rollout file FNT entry at rollout time. When the rollout file program, IRO, moves the job files to the rollout index, it checks each file residency. Bit twelve of the rollout file FNT entry is set to one if any file resides on a non-shared disk. This same flag may be set artificially from a control point data field which is, in turn, set by a parameter on the job card. Thus it is possible for a job to prevent its transfer between machines. All jobs entered from the system console select this parameter.

A rollout file may also be ineligible for transfer if the job which it references is attached to some resource list. Since the attachment is by absolute, rollout file FST address, transfer of the file cannot be effected without altering the FST address and destroying the linkage. Jobs in this class include those using mountable equipment (e.g., tapes), terminal control blocks, and those waiting for resource access in the equipment reservation table.

The parameter words of the requesting function block may apply additional parameters to the selection of a file for transfer. These include sixty bit product and difference masks for the FNT word, and pairs of limit values (lower, upper) for the various file parameters - job time limit, field length and queue priority. As many as three product/difference masks may also be supplied for testing the file identification field of the FST entry.

If a file meets all of the tests specified by the requesting function block parameters and by IRS or ICP, its FNT and FST words are removed from the FNT/FST table and transmitted to the requesting machine. An accounting dayfile message is issued to record the name of the file transferred, the file type, and the time of transfer. When the file is received in the requesting machine it is stored in an open FNT/FST entry and a corresponding dayfile message is issued to record its arrival.

In addition, when a file is transferred into a machine, the receiving program (IRS or ICP) requests a CPUMTR update from ECS for the file residence disk. If the file is a rollout file, all TRT's are updated, in order to assure that all index file device TRT's are current. This operation is necessary to bring all CM TRT file linkages up to date for random file linkage processing, and file to TRT linkage checking.

Tables

All of the processes of system intra-communication and machine inter-communication depend heavily upon central memory and extended core storage tables. Almost every aspect of the system from library directories through the control points can be viewed as tables. This chapter discusses in detail all of the essential tables of the system, in order of their appearance in figures two and three, the CM and ECS diagrams.

Control Point Zero

Control point zero is one of two control points (N+1 is the other) dedicated to system activity. It has a format unlike that of control points one through N, using the exchange area, pointers, and control card buffer areas of the normal control point for basic system table pointers and the PPU communications areas.

Word zero always contains zeroes. This value is used by a large number of PPU programs in forming five PPU words of zeroes. The usage of word zero is so heavy, in fact, that PPU monitor maintains a constant check on word zero, and stops the system if a non-zero value is detected.

Words one through 17_8 contain various table pointers and status flags - e.g., pointers to the file name/status table, a "debug" mode flag, etc. Word 20_8 is the system status word (STSW) and functions in a fashion similar to each control point status word to record control point PPU activity and memory assignment. The activities of the various PPU programs which function at control point zero for the system - MTR, DSD, IRS, etc. - are recorded here. The RA (reference address modulo 100_8) field is zero and the FL field is the size of central memory resident in 100_8 groups.

Words 21_8 through 47_8 contain more pointer words. Words 50_8-167_8, as already mentioned are the PPU communications area. Words 170_8-177_8 contain additional pointers. The specific formats of all of these fields can be found in the machine language (COMPASS) statements of the symbolic program library file (OPL) common deck PPCOM(Appendix A).

Control Points One Through N

The N control points (N may be specified at dead-start time as a number from one to twenty one) all have the same format. The first sixteen words are reserved for an exchange area. Word 20_8 contains the control point processor statuses (STSW). Word 21_8 contains the name of the job being processed. The name is blank if no processing is enabled at the control point; "NEXT" if processing is enabled but not in progress.

Words 22_8-27_8 contain various parameters for the control of job activity and CPU usage. Words 30_8-34_8 and 35_8-37_8 form two message buffers, console lines one and two, which are displayed by several different options of DSD. These lines are used by the executing job, and job activity processors to signal job progress and unusual activities (e.g., an I/O unit failure) to the system operator.

Word 40_8 contains input/output activity controls. Words 41_8-47_8 contain an accounting header created by the system job card translator when the input file for the job is written. It contains job description and execution limit data derived from parameters of the job card and a system accounting file.

Words 50_8-67_8 contain additional job execution control values. The job queue priority, which controls job scheduling, for example, is stored in bits 12-23 of word 60_8. Words 70_8-177_8 contain the job control cards. Room is reserved for one sector (100_8 words) plus sufficient additional space (10_8 words) for the maximum length partial control card. The job control card processors buffer control cards to this area from the job input file.

Access to the words of the control point area is controlled by a number of different conventions. The exchange area is accessible if the CPU is not assigned to the control point. The status word is changed only by CPUMTR in monitor mode. The control card buffer and pointers are modified by job advancement processors whose activation is controlled by CPUMTR. CPUMTR insures that when job advancement processors are activated, no other processors are active (e.g., the CPU, PPU's, etc.). Finally access to other pointer words is regulated by requiring PPU programs to use a CPUMTR function, request word change (RWCM), to modify them.

Control Point N+1

Control point N+1 is a short control point area for the use of CPUMTR, problem mode processes. It contains an exchange area (0-17_8), a control point status word (20_8), a CPU priority word (22_8) and CPU time control words (23_8-24_8). These words permit the scheduling of the CPU to the problem mode processes in a manner identical to the assignment of the CPU to control points one through N.

Track Reservation Table (TRT) Pointers

One, two word TRT pointer is allocated to each disk or pseudo-disk device. The TRT pointer words are addressed (located) from the equipment status table (EST) entries by a twelve bit field. Thus they must reside below 10000_8 in central memory. The TRT pointer words have the format:

Word	Bits	Contents
1	59-36	The address of the first word of the TRT.
	35-24	The length of the TRT in CM words.
	23-12	The total number of tracks referenced, excluding those faulted at dead-start time.
	11- 0	The number of tracks in use.
2	59-48	The current device position (logical track). This field is not used by some disk drivers.
	47-36	The number of sectors in a logical track for the device.
	35-24	The ECS TRT, flag register access code (zero if none).
	23- 0	ECS track count and TRT address.

Access to the TRT words is regulated by two conventions. Word one may be changed only by CPUMTR while in problem mode (track counting). Word two may be changed only by a PPU to thich the disk channel of the device is reserved (for position information updating).

Dayfile Pointers

The dayfile pointers contain buffer control and disk status information for the dayfiles of the system. Each control point is allocated a dayfile. In addition there are four special dayfiles allocated: the master dayfile for control point zero; the accounting dayfile; the error log dayfile; and the statistics dayfile. Each dayfile pointer has the two word format:

Word	Bits	Contents
1	59-36	The address of the central memory resident dayfile buffer.
	35-24	The number of words in the buffer.
	23-12	The ordinal of the next unused word in the buffer (the "IN" pointer).
	11- 0	The master dayfile recording status.
2	59-48	The dayfile disk equipment number.
	47-36	The first dayfile track.
	35-24	The current dayfile track.
	23-12	The current dayfile sector.
	11- 0	Unused.

Word one is the central memory buffer control word and word two is the file control word. Word two has a format similar to that of disk files in the file status table (FST).

The buffer control values and some file control values are allocated at deadstart time. The four special dayfiles are also referenced by the second, third, fourth and fifth file name/status (FNT/FST) table entries - SYSTMDF, PDUFILE, EQUIPDF, and STATDF. These FNT/FST entries may be referenced by dayfile analysis programs running as normal, control point jobs. Their access is controlled, however, by the status of permission data in the accounting header area of the control point.

When a job is rolled out, its dayfile pointers and dayfile buffer are recorded in the rollout file. Thus a job can be rolled in at a different control point without loss of dayfile information. When a job terminates, the dayfile is copied to the end of the job print (OUTPUT) file, and the dayfile disk space is released.

A job dayfile is used to record all control card calls, PPU program messages, and an accounting summary. With various control card options, a job user can also obtain statistics summary dayfile messages, and may specify the listing or suppress the listing of library procedure control cards. The system console operator may also enter messages in the job dayfile.

The master dayfile contains all messages recorded in the job dayfiles. The remaining special dayfiles contain some job dayfile information (e.g., accounting information), as well as messages routed only to them (e.g., performance statistics, equipment errors, etc.). The routing of messages to one or two dayfiles at a time is controlled by parameters of the PPU, dayfile message (DFMM) function call to CPUMTR.

Equipment Status Table - EST

All usage and reservation of equipment is managed through entries of the equipment status table (EST). The EST includes entries for all hardware input/output devices available to the system, as well as a number of pseudo-device entries which are used to manage special system resources (e.g., access to individual permanent file directories, access to ECS paging space, etc.).

Each hardware device EST entry contains values which describe the device type (tape unit, disk); the class of the device (shared, mountable); the driving characteristics (channel, unit, select code); the logical on/off status; and the current device assignment. The general format of the one word EST entry is:

Bits	Contents
59	Mass storage device flag
58	System resident device flag
57-48	Assignment status or shared flag, depending on device type
47-36	Channel
35-24	Device parameters
23	Logical on (zero) or off (one) status
22-12	Equipment type code
11- 0	Device parameters or TRT pointer address

Equipment Type Code

The equipment type code of the EST entry (bits 12-22) describes the device type in CDC, Display code. This code identifies the specific kind of device. It is used to select input/output drivers, regulate device management, and to identify special file/device relationships. The following device type codes are defined:

Code	Equipment Description
CR	405 card reader
CP	415 card punch
DB	6638 disk
DG	854 disk pack group
DH	821 disk
DJ	844 disk pack
DX	ECS page space pseudo-device
DY	854 mount queue reservation pseudo-device
DZ	mountable equipment queue pseudo-device
GD	252 display
LP	501 line printer
LQ	512 line printer
MI	729 IV tape drive
MT	604 tape drive
NE	null pseudo-equipment
PR	paper tape reader
PP	paper tape punch
RD	private, 854 pack
SC	film scanner
TH	high-speed front end machine
TL	low-speed front end machine
TP	pseudo-device for tape mounting control
TU	unit-record front-end machine

Equipment Class

The input/output devices of the system are divided into two main classes, mass storage or mountable. In general, mass storage devices are larger, fixed surface rotating devices such as the 6638 or 821 disks whose usage is managed by the system. An exception to this rule, the 854 disk pack, is a mountable, mass storage device.

Mountable devices, on the other hand, are those on which private storage media can be mounted (e.g., a reel of magnetic or paper tape) and whose usage, by the sequential nature of the device, is restricted to a single user. Thus the device is "assigned" to a user. Again, the 854 disk pack constitutes an exception to this rule. Its assignment is controlled with the special, disk pack pointer table (DPT) and TRT reservation bits.

Mass Storage Devices

A mass storage device is by its very nature a shareable device. In the Dual MACE system, mass storage devices are shared among system libraries, queued files, and local scratch files. When a device contains system libraries as well as other, user oriented files, it is also called a system mass storage device.

The leftmost bit (bit 59) in the EST entry defines the class of a device. If it is set, the device is a mass storage device, if not, it is a mountable device. The next bit (58) for a mass storage device indicates the presence (1) or absence (0) of a system copy on the device. Bit 48, if set, indicates that the device is shared by both machines of the Dual MACE.

All mass storage devices located in the first 8 entries of the EST are made available for general file usage (if on, of course). Space on these devices is assigned by the system as needed, in units of one logical track at a time. The size of a logical track depends on the device itself, but always consists of an integral number of sectors, each of which contain 100_8 central memory, 60 bit words. A logical track of the 6638 disk, for example, contains 49 sectors. Some of the track sizes include:

Device	Sectors
DB	49
DG/RD	25
DH	320
DJ	114

The logical tracks of each mass storage device are regulated with a track reservation table, TRT. The EST entry for a mass storage device contains a 12 bit address (bits 0-11) of a pair of pointer words which describe the TRT for the device.

Mountable Devices

As already noted, a mountable device entry in the EST is identified by a zero value in bit 59 of the entry word. When the equipment is assigned to a user, bits 48-56 contain an index to the user, and bit 57 defines whether the index references a control point or a device request block (DRB).

Of the two types of assignment, DRB assignment is the more flexible, since it permits the user process to move from control point to control point without loss of positioning on the device. Both types of assignment represent a three step process, to be discussed in a later section:

1. The user process must identify and request the item to be mounted (e.g., the tape reel number).
2. An operator must locate the item, mount it, and signal its availability to the job.
3. The job must be linked to the item through the device.

Driver Parameters

The remaining bits of an EST entry are reserved for the description of driver parameters related to the operation of the device itself. This includes the input/output channel, and hardware identifiers such as equipment code, unit code, and connect code.

Equipment Ordinal

Since the position of each device entry in the EST is fixed, equipments in the system are always referred to by equipment ordinal. Thus the 6638 disk, described by the third entry in the EST is called DB03 or simply equipment 03.

The EST is limited to a maximum of 64 entries (100_8 CM words). When an equipment is assigned to a control point, its assignment is not only noted in the EST entry, it is also signified by the setting of a bit in a word in the control point area. The bits in the word, starting from left (bit 59) to right (bit 0) represent equipment ordinals 4 through 77_8. No bits are reserved for equipments 0 through 3, since they are usually mass storage devices, and cannot be assigned to a control point.

EST Pointer

The EST is indirectly addressed through a pointer in word ESTP (5) of control point zero. Byte 0 (bits 48-59) contains the address of the first EST entry; byte 1 (bits 36-47), the address plus one of the last entry. Note two important aspects of this pointer:

1. It can be accessed in an economical fashion by a PP with the two instructions:

```
LDN     ESTP
CRD     CM
```

2. The EST must reside entirely below absolute CM address 10000_8, because of the 12 bit addressing.

On/Off Status

The on/off status of an EST entry (bit 23) is altered by CPUMTR. Two PPU functions are reserved for this purpose: ONEM to turn on a device; OFEM to turn off a device. A DSD command entry provides access to these functions from the system console.

Various PPU programs manipulate the on/off status of devices during their processing. For example, a tape drive (MI or MT) is turned off until a tape, released by a job, has been unloaded. This prevents the system from scheduling the usage of the drive until it is free. As another, example, the front-end communications processor, IIM, turns off a front-end equipment when it cannot establish communication with the machine.

Because the mass storage devices in the equipment range 0-7 are the basic system allocatable devices, the DSD command processor will not permit the console operator to turn all of them off. DSD provides other special equipment management in the form of general displays (the "E" display), as well as special displays on which the status of critical devices, such as the front-end machines, are noted.

ECS Equipment Status Table

Shared, mass storage device entries of the EST are also recorded in an ECS, equipment status table. Entries of this ECS, EST have almost the same format as their CMR counterparts. The only exception is the TRT pointer address (bits 0-11) which is a table ordinal in ECS.

The ECS, EST is LOADed during an ECS LOAD dead-start. During an ECS USE deadstart, the entries of the ECS EST are compared to the entries of the CMR EST by the ECS setup program, SXC. SXC notes any discrepancies via a console display. The ECS and CMR entries must be identical in equipment number, type code, channel number, and the first device parameter (bits 36-47).

File Name/Status Table - FNT/FST

The file name/status table (FNT/FST) is the system file index. Each two word entry names a file (word one, the FNT word) and specifies the status of information on the file (word two, the FST word). The table is addressed by a control point zero pointer word in location four (FNTP). Byte zero of FNTP contains the FNT/FST first word address (FWA) and byte one contains the last word address (LWA), plus one.

The FNT Word

Word one of the FNT/FST entry is the FNT word. It has the following format:

Bits	Contents
59-18	Seven character (Display Code) file name
17-12	Special, type dependent flags. Sometimes called the "eighth name character."
11- 6	File type code
5	Interlock flag
4- 0	Control point assignment

The file name may contain one to seven Display Code characters. It is left justified with zero fill, and, except for special system files, never contains any imbedded zeroes or character larger than 55_8 (space). Certain system processors may violate these rules in order to generate files which cannot be accessed by nonsystem programs (e.g., the system checkpoint file, rollout files, etc.).

The "eighth file name character" contains file type dependent parameters. These may have the following values:

File Type	Value of Eighth Name Character
Queued Input	Non-dual flag - i.e., the job may not be processed by the "other" machine of the Dual MACE System
Queued Output and Punch (01,04,05,06)	A single, Display Code character denoting file origin (e.g., Batch, Console, etc.)
Rollout (10_8)	Non-dual flag or Tape unit request flag or Unavailable resource request flag
Local (03)	Open/close flag
Delayed job (15_8)	Delay-end file type (input or rollout)

The interlock flag is used in two ways. (1) When set for any local file it indicates that the file may be read but not written. Thus it is sometimes called a write-interlock or read-only bit. (2) When set for an input file or queued rollout/input file, it serves as an interrupt inhibit flag - i.e., it must be set before an interrupt can be set for a CPU program, and when it is set, the CPU program may not be interrupted.

File Type and Control Point Assignment

There are seventeen file type codes which can appear in bits 6-11 of the FNT word. The file type code and the control point assignment field (bits 0-4 of the FNT word) together divide files into two main classes - queued and local.

Queued Files

Files whose five bit control point field in the FNT word is zero are called queued files, assigned to the system, or control point zero. The queue type is defined by the six bit type field:

Type	Description
00	Job Input Files
01	Job Print Files
02	Common Usage Data Files
03	Local To The System
04	Hollerith Punch Files
05	Binary Punch Files
06	Column Image Punch Files
10_8	Rollout (Swap) Files
15_8	Delayed Job
17_8	System Common Usage
20_8	File-index File

The FST word for queued files contains a combination of file status and job description. Consider the general FST format:

Bits	Contents
59-54	Routing Code ("ID")
53-48	Equipment Ordinal
47-36	First Track
35-24	Parameter 1
23-12	Parameter 2
11- 0	Parameter 3

The three parameters have the following significance with relation to type:

Type	Parameter 1	Parameter 2	Parameter 3
00	Job time limit $/10_8$	Job FL $/100_8$	Queue Priority
01	Dayfile Track	Dayfile Sector	Queue Priority
$02,16_8,17_8$	Current Track	Current Sector	Last File Status
04,05,06	Current Track	Current Sector	Queue Priority
$10_8,15_8$	Remaining Time $/10_8$	Job FL $/100_8$	Queue Priority

The job routing code ("ID") may be used to identify the source and destination device of the job. It is usually transferred from the input device (e.g., a remote card reader) remains through execution, and is then used to route job output to the appropriate device (e.g., a remote line printer).

Local Files

Files whose five bit control point field in the FNT word is non-zero are called local files - local to active jobs at control points. The files may include most of the queued file types, as well as several others applicable only to local usage. An important change takes place, however, in the name of some files which can be both queued and local.

When new job input is placed in the input queue (type 00), the FNT name is called the job name. This job name is one method by which a job is identified during its passage through the system from input queue to control point to rollout queue, back to control point, and eventually to the output queue. When the job is active at a control point, the control point is named with the job name of the original input file. In addition the rollout file and all of the job output files carry the job name.

When the job is still in execution, however, the queued files are essentially local files. They carry the various types (00,01,04,05,06) but their names are selected from the reserved list.

Type	Name	Contents
00	INPUT	Job Input
01	OUTPUT	Printer Output
04	PUNCH	Hollerith Punch Output
05	PUNCHB	Binary Punch Output
06	PUNCH8	Column Image Punch Output

When the job terminates, the output files are renamed with the job name and queued (assigned to control point zero). The input file is released.

Other, local file types which may be used by a job include:

Type	Usage
03	Local, scratch file
07,11,12,13	Permanent file types

A local file FST entry has the general format:

Bits	Contents
59-54	Device dependent parameters
53-48	Equipment number
47-36	Device dependent parameters
35-24	
22-12	
11- 0	Last I/O code and status

When a file is assigned to a disk, bits 12-47 describe the initial and current position of the file. These fields take on other values for other equipment types - e.g., the density, mode and record size for a tape file. The last I/O code and status field describes the last I/O function directed to the file and its result (e.g., end-of-file or record encountered). Bit 0 of this field also serves as a reservation control. When it is clear (zero) the file is in use; set (one), the file is available for use.

PPU Exchange Areas

Each PPU is assigned a separate exchange area. These ten, sixteen word areas immediately follow the FNT/FST Table. PPU zero is assigned the first area, PPU one the second, etc. At dead-start time the preset routine of each PPU resident program sets the PPU exchange address and the CPUMTR, MXN entry address in the FTN subroutine of PPR.

When a PPU requests access to CPUMTR, it rewrites the P, A0, B0 word of its exchange area, and issues a monitor exchange request (MXN). During the processing of the PPU request, CPUMTR can use information in the exchange area for time accounting, CPU switching, etc. In some cases, CPUMTR may even exit without the normal, exchange jump return.

In any case when no CPUMTR activity is in progress for the PPU, the exchange area always contains the following values:

Register	Contents
RA	0
FL	Machine memory size
RAX	0
FLX	Machine, ECS size, and flag register access code
B1	The PPU, output register address
B7	One
EM	0

Track Reservation Tables - TRT's

File space on disk devices is organized into and assigned by units called logical tracks. The track is both a reservation and a positioning unit (see Mass Storage Processing), since each track number can be uniquely mapped into a position on the device.

The track number is a twelve bit integer, of which 11 bits (2^0 through 2^{10}) are the actual track number and bit 2^{11} is a flag bit. The flag bit finds usage in distinguishing a track from a sector in the forward linkage chain of a file. The size of the track number, and the physical capacity of the device determine the exact mapping of the track to the device.

For example, in the 821 disk unit, which has 32,768 physical tracks, each of which have 20 sectors, the total number of sectors, divided by the capacity of 11 bits (2,048) determines a logical track size of 320 sectors.

$$\frac{32,768 \times 20}{2,048} = 320$$

Track reservation, linkage and releasing are managed via a table, called the Track Reservation Table (TRT) and CPUMTR functions. Each mass storage device in the EST has a separate TRT (maximum = 1000_8 words), whose size is determined by the device capacity. The TRT is located, with respect to the EST entry, via two pointer words, whose 12 bit address appears in the EST entry word.

The pointer words have the format:

Word	Bits	Contents
1	59-36	TRT address
	35-24	TRT length (in CM words)
	23-12	Total available tracks
	11- 0	Tracks in use
2	59-48	Current position (logical track)
	47-36	Number of sectors per track
	35-24	Flag register access code, if the device is shared
	23- 0	ECS track count and TRT address

While the length of the TRT is determined by the device capacity, the 11 bit range of the logical track number places an upper limit on the TRT length. That limit is equal to the number of tracks divided by four (1000_8 words), since each 60 bit word in the TRT can reference four tracks:

Bits	Byte	Contents
59-48	0	Track link 1
47-36	1	Track link 2
35-24	2	Track link 3
23-12	3	Track link 4
11- 8	4	Permanent file write access
7- 4		Permanent file access (if track reserved)
		Free bits (if track not reserved)
3		Track 1 reservation flag
2		Track 2 reservation flag
1		Track 3 reservation flag
0		Track 4 reservation flag

The address of a track word is the TRT first word address plus the 11 bit track address divided by four. The low order two bits of the track address index the byte within the word (0 through 3, left to right). Bits 3-0 signal that the tracks of the word, 1-4 are assigned (if 1) or unassigned (if 0).

Bits 4-7, the free bits, form the analog of the reservation bits (0-3). If a reservation bit is clear, then the corresponding position in the free bits must be set. Thus if the track represented by byte zero is free, bit 3 will be zero (unreserved), and bit 7 will be set (free). When the track is reserved, the free bit is used to indicate that permanent file access has been granted to the track. In any case, the combined use of the reservation bits is such that no TRT word can ever be zero. This condition is used by CPUMTR in performing TRT integrity checks.

The track bytes themselves contain file forward links. The track byte corresponding to the first track of a file contains the track number of the next track of the file and so on. The last track of the file will contain a link whose 2^{11} bit is zero, thus terminating the chain. The value of this last link represents the sector number on which file EOI is written.

Since files always begin on track boundries, the position of a given sector of a file can be expressed in terms of its displacement (ordinal position) from the first track. Knowing the number of sectors and using the forward linkage of the TRT it is possible to quickly locate the position of the sector on the device. This forms the basis of the random file capability.

Of course, the primary value of the TRT structure is that tracks and track chains can be rapidly updated with the CPU. Five CPUMTR functions are available for reserving and dropping tracks, reserving and dropping access to permanent files and modifying specific TRT bytes. In particular when a file is dropped, the TRT chain can be rapidly purged without reading/writing on the device.

When a disk device is shared, the master TRT for the disk is stored in ECS. The ECS address in TRT pointer word two locates a track usage count for the disk, followed by the TRT itself. Whenever a machine wishes to modify the ECS TRT, it reserves the TRT by setting the flag register bit indicated in TRT pointer word two.

The ECS TRT is modified by reading it into its assigned CMR area (addressed in the TRT pointer word one) eight words at a time. The eight word transfer is performed by CPUMTR in problem mode, and is the maximum data block which can be moved from ECS to CM without being interrupted by an MXN or XJ. Once the eight words have been modified in CM, they are copied back to ECS. The CM TRT which results is always correct for the files active in the machine, a fact particularly important to random file processing.

When a file is transferred from one machine to another, the TRT of the disk on which the file resides is refreshed from ECS to reflect the full linkage of the file. When a rollout file is transferred, all TRT's are refreshed, in order to move the linkages of all the files indexed by the rollout file from ECS to CM.

The ECS TRT is loaded from the CM TRT of the machine dead-started with the LOAD option. During a USE dead-start, the ECS TRT is copied to CM. In addition, selected portions of the CM and ECS TRT's are updated during dead-start (e.g., during file recovery) through the use of a CPUMTR function (RTCM). Another option of the RTCM function is used to refresh the CM TRT from ECS.

In addition to the free versus reservation bit consistency test, CPUMTR performs a number of additional checks of TRT validity. No track dedicated to the system library may be dropped. No track may be dropped which is not

reserved. No track chain may be extended if the previous track is not reserved, or is already linked to another track. Finally, no permanent file track may be accessed if the track is not reserved.

Errors detected during these checks are indicated with top-line error stops. Use of this emergency system stop procedure is necessary to prevent further system failure, and to take advantage of the early warning which the TRT consistency checks provide.

Equipment Reservation Table - ERT

The permanent file access bits of the TRT are used to control usage of permanent files. This method permits controlling permanent files through available TRT bits, and removes the necessity of storing large, named reservation tables in CM. The equipment reservation table (ERT) is used to queue requests for access to permanent files when the TRT permanent file bits indicate that the requested access cannot be granted.

The ERT is addressed by a control point zero pointer in word seven (ERTP). This word indicates the CM address (bits 36-53), ECS address (bits 0-23) and word length (bits 24-35) of the ERT. The queued entries of the ERT have the format:

Bits	Contents
59	The entry is deleted (no longer in use) if this bit is set.
58	When this bit is set, the indicated equipment/track is available and reserved for the indicated job.
57	The indicated equipment/track is in multiple, read-only use by "count" (0-11) users.
54	Zero for read access One for write access
53-48	Equipment
47-36	Track
35-24	Job address
23-12	Machine number
11- 0	Priority or count

The ERT is terminated by a full word of zeroes.

The equipment/track may represent an actual disk device or the equipment track of a number of pseudo devices - DU, DY and DZ. Equipment DU is used for reserving user segments of the small-file, permanent file subsystem (PFILES). DY and DZ are used for regulating access to single device mountable equipment (DZ) and to device request blocks describing mountable equipment (DY).

The ERT queue is used by PPU programs through the processing of two, CPUMTR functions, request device/track access (RDTM) and drop device/track access (DDTM). These functions effect the setting of the TRT access bits and the creation or deletion of ERT queue entries. When a job is scheduled for access to an equipment/track CPUMTR stimulates the addressed job to re-request and gain the requested access.

The job address may reference an input or rollout file (bit 24 = one) or a terminal user (bit 24 = zero). In the first case, CPUMTR raises the job priority, and the access is completed by a PPU program after the job reaches a control point. In the case of a terminal user, the terminal processor program (MESA) is notified via an interrupt flag in the terminal control block, and access is completed when MESA resumes processing for the terminal.

The machine number is used to distinguish the machine of residence of the addressed job. This field, in conjunction with the storage of the ERT and TRT's in ECS, is the basis of Dual MACE control of permanent files. The ERT is moved between ECS and CM in a block transfer. Its modification is regulated with an ECS flag register bit.

A periodic check of the ERT is performed by request of the PPU program IRS, repetitive system services. This eliminates the need for system intercommunication at the release of requested equipment/tracks - i.e., a released resource will be assigned to a waiting, ERT entry in the other machine at the time of the general check, rather than immediately after its release, as is the case in the same machine. The queue priority value is used to rank ERT entries and thus to prevent one machine from dominating access to a particular equipment/track.

PPU Program Library Directory - PLD

The PPU program library directory (PLD) indexes and locates the PPU programs available for system use. The main method of PLD access is the CPUMTR, search PPU library function (SPLM). The use and actions of that function have already been described in the discussion of PPU resident. The format of the PLD is described in that same section.

When an ECS LOAD dead-start is performed, the completed PLD is copied from CM to ECS by STL, provided that the system disk is a shared device. When a USE dead-start is performed, STL copies the disk resident and ECS resident entries of the ECS PLD to CM. The CM resident PLD entries are loaded from the dead-start tape to CM by STL.

The PLD entries may be altered by the use of a system library editor program, EDIT. EDIT, in turn uses the PPU program ESL. EDIT permits the replacement, insertion or aliasing of PPU programs. It also permits altering PPU program residence. During PLD changes the PLD is reserved by setting the sign bit of its first entry. When this bit is set, the CPUMTR function SPLM returns a repeat request to all callers.

System, CPU Program Library Directory - SLD

All control card called programs and their overlays are indexed and located by the system, CPU program library directory, SLD. These programs include compilers, assemblers, utility programs, overlays, etc. All programs are disk resident. Entries of the SLD have the format:

Bit	Contents
59	One
58-18	Seven character, display code name. The first character must be less than 5.
17-11	The first disk sector of the program.
10- 0	The first disk track (minus the 2'' track bit flag) of the program.

The SLD entries are not sorted. They are searched sequentially by the two, PPU program loaders, 2TS (translate control card statement) and LDR (program loader). The SLD is terminated by a zero word. Entries may be added or modified using the programs EDIT and ESL.

Dayfile Buffers

The dayfile buffers are used to contain dayfile messages issued for jobs and fc system logs (e.g., the error log). The buffers are described and managed with the use of the dayfile buffer pointers, located in low CM, preceding the EST. The PPU program IDD dumps the dayfile buffers to disk files.

During rollout and rollin the job dayfile buffer is copied between CM and the rollout file. This is necessary, since the dayfile buffers are associated with control points. Typically a job may use a number of different control points during its full execution, and a control point may be used for the partial processing of many different jobs during the time a job is rolled out.

When a job completes, its dayfile, including the dayfile buffer and what has been recorded on disk, are copied to the end of the job output file. Bytes two and three of the output file FST entry give the track and sector address of this dayfile copy.

Inter-machine Communication Buffers

Together with corresponding ECS areas, the inter-machine communications buffers permit the transmission of functions and replies between machines of the Dual MACE System. The buffers are located by a control point zero pointer word (XIBP, word 27_8).

Bits 0-16 give the CM address of the buffers. Bit 17 of XIBP is set if inter-machine communication is disabled. Inter-machine communication is also disabled if bits 0-16 are zero - i.e., no buffers were allocated by virtue of the dead-start type (no ECS table LOAD or USE).

The first two words addressed in XIBP are the "IN" and "OUT" buffer pointers. The "IN" pointer describes the buffers in which the machine receives requests from the other machine. The machine transmits function requests in the "OUT" buffers. The buffer allocations are reversed from machine to machine - i.e., the "IN" buffers of one machine are the "OUT" buffers of the other. The two pointer words have the format:

Word	Bits	Contents
1	41-24	CM address of the "OUT" buffers
	23- 0	ECS address of the "OUT" buffers
2	41-24	CM address of the "IN" buffers
	23- 0	ECS address of the "IN" buffers

These pointer words are followed by two sets of eight word blocks, the "IN" and "OUT" blocks. Each block has a single, fixed format header word, and seven words used for transmitting function dependent parameters and replies. The header word has the format:

Bits	Contents
59	Set if the block is active
58	Set if a reply is waiting to be transferred
57	Set if function processing is in progress
56-48	Assignment status (PP output register address)
47-36	Address of assigned job (FST address or 2*(TCB number plus one).
17- 0	Function or reply status

The function blocks are transmitted between CM and ECS by MTR and CPUMTR, based on the status of the header word fields. The PPU programs IRS and ICP initiate function requests and form replies.

CPU Monitor (CPUMTR) Code

The relocatable code of CPUMTR is stored in central memory during dead-start by the PPU program SET. SET reads the relocatable code from the dead-start tape stores it in CM, relocates all addresses to their correct values and starts the CPUMTR preset routine.

In addition to normal, start-up activities, CPUMTR also constructs control point zero pointers to a number of system tables stored within CPUMTR. These tables include the mass storage allocation table, file blocks, the PPU delay stack, the channel reservation table, the error log table, the flag register reservation table, and the CPU request queue.

Mass Storage Allocation Table

The mass storage allocation table is used by the CPUMTR function request mass storage space (RMSM) for the assignment of disk space to files. The one word entries of this table have the format:

Bits	Contents
59	If set the device is not available
17-0	Channel

There are eight entries, one corresponding to each of the eight possible allocatable disks, equipments zero through seven. A zero entry signifies that no device is defined.

File Blocks

Each terminal control, MESA control point is allocated a block of 30_8 files in the FNT/FST table. All files used by a terminal controlled by MESA are allocated to the file block area. When a terminal is interrupted, and its control information is swapped to ECS, the entire 30_8 entry file block is also swapped. A free entry in the file block has the format:

Word	Bits	Contents
1	59-54	Zero
	53-48	16_8
	47-12	Zeroes
	11- 6	16_8
	5	One
	4- 0	Zero
2	59- 0	Zeroes

CPUMTR maintains a file block control table. Its address is stored in control point zero word 45_8 (FBTP), bits 36-53. The file block control table contains 26 entries, one for each possible control point. Each entry has the format:

Bits	Contents
59-48	FNT address of block assigned to control point
47-36	FNT limit of block
35-24	Free entry indicator (e.g., 0016_8)
11- 0	Free block address

A free block may be addressed in any control point table word. It is signified by an address in bits 0-11 and zeroes in bits 48-59. When a block is assigned to a control point, the word for that control point contains the block address (non-zero) in bits 48-59 and zeroes in bits 0-11.

The PPU Delay Stack

The PPU delay stack contains the input registers and recall times of PPU programs which have voluntarily surrendered execution for a fixed time interval. Bits 0-17 of control point zero word 42_8 (PPUP) carry the address of the PPU delay stack. Bits 0-59 of control point zero word 43_8 (PRTL) indicate the time at which the next entry of the stack is to be reactivated. PPU monitor uses PRTL to stimulate CPUMTR to activate that entry.

The PPU delay stack consists of two, forward linked lists. CPUMTR carries two pointer words, one pointing to the top active entry; the other, the top free entry. The delay stack is empty when the active pointer is zero; full when the free pointer is zero. Each delay stack entry consists of two words of the following format:

Word	Bits	Contents
1	18-59	Time of recall in milliseconds times four
	17- 0	Forward link (zero if end of chain)
2	59- 0	The PPU input register

CPUMTR may also place entries in the PPU delay stack when a CPU program requests access to a PPU program (via an RA+1 or XJ call) and there is no pool PPU available. In that case, CPUMTR adjusts the CPU status of the job to reflect the status of the PPU call (e.g., "R" rather than "I") and enters the PPU with a time of recall equal to the current time.

Channel Reservation Table

All hardware and software channel control passes through CPUMTR. A channel table, addressed in bits 0-17 of control point zero word 45_8 (CHTP), reflects channel table status. Each hardware or software channel has the one word format:

Bits	Contents
59	Set if a request is pending for an assigned channel
11- 0	The output register address of the PPU to which the channel is assigned

A channel is unassigned if the entire word is zero.

When CPUMTR detects a request for a reserved channel, it sets bit 59 of the channel table word. Depending upon its processing mode, the PPU program to which the channel is assigned may occasionally test bit 59, and release the channel if the bit is set. PPU programs of the BATCHIO, card-reader, line-printer, card-punch driver subsystem share the device channel in this fashion. When a channel is dropped CPUMTR assigns it to the next PPU in round-robbin fashion, thus evenly distributing channel access.

There are twelve hardware channels, $0\text{-}13_8$, and two software channels which are allocated entries in the channel table. Software channel 14_8 (TBCT) is used to reserve portions of the terminal control block table (TCB). Software channel 15_8 is used to reserve the FNT/FST table for manipulation of queued files. Non-queued files are managed with the set file busy (SFBM), CPUMTR function.

The Flag Register Reservation Table

A PPU may reserve and clear flag register bits with the ECS transfer (ECSM), CPUMTR function. The flag register reservation table contains one word for each PPU, indicating the flag register bits reserved for the PPU. The table is addressed in bits 18-35 of control point zero word 27_8 (XFRP).

When a PPU program reserves flag register bits, the reservation is noted in the table. When it clears the reservation, the table bits are cleared. If a PPU program drops usage of the PPU (terminates or enters delayed recall) without clearing the reserved bits, CPUMTR uses the table entry to automatically clear the reserved bits.

The CPU Request Queue

The assignment of a CPU to a control point program is controlled by CPUMTR using a CPU request queue. The CPU request queue consists of three tables, a linked, control point list and two pointer tables.

Each pointer table contains an entry for each possible CPU priority, zero through four. There is a head pointer table and a tail pointer table. A given queue is empty if the pointers are zero. The control point, linked

list contains one word for each control point, including the CPUMTR problem mode control point, N+1. Control point entries of equal priority are linked together. The pointer tables have the same format - bits 0-17 contain the address of the head or tail entry in the control point list. Entries of the control point list have the format:

Bits	Contents
59-42	Control point address
35-18	CPU priority
17- 0	Link to next entry

When the CPU is requested for assignment to a control point (e.g., by a PPU program), the CPU status of the control point is set to 'W' and the control point is entered in the proper queue. CPUMTR selects the next control point for assignment by searching the queues in order from largest to smallest, stopping at the priority of the current CPU user or zero, whichever applies.

The Installation Area

The installation area was originally designated by the MACE designers to provide space for installation dependent tables. As the structure of the Purdue system has evolved and grown separate from a "standard" MACE system (now called KRONOS) the original meaning of the title has lost all significance. A more appropriate title for this area should be indirect tables, since the area is devoted to tables whose addresses are formed by adding a displacement value to the address of the installation area. That address is located in control point zero word 12_8 (INSP), bits 36-53.

The tables of the installation area include, in order, the disk pack pointer table, the IRS table, the system status line, the device request blocks, the permanent file device table, the queue priority weight tables, the job scheduler, category tables, and the stack processor I/O stacks.

The Disk Pack Pointer Table - DPT

The disk pack pointer table (DPT) is used to control the mounting and usage of private, 854 (RD) permanent file packs. It contains six (NDPC) entries and starts at displacement zero (DPTI) in the installation area. Entries of this table, whose usage is controlled by the reservation of channel 15_8 (FNCT) have the format:

Bits	Contents
59-18	Pack name
11- 6	Equipment number of RD device on which the pack is mounted
5	If set, the pack has been logically dismounted
4- 0	A count of the number of jobs using the pack

The IRS Table

The system repetitive services PPU program, IRS, uses a four (RSTC) word area to store the times which regulate the activities of IRS. The table begins at displacement 61 (RSTI). It has the format:

Bits	Contents
239-216	Time one
215-192	Time two
.	
.	
.	
23- 0	Time ten

The System Status Line

The dynamic display driver, DSD, maintains a set of basic system displays at the top of each of the two display screens of the system console. Among the items displayed at the top of the left screen is a line which describes the status of various system flags, counts and switches [8]. This line is constructed by IRS once every four seconds and stored in the installation area in three words, displaced 65_8 (SSLI) words.

The Device Request Block Table - DRB

During the time that a user process is waiting for an operator to locate a medium (e.g., a reel of tape), mount it, and associate its equipment to the job, it is most economical of system resources to queue the job on a mass storage device. Only as much information as is necessary to identify the medium, the job and the file to be assigned need be kept in central memory.

The entries of the DRB table perform this function. Each four word entry contains space for linking a job to a file/equipment request, and receiving a mount indication from the operator. DRB's may be linked together to process parallel requests, (e.g., two tapes).

The DRB contains 16 (NRBC), four word (LRBC) entries. It is displaced 154_8 (DRBI) words from the beginning of the installation area. The DRB is associated with the requesting job by the address of the job entry in the file status table. It is associated with the equipment by the equipment ordinal, and the equipment in turn is associated with the DRB in the EST entry.

The contents of the DRB's may be displayed on one screen of the system display console. The DRB display informs the operator of requests, and through the display driver program, DSD, the operator keys entries to indicate the assignment of requested devices.

In the case of parallel requests, the job is queued (rolled-out) until all requests have been assigned. At the entry of the last assignment through the keyboard, DSD makes the job eligible for processing again (roll-in).

The Permanent File Device Table

The permanent file device table contains an entry for each fixed device which contains permanent files. It contains a maximum of 12_8 (NPFC) entries, one word per entry. It is displaced 254_8 (PFTI) words from the beginning of the installation area. Each non-zero entry has the format:

Bits	Contents
59-48	Device equipment number
47-36	First track of device directory
35-24	Number of sectors per directory

The Queue Priority Weight Tables

Job execution queue priorities are periodically re-evaluated by various PPU program job processors - the job scheduler, the job rollout program, etc. These programs use a common routine "CQP" to compute the twelve bit queue priority. CQP, in turn uses a 26_8 (QPWC) word table, stored in the installation area at a displacement of 330_8 (QPWI) words to compute the queue priority. This table has the following format:

Word	Bits	Contents
1	59-48	Zero when the table is not defined
	47-36	"First-pass" limit
	35-24	Minimum queue priority
	23-12	Maximum queue priority
2	59-54	Parameter index
	53-48	Test index
	47-36	Increment value
	35-24	Shift instruction
	23-12	Test value
.		Additional, 48 bit fields
.		of the above format
.		

The parameter index, test index, increment and shift value combine to describe the manner in which a job parameter (e.g., CPU time used) is to be tested against the test value (e.g., greater than ten seconds), and the action to take if the test is true - add the increment and the "shifted" (e.g., times 1/2, 2, etc.) absolute difference |test-value-parameter| to the queue priority. The use of the queue priority tables is fully described in [9].

Job Scheduler Category Tables

The job scheduler category tables control the manner in which the system job scheduler PPU program, ISJ, assigns jobs to control points for execution. These tables occupy 15 words of the installation area, starting at displacement 357_8 (JSCI). The tables have the format:

Word	Bits	Contents
1	59-48	Zero if the tables are not defined
2- 3	119- 0	Ten queue priorities, terminated by a zero byte
4- 5	119- 0	Nine job limit bytes
6- 7	119- 0	Nine candidate table limits
8- 9	119- 0	Nine FL limits
10-11	119- 0	Nine control point counts
12-13	119- 0	Nine candidate counts
14-15	119- 0	Nine FL usage values

The fourteen words, 2-15, form a nine by seven array of twelve bit values. The first column defines nine queue priority ranges. The remaining six columns control or describe scheduler activities related to the queue priority classes.

Briefly, the job scheduler will consider the specified number of candidates in each category for execution. It will schedule no more than the specified number of control points and FL to the category. Columns 4-7 of the array carry the current category statuses for DSD display (the "W" display).

Terminal Control Blocks - TCB's

The terminal control blocks manage all remote I/O activity. The TCB's are manipulated by the inter machine stack processor PPU program IIM, and by CPUMTR. The three word TCB has the format:

Word	Bits	Contents
1	59-48	Device mode flags - on, off, etc.
	47-36	Alias - a link of one TCB to another
	35-24	Device physical status - e.g., end-of-reel, etc.
	23- 0	Logical status flags
2		Swapping controls
	59	Word interlock
	58-54	Page residence - ECS, active, etc.
	53-48	MESA processor index (e.g., PIRATE, ALFIE, etc.)
	47-36	Swap cause - I/O, time, etc. - and page sector size (if on disk)
	35-24	MESA directive function
	23- 0	Page residence address in ECS or on disk
3	59	Class ID flag for batch devices
	58-42	User ID or class, ID routing code
	41-36	Logical interrupts pending - message, etc.
	35-24	Interrupt enable mask
	23-12	Alternate TCB user job FST address
	11- 0	Current TCB user job FST address

The Input/Output (I/O) Stacks

Control of input/output devices which are clustered on single channels - the 604 tape units and the PROCSY 2.0 front end processors - is managed using I/O stack processors. The tapes are managed by ITF and the front end processors are controlled by IIM. An I/O stack area is reserved for each control point area, in which requests for I/O activity are placed, and from which the stack processors select operations to be performed.

Each I/O stack is eight (JLDC) words long. The first I/O stack is displaced 400_8 words (JSCI) from the beginning of the installation area. There are N+2 stacks, one for each control point, 1- N, and two for control point zero and control point N+1. An I/O stack has the format:

Word	Bits	Contents
1	59-58	A two
	57-48	Entry count
	47	Buffer activity flag
	24	Stack processing suspended if set
	23-18	Suspended operation count
	17- 0	Active operation
2-7		Stack entries

There are a number of different stack entry formats, depending upon the processor, function, and processing status involved. A normal, I/O operation entry has the format:

Bits	Contents
59	Set if entry active
58	Set if entry suspended pending the completion of other activity (e.g., the rewinding of a tape)
51	Operation continued, if set
50	Operator pause in effect (e.g., after a parity error failure)
49	End of reel, if set
48	Parity error, if set
47-42	Processor dependent
41-36	Processor index (1TF or 11M)
35-18	Operation count
17	File operation, if set
16- 0	File FET address in job FL

Other entry formats describe special (non I/O) request, suspended operation, and special system request processing.

The MESA Library Directory - MLD

The MESA library directory, MLD, indexes and locates programs of the MESA, CPU program system. A special library is required because of the unique structure of MESA. Briefly the structure is such that every processor is actually an overlay to MESA. When any processor is requested (e.g., via a control card) MESA is loaded and in turn loads the required overlay. The MLD has the format:

Word	Bits	Contents
0	54-48	Disk equipment for MESA
	47-37	Disk track for MESA minus bit 2''
	36-30	Disk sector for MESA
1	59-48	Program, ECS track in equipment DX
	47-37	Program disk track, minus bit 2''
	36-30	Program disk sector
	29-18	Number of entry points
	17- 0	Link to next main program
2	59-18	Main program name
	17- 0	Usage count
3-m		Alternate entry point names
n+1	59-48	Segment ECS track
	47-37	Segment disk track
	36-30	Segment disk sector
n+2	59-18	Segment name
	17- 0	Usage count

The MLD is organized such that an entry for a main program (e.g., PIRATE) is always followed by entries for its overlays or segments, if any. The linkages of the MLD are set such that a processor, searching for a main program, can skip from one main program entry to the next without examining the segment entries. In addition, the position of each main program and segment are translated into six bit program indices which are stored in the TCB and the MESA, data page. These indices assist in the efficient reloading of code segments when a terminal is swapped in for execution.

Special TRT's

A number of special TRT's are used in the Dual MACE system to enable control of resources through TRT-like structures. These include DU, small permanent file (PFILES) control, DX, ECS page space control; DY, DRB reservation control; and DZ, single device, mountable equipment control.

Small Permanent Files TRT - DU

The PFILES, permanent file subsystem provides an efficient method of managing permanent file storage, by artificially segmenting one or more extremely large permanent files (up to 131,072, 640 character sectors) into

a large number of user oriented sections. Each section is associated with a device called a user data block (UDB) which is uniquely identified by user assigned parameters - account code, user identifier, and password.

To the user, each UDB section appears as a fully indexed file storage system. To the operating system, it is a number of randomly distributed data sectors stored within one or more large files. PFILES provides user access to information stored in the system by copying data between the large file and local files via "GET"s and "PUT"s.

Access to an individual, UDB section is controlled by a three character (A through 5) code and the DU, TRT. PFILES accesses a UDB with a standard, RDTM call to CPUMTR. The equipment type causes CPUMTR to process the function in a special fashion. The TRT for DU is considered a list, containing one access per word. A word takes the form:

Bits	Value
59-36	The three character identifier
35-24	The job address
11- 4	Permanent file access flags
3- 0	17_8 (all tracks reserved)

CPUMTR searches the TRT for the indicated UDB. If none is found, a new entry is made, and the permanent file access flag is set for track byte zero. If a reserved UDB is located, the caller is placed in the ERT, queued for access to the reserved equipment track. Track 4003_8 is used when the TRT fills: the caller is queued on access to 4003_8, and the next PFILES process which releases a UDB, tests the 4003_8 queue, and releases access to 4003_8, thus stimulating the PFILES calls which require UDB space.

ECS Swap Space TRT - DX

ECS swap space is allocated in this TRT-like table. Each track represents 200_8 words of ECS displaced from the beginning of a fixed ECS address (stored in the EST entry) by an amount equal to bits 0-10 of the track number, times 200_8. Thus the DX equipment allocates 4000_8 times 200_8 words of ECS - 1000000_8 or $262,144_{10}$.

One DX track is linked to another via the linkage byte feature of the TRT. The last track lacks the 2" bit in its link, the link value instead indicating the number of words less than 200_8 allocated at the end of the chain.

In order to improve the speed of access to these relatively small units of ECS space, tracks are allocated from a free chain, rather than with the standard TRT management facilities. This allocation function is available to the MESA control point programs through special XJ function calls. PPU programs may access the functions with the ECS paging services function (XSPM) to CPUMTR.

The DRB Reservation TRT - DY

The DY TRT is used to queue requests for the mounting of a private, 854 (RD) pack behind a single device request block (DRB). Each of the sixteen tracks of this TRT represents a DRB. When the permanent file processing PPU program, PFM, detects that a request for the mounting of a pack is already receiving attention via a DRB, it queues the requestor in the equipment reservation table (ERT) using the DY equipment, and the corresponding DRB track.

When the pack is mounted and assigned the original requestor drops access to the DY track and the other requestors are rolled in to share access to the pack. If access cannot be shared, the requestors are then queued in the ERT for access to the equipment and track of the mounted pack.

Single Device, Mountable Equipment Control - DZ

The queueing of requests for all single device, mountable equipments is managed with the TRT of equipment DZ. These devices include paper tape equipment, plotters, film scanners, etc. Only one device of each type exists.

Each device is assigned a track in equipment DZ corresponding to the 11 bit equipment code - e.g., PR (2022), SC (2303), etc. Requests for the equipment are queued in the equipment reservation table (ERT) on equipment DZ, and the equipment code track. The DZ TRT is actually not used. It is allocated one word for consistency with all other TRT's and CPUMTR handles DZ requests in a special fashion not involving the TRT.

Resident PPU Library - RPL

Central memory resident PPU programs are stored in the resident peripheral library (RPL). These programs are generally located in CM because of their access frequency. They can be moved to PPU memory with a single, block read.

Each program of the RPL begins with the standard PPU program header word of the format:

Bits	Contents
59-42	Program name
35-24	PPU load address
23-12	Optional checksum, used for DSD overlays
11- 0	Central memory word length

An extra word separates each program of the RPL. This word is allocated in order to make the loading of CM and ECS programs the same operation for the PLL subroutine of PPU resident. In that operation the PPU reads a CM block using a word count and address supplied in the SPLM, CPUMTR function reply. The PPU then sets the next CM word non-zero. The latter operation is a signal to CPUMTR that the CM segment used for reading the PPU program from ECS is free for other use. The write operation has no significance when the PPU program is in the RPL.

User Program Space

The remainder of central memory is allocated to user programs, assigned to control points. The amount and location of the memory allocated to each control point is indicated in the field length (FL) and reference address (RA) stored at each control point and in the exchange packages (areas). PPU monitor manages this space, moving it about as required with CPUMTR assistance and by manipulating RA values to reflect these movements. Memory movement is performed by CPUMTR in problem mode at control point N+1.

RA

A basic structure is assigned to the first 100_8 words of the memory assigned to each control point. This area is sometimes called the "SYSTEM" area. The word at (RA) plus zero, also called RA, is used to receive system information such as sense switch entries, etc., and for exit mode control.

When an automatic CPU error exit is permitted to occur (as defined by EM, the exit mode register) the CPU stores exit information in RA and stops with the instruction counter, P, set to zero. MTR and CPUMTR can detect this condition and initiate error processing.

RA+1

The location (RA) plus one, RA+1, is used by a control point, CPU program to request system activity. The program stores the request in RA+1, and RA+1 is cleared when request processing has been activated (no auto-recall), or completed (auto-recall). This function can also be performed far more efficiently with an exchange jump (XJ) function request.

The RA+1 call takes the form:

Bits	Value
59-42	CPUMTR request or PPU program name
40	Auto-recall request, if set
35- 0	Request parameters

CPUMTR recognizes a number of special requests and performs immediate processing on them. These include:

ABT	End the CPU and set the CPU abort error flag
CCR	Conditional CPU recall
CIO	Intercept stack processed I/O calls. All others are passed to the PPU program CIO.
END	End the CPU
RCL	Recall the CPU (see the X status discussion)
MSG	Process a message
TIM	Return time value

All other values in bits 59-42 constitute PPU program calls. The program name must begin with an alphabetic character. CPUMTR assigns a PPU to execute the program by replacing bits 36-41 of the RA+1 word with the control point number and placing the word in the input register of an available pool PPU. If none are available, the entry is placed in the PPU delay stack. If the delay stack is filled, or if there are already four entries in the stack for the control point, the control point CPU status is set to X and RA+1 is left unchanged. For an XJ call, the only variation is that the call word may come from RA+1, any word in the central point area, or any X register of the control point exchange package.

When the auto recall request bit (40) is set, the CPU is not returned to the control point until the PPU program has completed. RA+1 is used to contain the program location - PPU output register address or delay stack address.

A call to the PPU program CIO receives special treatment. CPUMTR locates the FNT/FST entry referenced by the controlling FET (its address is in bits 17-0 of the CIO call). If the file is assigned to a stack-processed equipment - MI, MT, SC, TH, TL, TU - CPUMTR makes a direct stack entry and activates the appropriate stack processor. No PPU is assigned to the CIO program call. If the equipment is not stack processed (or no FNT/FST is located) a PPU is assigned to execute CIO.

RA+2 - RA+77_8

The remainder of the system area is used to convey program load information. Words RA+2 through RA+63_8 are a parameter list area. Word 64_8 is used for program name and parameter count. Words 65_8, 66_8, and 67_8 contain various load addresses - e.g., the last program space address, loader bits, etc. Words 70_8 through 77_8 contain the control card which caused the program load.

The remaining assigned FL is available for any legitimate program usage - code, tables, buffers, etc. The program may also use RA+2 through RA+77_8 if required. RA may be used, but is vulnerable to entry of system flags, as already noted.

Files

A file in the Dual MACE System is a structured information set. Files are used in almost every aspect of system processing. An understanding of their structure and processing is an important key to an understanding of the manner in which Dual MACE assists in the management of computational tasks (jobs).

The basic unit of a file is the central memory (CM) sixty bit word. CM words may be grouped into coded (character) images or binary, random length units. These word groups may be grouped into records, and the records into files.

A coded (character) image is a character string in CDC Display Code, six bits per character. It may contain a variable number of CM words. Imbedded zeroes (six bits) are avoided. Usually the characters occur in multiples of two, and trailing blanks (spaces) are removed. The last word in the string contains zeroes in bits 0-11 of the word. It may be entirely zero (bits 0-59) or consist of zero, right fill in bits 12 through 54.

Binary data may be found in variable length units. In a large number of cases, table header words specify the type and length of the binary data [10]. In all cases, the first level structure of binary data is controlled by its content or by the programs processing the data.

CM words, in either coded or binary form, may be organized into records. A record is signified by the presence of an end-of-record (EOR) flag. A record may be null - i.e., contain no data words. The EOR flag may vary with the medium. On disk it is a sector with a control byte word count less than 64. On cards, it is supplied by a separate card with rows seven, eight and nine punched in column one.

CM words, with or without EOR organization, may also be structured into files. Again, a special separator is used - the end-of-file (EOF) flag. An EOF flag may terminate a null amount of data. Usually it follows an EOR, thus signalling the end of a group of records. On disk, the EOF flag is a sector with a zero link control byte, and a non-zero word count control byte, which contains the forward linkage usually contained in the link control byte. On cards, the EOR is supplied by a separate card, whose first column contains punches in rows six, seven and nine.

An information set, or file, is the entire collection of these groups - words, records, and files. The entire information set is terminated with an end-of-information flag (EOI). On disk, the EOI signal is a sector both of whose control bytes are zero. The EOI card is one with six, seven, eight and nine punches in column one and none in column two.

Writing and Reading Files

The writing and reading of files is a PPU task, since only the PPU's have access to the data channels. The type of PPU program used to process a file, and the manner in which the program is invoked, depend to a large extent on the medium on which the file is recorded. The PPU resident area contains provision for disk file processing. The processing of other media (e.g., magnetic tape), is usually left to specific programs (e.g., the tape stack processor PPU program, ITF), or in some special cases, to code local to the PPU program. In the latter case, programs of the dead-start sequence - DSS, SET, STL - contain tape input processing code for reading the dead-start tape.

Each PPU program which uses the resident disk drivers to write or read a file must follow the general rules for specifying the file organization in the sector control bytes. As each sector is formed, the PPU program stores the file linkage and sector length information in bytes one and two of the 322 byte sector buffer in PPU memory [(64 CM data words times five) plus two control bytes]. The PPU program supplies the buffer address and disk address (logical track and logical sector) to the disk driver, which records the 322 bytes at the addressed location.

The PPU program which writes a file using the resident disk driver must also contend with space reservation and disk positioning. Space is reserved using the request track (RTKM) function of CPUMTR. As the file lengthens, track by track, the sector linkage bytes and the TRT linkage bytes both reflect the forward file linkage. In addition, each time the PPU program records a new EOI sector, it must update the last TRT linkage byte, using the partial drop option of the drop track (DTKM) CPUMTR function. Before using the disk driver to record the first sector of a new logical track, the PPU program must execute the disk driver subroutine POS, in order to insure that the disk is properly positioned.

The resident disk driver may also be used to read sectors of a file. In that case, the driver moves 322 bytes from the addressed disk location to the specified PPU memory buffer. The calling PPU program then uses the control byte contents to discover the organization of the file, sector by sector.

Direct, PPU to disk and disk to PPU file processing is used for both system and CPU- program input/output. The system I/O functions performed cover a wide range of activities - CPU program loading, rollout/rollin, various levels of job control, and, of course, CPU program I/O services.

CPU Program File Processing - The FET and CIO

The processing of files for CPU programs is directed by a central memory table, resident in the job field length, called the file environment table (FET). The FET contains a wide variety of parameters [11] describing all aspects of file manipulation. Basically, however, the FET describes to the I/O processing PPU program the name of the file, the type of I/O processing required (function), and a description of the CM data buffer (location and length).

The PPU program CIO (central input and output) performs almost all CPU program file processing. The CPU program employs CIO in a three step process.

1. Data is entered in the CM buffer (write), and the buffer pointers are positioned to indicate buffer status.

2. The CPU program stores the file name and I/O processing type in the FET. The processing type is specified by an even numbered function code.

3. The CPU program calls CIO via the RA+1 or XJ request mechanisms. When CIO has performed the requested operation it replies by moving the buffer pointers and setting the function code odd. CIO may also return flags to indicate EOR, EOF and EOI status conditions (read).

The FET location is transmitted to CIO in bits 0-17 of the PPU call. The FET has the following general form:

Word	Bits	Value
1	59-18	File name
	17- 0	Function code and status reply
2	17- 0	The address of the first buffer word - FIRST
3	17- 0	The address of the next input word - IN
4	17- 0	The address of the next output word - OUT
5	59-48	Last known FNT address
	17- 0	The address of the last buffer word plus one - LIMIT

The FET contains a number of other parameters, defining additional words used for random file processing or other special file operations (e.g., random positioning).

The FET describes the transfer of data to and from the file. The residence, position and current status of the file are described by a two word file name table (FNT) and file status table (FST) entry. The FET and

FNT/FST entries are linked together by the file name, and the control point to which the CPU program (hence the file) are assigned. In order to facilitate the location of the FNT entry at each CIO call, the FET carries the last known address. Since the FNT address can change as the job is rolled out and in, CIO uses the last known address as an estimate only.

The buffer pointers describe a CM block which is used for data transfer in a circular fashion. Data is input to the buffer, starting at IN and ending at OUT minus one. Data is output from the buffer starting at OUT and ending at IN. Whenever either pointer reaches LIMIT, it is reset to FIRST.

A CPU program processes input data by removing it at OUT, and signifies that a word has been processed by advancing OUT until it reaches IN. If OUT reaches LIMIT, it is reset to FIRST. When IN equals OUT, the buffer is empty, hence the file input processors terminate at IN minus one. A CPU program enters output data at location IN, and then advances IN to indicate the presence of data. Again, the CPU program avoids setting IN equal to OUT (empty buffer) when placing data in the buffer.

The function codes which direct the activity of CIO have some general characteristics. The function codes are even numbers, twelve bits in length. Bit zero is the busy (zero) - not-busy (one) flag. Bit one indicates the mode for special media such as tape - zero for coded, one for binary. Bit two is clear for read operations and set for write operations. Function numbers less than 40_8 are read/write functions.

When CIO replies to a read function request, it uses three bits of the function field to indicate the presence of file organization flags encountered while transferring data. Bits three and four are set to 10_2 to indicate an EOR; to 11_2, an EOF. Bit nine is set to a one, and bits three and four to 11_2 to indicate an EOI.

The values contained in these twelve function field bits form the last code and status field of the FST entry. Bit zero becomes the FST word reservation control (busy if zero, not busy if one). The remaining bits reflect the last I/O function performed on the file, and its resulting read status (EOI, EOR, EOF).

CIO performs very few functions itself. Instead, it serves as a housekeeper for specialized device driver overlays. At start-up, CIO verifies the FET contents, locates the FNT/FST entry for the file, and establishes various pointers to the FET and the buffer. Then CIO loads the appropriate device driver, as indicated by the function, the FST equipment number, and the equipment code of the EST entry. The driver performs the I/O operation, updates the FET buffer pointers and returns to CIO. The CIO clean-up consists of releasing access to the FST entry, setting the FET function complete, and updating the job, I/O transfer count. All communication between CIO and the drivers takes place via direct cells.

FNT/FST Entry Management

One of the most important start-up activities of CIO is the location and reservation of the FNT/FST entry of the file. In the process of performing this task, CIO makes use of CPUMTR and two PPU overlays. The CPUMTR function is set file busy (SFBM). This function permits CIO to request a search of the FNT for the specified file, and its reservation, if possible. CIO supplies the file name to CPUMTR; the SFBM function response returns the file status (reserved or not) and FST address, if reserved.

If CPUMTR cannot locate the file, CIO loads its service overlay, 2CS. The linkage between CIO and 2CS is so constructed that the overlay 2CS is loaded the first time that CIO references any subroutine of 2CS. Subsequent calls to 2CS subroutines are linked directly to 2CS. The routines contained in 2CS are those required on an infrequent basis (once in every twenty CIO calls on the average). Their location and method of reference in 2CS greatly reduces CIO load time.

2CS contains a subroutine for creating a new FNT/FST entry. CIO uses this subroutine when the SFBM reply indicates that no entry exists. For certain functions (e.g., the releasing of an FNT/FST entry) for which the creation of an entry would be redundant, 2CS simply returns to CIO. When an entry must be created, 2CS loads the location-free (zero-level) routine ØBF, begin file. 2CS supplies the file name and special parameters to ØBF in direct cells.

Entering New Files - ØBF

ØBF searches the FNT/FST table for an open slot (zero FNT word) and creates the requested entry. If a file block is assigned to the control point of the CIO user, ØBF locates the file within the 30_8 entry file block assigned to the control point. In the latter case, the free entry is signified by an FNT entry with 0016_8 in byte zero, zeroes in bytes one, two and three, and 1640_8 in byte four.

ØBF sets the file type in the FNT word to 3 (local) unless the file is named INPUT (type = 0), OUTPUT (type = 1), PUNCH (type = 4), PUNCHB (type = 5), PUNCH8 or P8 (type = 6), or ROLOUT (type = 10_8). ØBF will request a track for the file if the equipment is specified in the calling parameters, or if the file type and mass storage allocation word correspond. The mass storage allocation word (MSAL) is stored at location 173_8 in control point zero. In can be used to direct all files of a specific type to a specific mass storage device. It has the format:

Bits	Value
59-48	Local (3) file equipment in bits 48-53, if the byte is non-zero.
47-36	Input (0) file equipment in bits 36-41, if the byte is non-zero.
35-24	Output (1) file equipment in bits 24-29, if the byte is non-zero.
23-12	Rollout (10_8) file equipment in bits 12-17, if the byte is non-zero.

ØBF interlocks access to the FNT/FST using the software FNT channel, 15_8 (FNCT).

Releasing Files - ØDF

CIO uses another location-free (zero-level) overlay to release files - ØDF, drop file. Again, CIO passes parameters to ØDF in direct cells. The processing of ØDF is regulated by the file type of the FNT entry. According to the file type, ØDF will drop disk tracks, invoke the permanent file update PPU program IPF, and initiate special device release processing for tapes, terminals, etc. ØDF restores file block entries to their special, non-zero, free status.

Device Drivers

CIO passes FET and FNT/FST values to the device drivers in the following direct locations:

Location(s)	Value
FS-FS+4 (20_8-24_8)	The FST entry word
RB (25_8)	Random file flag bit
PF (27_8)	Permanent file flag
ES-ES+4 (30_8-34_8)	The equipment EST entry
TC-TC+1 (36_8-37_8)	The I/O transfer count
FN-FN+4 (40_8-44_8)	FET word one (file name, etc.)
LS (45_8)	Last FST status
OF (46_8)	The file type
FI (47_8)	The job input file FST address
FA (57_8)	The file FST address
FT-FT+1 (60_8-61_8)	The FET FIRST pointer
IN-IN+1 (62_8-63_8)	The FET IN pointer
OT-OT+1 (64_8-65_8)	The FET OUT pointer
LM-LM+1 (66_8-67_8)	The FET LIMIT pointer

CIO uses the following device driver overlays:

2DS - system console
2EP - electrostatic printer
2LP - 501 line printer
2PC - 415 card punch
2PD - Random file positioning
2RC - 405 card reader
2RD - read disk
2WD - write disk
2GD - 252 graphics console
2PR - paper tape reader
2PP - paper tape punch

When a driver has completed its processing, it returns to CIO with the I/O transfer count in TC-TC+1 advanced by the I/O amount processed. CIO completes the FNT/FST entry and the FET, updates the control point I/O transfer count, releases the PPU and returns to PPR.

Stack Processing

A number of I/O devices are served by stack processors. These devices are identified by their equipment type (bits 12-22 of the EST entry). For these devices, CPUMTR and other PPU programs create stack entries directly (CPUMTR) or with the request stack processing (RSPM), CPUMTR function. The equipments processed in this manner include:

MI - 7291V tapes
MT - 604 tapes
SC - film scanner
TL,TH,TU - PROCSY 2.0, front-end processors

On a periodic basis CPUMTR activates the stack processor PPU programs. They, in turn, scan all control point I/O stacks for processable requests. As each I/O function is completed, the stack processors update the FET, the FNT/FST entry and I/O transfer count in exactly the same fashion as CIO. The singular advantage of the stack processors is that they maximize channel usage, while minimizing the PPU capacity required to utilize the channel. There are currently two I/O stack processors in Dual MACE:

ITF - 604(MT) tape processing
IIM - inter-machine processing
for MI,SC,TL,TH and TU
devices

Error Processing

A number of errors may be detected during the processing of a CIO request. CIO may detect errors in the FET. A device driver may detect errors during its processing. Finally, CIO may detect an error after a return from a device driver. In these cases, CIO stores an error number in direct cell 46_8 (EN) and calls its error processor, IEP.

CIO calls IEP by rewriting the CM, PPU input register word with the IEP program call and returning to PPR. PPR then causes IEP to be loaded without destroying any of the direct cell values stored by CIO. IEP uses these values to issue error messages, FET and FNT/FST dumps, and to conclude processing of the erroneous request.

Random File Processing

CIO will perform random positioning for disk resident files. Random operations are directed by FET parameters. These include:

FET Word	Bit(s)	Value
2	47	Random file flag bit (one)
7	59-30	Random return request
7	29- 0	Random request

In addition to these values, the FET length parameter (bits 18-23 of word 2) must be large enough to contain the random operation parameter words.

When these parameters indicate a legitimate, random file request, CIO uses the position disk, 2PD, overlay to perform actual positioning. 2PD uses the direct cell values supplied by CIO, as well as the file linkage of the TRT to perform positioning. The basic random address is a sector ordinal, relative to the beginning of the file. Thus to locate a given address, 2PD threads the TRT chain until it has reached the requested sector ordinal. 2PD returns the logical track and sector values in the FST entry direct cells FS+2 and FS+3.

The random request field (bits 0-29 of FET word seven) are used in different fashions for reading and writing. When reading, the field always contains a sector ordinal with bit 17 set. When writing, it contains a sector ordinal in bits 0-16 if bit 17 is set. This is a random re-write or write-in place request.

If bit 17 of a write, random request field is clear, bits 0-16 contain a response word address. This is a file extend request, directing CIO/ 2PD/2WD to add the addressed data to the specified file, and to return the sector ordinal of the data in the response word. The response word is thus part of the random file index or directory.

2PD may also be called upon to perform a number of random-like operations. These include mass storage skipping (records or files) and backspacing (coded images, records, or files). CIO calls 2PD for these operations as directed by the FET function code. In this case, when 2PD returns to CIO, the operation is complete. When CIO calls 2PD prior to a read or write disk operation, CIO calls 2RD or 2WD after 2PD returns, provided that 2PD was able to locate the requested random address. Otherwise, CIO calls 1EP.

CIO can also be requested to return a random address (sector ordinal) without the use of 2PD. If the random request field (bits 0-29 of word seven) is zero, but the random return field (bits 30-59) is non-zero, when 2RD or 2WD return to CIO, CIO advances the return address field by the number of sectors transferred by 2RD or 2WD. 2RD or 2WD return the sector count in the PPU, A register. Thus, for example, a CPU program can create a random file without requiring the extra overhead of loading 2PD for the file extension operation. Of course, the CPU program must then move the random address from the FET to the file index or directory itself.

Job Files

All the information related to job processing is stored on files. Prior to execution the job control information and data are recorded on an input file. During execution the CPU programs used by the job make use of a wide variety of files. When job execution is temporarily suspended (rollout) a file is used to contain the job execution status. Finally, all printed and punched card job output is contained in files.

The job input file may be written by a number of different processors, depending on its origin - console terminal, local card reader, remote card reader, etc. In each case the file has a standard structure - words, records and files - although different PPU programs (CIO, 1BP) may be used to record the file.

The job input file has a preset format. Record one must contain all control statements for the job. The remaining records and files may contain data for system processors (e.g., FORTRAN statements for a compiler), binary program modules, or program execution input data. The first six words of the first record contain basic, job accounting data. These words are supplied by the system job card translator, TJC.

TJC must be called by each input file creation processor, prior to the recording of the first input file record. The TJC call addresses a FET, whose buffer pointers locate a six word area to receive the accounting data. The calling program supplies job card data to TJC in a series of coded images, starting at word seven of the buffer.

The function supplied to TJC in word one of the FET defines the number of coded images to be processed. There must be at least two. A third may be used to supply the job name. If the job name is supplied on a third coded image, TJC prefixes that image to the main, job card coded image.

The main job card coded image contains a job name, accounting fields and job execution parameters [12]. At the option of the user to whom the account is allocated, he may add password protection to the account. If he does so, the last coded image supplied to TJC must contain the correct password. The job execution parameters define the initial execution FL, and the limits of resources to be used during job execution - e.g., CPU seconds, print lines, I/O transfer units, etc.

One of the accounting fields is a three character user identifier. TJC uses this identifier to locate an account file validation entry. The identifier consists of three characters from the set (A,5). Each character covers the range (1,32). Decremented by one to cover the range (0,31), reduced to five bits, and placed side by side, the three characters form a fifteen bit random index which directly locates the account file validation entry.

The account file validation entry is a key part of the small permanent file system, PFILES. It contains the user account code, password, individual job resource limits, and default job resource values. The default entries supply values not entered on the job card coded image. Values specified on the coded image are matched against the limits in the accounting file entry.

If errors are detected, TJC returns an error code in the FET, and an error message in the FET buffer. If no errors are detected, TJC returns a six word accounting data block in the FET buffer. These six words contain the account identification, resource limits, and a set of resource access permission flags. These flags regulate job access to such resources as permanent files, plotters, etc.

Once TJC has completed job card processing, and assuming that no errors were detected, the input file processor can record the input file. During this phase, the file is treated and recorded as an ordinary local, disk file - e.g., CIO may be used to record the file. Once the job file is completed, it is released to the job input queue.

Releasing a local file to the job input queue is simply a matter of manipulating the FNT/FST words. The file type must be changed to input (00). The control point assignment must be set to zero. The job execution parameters must be placed in the FST word - byte two is the CPU time limit divided by eight; byte three is the initial FL divided by 100_8; and byte four is a non-zero, queue priority.

The queue priority determines the order in which the job is selected for execution. Once a job has been selected, the input file becomes assigned to the selected control point and is positioned to the beginning of the second record. The location of the first record is recorded in the control point area (word 67_8 or CSPW) for use by job control card processors.

The Rollout File

Job execution can be interrupted and restarted using the processes of rollout and rollin. When a job is rolled out, the PPU program IRO creates a file which contains all the information necessary to restart (rollin) the job at a later time. The rollout is determined by the job scheduler, ISJ. CPUMTR and MTR cooperate to assign a PPU to IRO as soon as the control point has no PPU activity.

The rollout file written by IRO has the following format:

Sectors	Contents
First Record	All FNT/FST entries assigned to the job. The file INPUT is the first entry.
Next Two	The 200_8 word control point area
Next One	The Dayfile buffer pointer
Next One	The dayfile buffer
Next FL/100_8	The job central memory
Next One	The PPU delay stack entries
Next One	I/O stack entries
Last One	EOI

The rollout file FNT/FST entry replaces the input file FNT/FST entry. Thus the address of the rollout and input file remains the same during job execution. This address is sometimes called the job address. It is used for placing the job in queues (ERT) and assigning resources to the job (e.g., a DRB or TCB). The job address can change when a job is transferred from one machine of the Dual MACE System to the other, but only if the job address does not appear in any queues or resource lists in the originating machine.

When the rollout is complete, IRO normally frees the control point for further usage. The job rollout file FNT/FST entry has the format:

Word	Bits	Value
1	59-18	Job name
	17-12	Job control flags
	11- 6	10_8
	5	Interrupt inhibit flag
	4- 0	Zero
2	59-54	The job ID (routing) code
	53-48	The rollout file equipment
	47-36	The rollout file first track
	35-24	The remaining CPU seconds divided by eight
	23-12	The rollin FL divided by 100_8
	11- 0	The job queue priority

Again, the queue priority is used by the job scheduler to determine the eligibility of the job for continued execution.

Output Files - Print and Punch

Print and punch output produced during job execution is preserved and directed to output device processors using a set of five, reserved files:

OUTPUT - printer output
PUNCH - coded (hollerith) card output
PUNCHB - binary card output
PUNCH8 - column image card output
P8

To the executing job processors, these files are standard, disk files. They carry special file types to define their special usage. The file types are established by ØBF when the files are entered in the FNT/FST table.

When a job terminates, the job completion PPU program, ICJ, releases these files to their respective queues. The files are renamed with the job name, the file type remains unchanged, and the control point assignment field is zeroed. The fields of the FST entry depend on the file type:

Bits	Value
59-54	Job ID (routing) code
53-48	Disk equipment number
47-36	First Track
35-24	Unused for PUNCH, PUNCHB, PUNCH8 and P8. For OUTPUT, the first dayfile track.
23-12	Unused for PUNCH, PUNCHB, PUNCH8 and P8. For OUTPUT, the first dayfile sector.
11- 0	The queue priority - a function of file volume [9].

Bytes two and three describe the location of the job dayfile within the OUTPUT file. The dayfile is copied from disk to the end of the OUTPUT file at job termination. It includes job execution messages and job performance data. The presence of the dayfile address permits console output processors to locate and provide a job performance preview without requiring a time consuming skip over execution print data in the OUTPUT file.

The queue priority field is derived from the file volume and from the job account number. Three bits describe the account class, the remaining nine bits describe the volume. The volume descriptor is based on 512 line groups for OUTPUT files, and 16 card groups for PUNCH files. It is formed such that the smaller the volume is the larger the volume descriptor will be - i.e., a value of 777_8 is the smallest possible volume descriptor.

Output files are routed to the appropriate device processors by the job ID code. This six bit code is transferred from the job input file at termination time. It may have been placed there by the processor which made the input queue entry, or it may have been placed there during job execution. In general, the following values are used:

ID Code	Routing
00	The job came from and returns to the system unit record I/O processor, BATCHIO
10,40-47	The job came from and returns to remote I/O processors
20	The job came from and returns to a remote console of the PROCSY system

Other codes may be used to indicate special forms requests, etc.

Unit Record Input/Output - BATCHIO

The main, direct, system, unit-record ("spooling") processor is the BATCHIO sub-system. The PPU programs of the BATCHIO sub-system process card reader job file input and output of job files to printers and card punch. Currently BATCHIO manages four line printers, two card readers, and one card punch.

The main PPU program of BATCHIO is the combined driver, 1CD. This program transfers data between the unit record I/O devices and CM buffers. The device and buffer together are called a "buffer point". 1CD calls upon an auxiliary PPU program, 1BP (BATCHIO file processor) to transfer data between the CM buffers and disk files. 1BP uses the 2RD and 2WD overlays for disk file processing. 1BP also performs job card translation (using TJC) and various other utility functions (e.g., page banner formation), using various 1BP overlays.

The third PPU program is 1IO, the BATCHIO manager. This program selects print and punch files from the queues for processing by 1CD and 1BP. 1IO also monitors the status of idle card readers, activating 1CD processing when it senses that the readers contain cards for processing.

When BATCHIO is activated at a control point by a type-in from the system console keyboard, it associates itself with the unit record equipment by scanning the EST for the equipment types CR (card reader), CP (card punch), and LP/LQ (line printers). 1IO attaches these devices to the control point as required. Thus they may be used by control point jobs when not in use by BATCHIO. In practice, this is done infrequently, usually only when device diagnostics must be run during system production time.

Each device can have an ID routing code associated with it. When a job file is released to the input queue, the ID code of the reader is inserted in the file FST entry (bits 54-59). When 1IO selects a print or punch file for processing, it matches the file ID code to that of a device. The ID codes 70_8 through 77_8 are used to denote special processing. When assigned to a card reader, they specify that the job is to be given express mode handling. When assigned to a printer, they direct 1IO to select only files of specific volumes, using the nine bit volume field of the queue priority. For example, a line printer device with an associated ID code of 76_8 is restricted to the printing of jobs whose queue priority indicates a volume of 512 lines or less.

A number of console entry commands are available for controlling the actions of BATCHIO. These permit the suppression of carriage control processing, the termination of file processing, etc. These commands are transmitted to BATCHIO by DSD in the control point area.

DSD recognizes BATCHIO for this type of processing by the presence of an eighth job name character of one in bits 18-23 of word JNMW of the control point area. The presence of this character also stimulates two DSD display overlays ("I" and "O") to picture special device, buffer point status information.

In addition to its processing of job input and output files, BATCHIO references an additional special file. This file is called BANMESS. It is a common file (see

the Permanent Files section), and contains pertinent, up-to-date system announcements. It is referenced by IBP and its contents included in the first, print file page, the job banner page. This page, in addition to the banner file data, contains the job name in block letters, date and time information, and marker characters, printed across the bottom fold of the face-up banner page. This banner page format provides easy identification and location of the beginning of each job output stack.

Permanent Files

Permanent files in the Dual MACE system are those files which can be stored and retrieved in spite of intervening TAPE dead-starts. There are three forms of permanent file storage: (1) private, 854 disk packs; (2) large permanent files; and (3) small permanent files (PFILES). All three use the same basic mechanisms. The small file system provides for the separation of a large file (or files) into separate, smaller allocation units.

The permanent file structure of Dual MACE is based upon the presence of a track reservation table (TRT) and a directory index in a fixed track of each permanent file device. The first track is used on 854 packs; it is specified at dead-start for other disk types. When the system is dead-started (or an 854 pack is mounted) the reservations of the disk TRT's are merged with those of the CM and ECS TRT's. The directory index provides the method of locating an individual file.

The directory of all devices except 854's are copied to a single master directory at each dead-start. This facilitates directory searching. The entries in the device directories, and hence the master directory, reference only sub-directories. The first reference to a file occurs in these sub-directories. The sub-directories may themselves reference other directories. A directory which is subordinate to a first level directory (one referenced in the device directory) is called a family.

The 854 pack represents a minor exception to this structure. In the first place, the 854 directory is not referenced by the master directory. Secondly, the 854 directory can reference files as well as directories (families).

All permanent files and directories must be declared before usage. Files must be saved after creation. Thus it is not possible to record a file on a permanent file device and then declare it to be permanent. Files and directories may be accessed in one of four modes: (1) read, if the file or directory exists and is to be read only; (2) write, if the file or directory exists and is to be modified; (3) old/new if the file or directory is to be modified, but its existence is not known; and (4) new if the file or directory does not exist.

Permanent file access and management activities are performed by the permanent file management PPU program, PFM. PFM, in turn, makes use of other PPU programs, and the permanent file management facilities of CPUMTR. A number of CPU programs exist for using the facilities of PFM. They may be called with the following control cards:

Control Card	Action
FILES (x)	locate and obtain access to directory x.
FAMILY (x,y)	locate and obtain access to family directory y in directory x.
ATTACH (y,z)	locate and obtain access to file z in family or directory y.
MOUNT (p)	mount 854 pack p.

Others exist for purging files, directories, etc.

All PFM activity is performed in terms of files. When a directory is accessed, PFM creates a file, attached to the control point of type 12_8 (directory-read) or 11_8 (directory write). An FNT entry for a file attached to a directory or family has a type 13_8. Even the master directory is processed in file fashion, although, in general, no FNT/FST entry exists for it. Thus most requests for PFM processing refer to two files (e.g., the directory and the file, or the master directory and the directory, etc.). A request to mount an 854 pack is a slightly different case and will be considered separately.

Accessing a Directory

When PFM is called to access a directory (e.g., the FILES control card), the parameter block addressed in bits 0-17 of the PFM call supplies the name of the directory in which the named directory is located, the read and write passwords to the directory, and the type of access - read, write or old/new. If no directory is supplied, PFM uses the master directory. If a directory is specified, PFM must be able to locate the directory file attached to the control point. The directory file type indicates the access mode permitted (DR for read, DW for read, write or old/new).

A search of the master directory, if successful, is followed by a search of the device directory referenced in the master directory. The search must result in the location of the directory for read or write access requests. It may or may not result in the location of the directory for old/new access. It must not result in the location of the directory for a new access request.

All directories, including the master directory have essentially the same format. The first sector of the directory is a label sector, containing the directory name and access passwords. The remaining sectors contain the indexed entries - files or directories - in eight word blocks, eight to a sector.

Each entry has the format:

Word	Bits	Value
1	59-18	File or directory name (0 if none).
	11- 6	Entry Type 11_8 = directory 13_8 = permanent file
2	53-48	Equipment
	47-36	First track
	35-24	Current track (0 if file not "saved").
3	59- 0	Read password
4	59- 0	Write password

The access passwords, if specified, must match. If only a write access password is required and it is not supplied correctly, read-only access is forced. The type of access granted is indicated by the inserted file type and write-interlock bit (bit 5) in the FNT entry.

Accessing a File or Subdirectory (Family)

The ATTACH and FAMILY control cards use PFM to access files and subdirectories. PFM accepts four parameter words, specifying the directory, file, and passwords. The directory file must be attached to the job control point. PFM searches the eight word directory entries for the referenced file or directory. A password check is performed. The accessed file or directory is returned as a file of the appropriate type.

When PFM creates a new file or directory entry, it fills in the eight word block with the supplied parameters. A single track is assigned. The label of a new directory is recorded and all index entries are zeroed. The current track field of word 2 of a file entry is set to zero to indicate that the file is new and not "saved".

Saving a File

A file may be "saved" by a PFM call through the SAVE control card. The PFM parameter block has the same four word format-directory, file and passwords. PFM sets the current track value of the directory entry to the first track. It then loads 1PF into the PPU by re-writing the PPU input register word in CM and returning to PPU resident. Files which are not specifically saved are, in general, automatically saved by ØDF at job termination.

IPF updates the device TRT. It does so by reading the TRT chain from CM and moving the chain to the device TRT. IPF is also used by PFM to purge a track chain when a file is purged.

Permanent File Interlocks - DDTM and RDTM

All access to existing permanent files is regulated by the TRT access bits processed by CPUMTR. When PFM has located an existing entry it requests access to the equipment and track specified in the entry by issuing the RDTM (request device/track access) function to CPUMTR. The function parameters indicate the access mode (read or write), the equipment, track and calling job address.

If CPUMTR cannot grant access - the file is already in conflicting use - job processing is usually suspended, to await granting of access via the equipment reservation table processing of CPUMTR. The RDTM function places the job request in the ERT, although the caller may set parameters to avoid this queueing. While the job is waiting for access PFM can lower the queue priority and enter PPU delayed recall (normal job), or it can send an interrupt to a terminal user.

When access is granted the job is rolled in and PFM completes the access. In the case of a terminal user, the MESA terminal processor is interrupted and a PFM call is reissued for the completion of an access request.

When a permanent file is saved or returned, the PPU program returning the file - ØDF or PFM - issues a DDTM (drop device track access) function request to CPUMTR. CPUMTR clears the access bits from the TRT and examines the ERT for an eligible, waiting entry. If one is located, the job is stimulated (queue priority raised or interrupt flag set) to resume and complete the processing of the suspended access request.

854 Disk Packs - MOUNT

With one exception, the 854 disk pack is processed exactly as are fixed surface permanent file devices. The exception is that the pack directory is not included in the master directory. A pack is accessed with the MOUNT control card. Processing of a MOUNT request by PFM requires console operator communication for the physical mounting of the pack.

PFM uses the device request blocks (DRB's), the two special equipments DY and DZ, and a special disk pack pointer table (DPT) in the operator communication process. PFM first examines the DPT. This table indicates which packs are already mounted and on what equipment. If the pack is located in this table, PFM requests access to the pack directory track with an RDTM function.

If the pack is not located in the DPT, PFM examines the DRB table for a pending mount request. If one is located, PFM requests access to the DRB, using equipment DY and the RDTM function. Job processing is then suspended via rollout until the pack is mounted.

If no DRB request is located, PFM requests access to a drive with an RDTM request on equipment DZ. If the request is granted, PFM fills out a DRB, thus requesting that the operator mount the pack. PFM then rolls out the job. If no drive access can be granted, the job request is queued in the ERT, and PFM rolls out the job to await the availability of a drive.

When a new pack is mounted, PFM copies the TRT of the pack to the CM TRT. Since 854 drives are not shared by the two machines of the dual system, no ECS TRT manipulation is necessary. Further access to the pack is regulated by RDTM functions on the pack directory track. PFM also creates a DPT entry for the pack.

Access to files and directories of the pack are regulated by searches of the pack name directory file and RDTM requests. At this point, all PFM processing for an 854 pack is identical to that for fixed surface permanent file devices.

Common Files

Common files in the Dual MACE system are files containing frequently accessed data. These include relocatable subroutines, program source libraries, statistical systems, virtual systems (see Job Processing), etc. These files are accessed by assemblers, loaders, control card processors, etc. Every common file is also a permanent file. To simplify and streamline common file access, the PFM FILES, FAMILY, and ATTACH mechanisms are bypassed for common file processing.

Instead, common files may be accessed with a single request - e.g., a COMMON control card. The local file manager PPU program, LFM, processes the request. A common file can be located in the FNT/FST (type = 02, write protect bit = 1, control point assignment = 0). It may also be indexed by a special, common file index file. This file, named COMMONX, of type 17_8, indicates the directory name in which the common file is located.

LFM searches the FNT/FST and then COMMONX, the master directory and the referenced directory on the indicated device. When the file is located, LFM creates an FNT/FST entry of type 7, and requests read access to the file with an RDTM function. LFM waits for access via rollout or interrupt exactly as does PFM. Note that since LFM grants read-only access to common files, a number of simultaneous accesses to the file can be permitted.

Small Permanent Files - PFILES

The manner in which disk tracks are allocated to permanent files and directories is often wasteful of space. The main permanent file device of the Purdue hardware complex is the 821 disk. Each logical track of the 821 contains 320 sectors. Since each directory and file reside on a separate track, the potential for wasting space is very large.

To make more efficient use of disk space, a small permanent file subsystem, PFILES, is available in Dual MACE. This sub-system divides several very large, random permanent files into a large number of individual user segments. PFILES allocates space in these files in four sector units. The division control is the user data block (UDB) which is addressed with a three character (A through 5) user identifier.

PFILES maintains a user data block area for each identifier. This area contains the accounting data used by the job card translator, TJC. It also contains an index of the files stored for the identifier, including passwords, activity counts and file age. Files stored in the PFILES sub-system must be copied to an utility disk before usage ("GET") and must be written on an utility disk in order to be "PUT" (copied) by PFILES. This insures that PFILES retains full control of the space allocation of the large files.

Since the files are random files, PFILES can make use of the standard, CIO/2PD random file functions. Since PFILES manages all space on the file, it can make use of open sector blocks in the file as files are deleted and new files stored. The entire PFILES data base is periodically backed-up on tape. On shorter intervals, changes to the data base are dumped to tape. Thus the file can be fully or partially restored in the event of hardware or software failures.

Access to individual user data blocks is controlled by a special TRT and equipment DU. The CPUMTR functions RDTM and DDTM provide access control. The TRT of equipment DU is handled in a special manner - each TRT word represents one user data block, and the TRT is used more as a list than as a table. When an access to a user data block is in progress, some TRT word will represent the user data block. Another access request for the user data block will be queued in the equipment reservation table, and allowed to proceed when the first program releases access with a DDTM function.

Track 4003_8 of the DU TRT is used to queue access requests when the TRT is filled. Each time a user data block access is released (and possibly a TRT word), the PFILES control PPU program, CTL, checks for a queue on track 4003_8 and issues a DDTM, if necessary, to stimulate the programs waiting for space in the DU TRT.

Special File Types

There are a number of special file types. These include:

File Type	Usage
14_8	PFILES utility
15_8	Delayed job
17_8	System common
20_8	File index

The PFILES utility file type is used for access to the permanent files of the PFILES sub-system. Usage of this special type permits PFILES to bypass some of the complications of permanent file access resulting from the special functions of PFILES. It also permits the definition of special handlers for the returning of PFILES access in ØDF.

The delayed job type provides a mechanism for the periodic running of special system tasks. A file of this type represents a job whose execution is to be resumed at a later time. The time delay is specified in byte two of the FST entry in units of eight seconds. Once each eight seconds the repetitive services PPU program, IRS, decrements the time delay field.

When the delay is completed, IRS returns the job to the input or rollout queue. Bits 12-17 of the FNT entry word carry the queue file type - 00 for input, 10_8 for rollout. One of the system programs which uses this facility is a CPU program named SIFT. SIFT transfers files between machines of the Dual MACE System, using the inter-machine function facilities provided by the PPU programs IRS and ICP.

The system common file type is used to identify files of special system-only usage. One such file is used to buffer inter-terminal messages. Another is used as an alternate page swapping device for the terminal system processor, MESA. Access to system common files is restricted to specially authorized CPU and PPU programs.

The file index file type is used to identify files which index other job files. Currently this file type is used to queue jobs waiting for compilation by the batch version of MNF, the Minnesota FORTRAN Compiler [13]. During dead-start recovery, this file type directs the PPU recovery program, REC, to recover the files indexed as well as the index itself. Use of a file index file has the obvious advantage of releasing FNT/FST table space while the indexed files are waiting for processing.

Special Media

A number of special media are available for file storage in addition to disk devices. These include card devices, magnetic tape units, a paper tape reader and punch, and an electro-static printer. The basic file characteristics of these devices are the same as disk files - coded images, records, files - but the characteristics of the devices require the introduction of a recording unit, the PRU or physical record unit.

Actually, a disk file has a PRU value of one sector. Fortunately the PRU size is so transparent to the disk file user that it is hardly ever worth mentioning. Unfortunately, the PRU size of special media usually has a significant influence on the processing of files on the media. For example, one must know the physical record length, or number of characters recorded between gaps on a tape, in order to be able to process it correctly. As another example, the frame count and size are special characteristics which must be known for proper paper tape processing.

All of these various special characteristics are designated for the device drivers in fields of the file, FST entry. Bytes one, two and three (bits 12-47) are used to contain these values. The fields are entered from parameters supplied on the request to mount the special media (e.g., a tape).

Access to Special Media

All requests to mount a file on a special medium device are processed by the request processor PPU program, REQ. The REQ call addresses a parameter block which defines the file name, external identifier (e.g., tape reel, visual label) and other device processing parameters as required - tape density, physical record character count, etc.

REQ manages access to the special media devices in two ways: (1) single devices such as the paper tape reader, are accessed via equipment DZ and the RDTM/DDTM facilities of CPUMTR; (2) tape units are allocated in cooperation with the job scheduler. When a job requests access to a paper tape reader, for example, REQ issues an RDTM request to CPUMTR. If the device is available, REQ fills a device request block (DRB) with mounting instructions and rolls the job out to await operator action. When the device is ready, the operator enters the assignment, the job rolls in and processing can continue.

When the device is not available, the equipment request is queued in the equipment reservation table by CPUMTR. REQ rolls the job out. When the equipment becomes available, CPUMTR raises the job queue priority, rolls the job in, and REQ posts the mounting instructions in a DRB.

After a special medium file has been mounted, REQ uses the location free (zero-level) PPU overlay ØAE, assign equipment, to create the FST entry. Using the parameters supplied to REQ, and the equipment type, ØAE creates the device defined special fields of the FST word.

Tape Scheduling

REQ processes magnetic tape requests in a slightly more sophisticated fashion, in order to be able to process multiple unit requests efficiently. The number of possible units required must be specified in advance on the job control card (the TPn parameter) or by a special call to REQ (e.g., with the TAPELIM control card). When REQ detects the first tape unit request, it rolls the job out and sets the value 4x in bits 12-17 of the rollout file FNT entry. "x" is the number of tape units required.

The system job scheduler, ISJ, regulates the committment of tape units. Units are scheduled to jobs according to their availability and job queue priority. After ISJ has committed a tape unit or units to a job, the job is rolled in and REQ continues processing. REQ fills in a DRB with the mounting instructions for the first request. It then reads the next control card of the input file first record looking for another tape request. All such consecutive requests are posted in DRB's. The DRB's are linked together circularly.

When all consecutive requests are posted, REQ rolls the job out, placing a call for the device manager PPU program, IDM, in the rollout file delay stack sector. When all tapes have been mounted and assigned by the operator, as indicated by the DRB linkages, the job is rolled in.

IDM completes the request processing, using the DRB fields. IDM uses ØAE to form the FST word from parameters stored in the DRB. In addition, if any of the tapes are assigned to 7291V units, IDM sends a function to the front end, IBM 7094, machine to indicate that processing of the tape has been initiated.

Card Formats

As special media, the card punch and card reader are not often used for direct, file processing. Instead, the devices are usually driven by the combined driver PPU program, ICD, of the BATCHIO sub-system. ICD processes three card formats - coded, binary and column images - and a number of special flag cards - EOR, EOF, EOI and column image escape.

A coded card is one punched in standard, IBM 026 key-punch format [14]. BATCHIO translates these codes to CDC Display Code [7], six bits per character, in a coded image. Trailing blanks, in multiples of two, are deleted.

A binary card is identified with punches in rows seven and nine of column one. It may contain as many as fifteen CM words in columns three through seventy seven. A binary card also contains a word count (rows zero through three of column one), a checksum (column two) and a sequence number.

The column image card contains sixteen CM words in columns one through eighty. Column one contains bits 48-59 of word one; column two, bits 36-47; column five, bits 0-11; column six, bits 48-59 of word two; etc. Row twelve contains the high order bit (e.g., 59) and row nine, the low order bit (e.g., 48). The beginning of a set of column image cards is signalled with a flag card, containing punches in rows five, seven and nine of column one, and rows four, five, six, seven, eight and nine of column two. The end of a set may be signalled by the same card or an EOI card (punches in rows six, seven, eight and nine of column one and none in column two).

These three card formats hold for card input and output. BATCHIO translates the data in the PUNCH file (type 4) to coded. The PUNCHB file (type 5) is translated to binary. The PUNCH8 or P8 file (type 6) is translated to column image.

The same flag cards hold for input and output with one exception - no column image flag cards are punched. An end-of-record (EOR) is a card with punches in rows seven, eight and nine of column one. Column one of an end-of-file (EOF) card contains punches in rows six, seven, eight and nine.

Job Processing

Once a job input file has been recorded and the FNT/FST entry released to the job input queue, the job is a candidate for execution at a control point. The actual assignment of the job to a control point depends on a number of criteria - job field length, memory availability, control point availability. The selection of a job for execution which satisfies these criteria is performed by the system job scheduler.

Job Scheduler

The system job scheduler is the PPU program 1SJ. 1SJ is activated at dead-start by STL. It runs on a four second delayed recall cycle. It may also be activated with the CPUMTR function RSJM. This function is used by various programs - e.g., BATCH10 - when they have made a system change which might affect job scheduling. 1SJ activities are directed by means of tables, control point status flags, system status data, and job queue priorities. Job queue priority is the primary factor. It is used to relate jobs in the input and rollout queues to the queue priority category tables of the installation area. It determines, in general, the order of selection of jobs for execution and rollout.

The job queue priority is a twelve bit field, stored in bits 0-11 of the queued file FNT/FST entry. When a job is executing, the queue priority is stored in bits 12-23 of word 60_8 (CPCW) of the control point area. Two main classes of queue priority exist - first pass and execution. A number of other queue priority classes are defined for control of interactive jobs, resource assignment and system job activities [9]. A queue priority of zero indicates that the FNT/FST entry is being modified - i.e., the entry is reserved.

Job Queue Priority

The first pass queue priority is assigned to a job when it is placed in the input queue. The value is generally computed by TJC during job card translation, but it may also be determined separately by the processor which releases the job to the input queue. When TJC computes the priority, it returns the value in the parameter block (FET) controlling the translation operation.

The first pass value computed by TJC is based on three factors: job origin, a job card priority field, and estimated job track usage. Job origin may be a local or remote card reader, remote console, etc. The job card priority field translates to a six bit value. Queue priorities derived from these two factors generally fail in the range $60xx_8$ to $63xx_8$ where xx is the job card priority value. If the estimated job track usage is large (over 105) the first pass queue priority will be $24xx_8$.

The first pass queue priority remains in effect for "first-pass" computation units (generally "first-pass" = 25). A CPU second or sixty five I/O transfer units (one I/O transfer unit is 1,000 characters transferred to or from an I/O device) constitutes a computation unit. At the end of "first-pass" units, the queue priority is re-evaluated using the PPU common deck program CQP and the queue priority weight tables. This new value is called the second-pass or execution queue priority.

All execution priorities are limited to the range 100_8-5777_8. The maximum and minimum values are stored in the queue priority weight table header word and can be modified with console entries. The upper limit on the execution queue priority has been chosen such that all execution priorities are less than all first pass priorities. This permits new jobs entering the system to obtain "first-pass" units of execution very rapidly. As many as 75% of the jobs processed by the Dual MACE system complete their execution in "first-pass" units.

A wide range of factors enter into the computation of the execution queue priority. Essentially they describe the resources in use by a job (e.g., field length) and the resources required to complete job execution (e.g., CPU seconds). The parameters of the queue priority weight table describe the parameter test values and their influence in the queue priority computation. That influence is reflected by a fixed increment and by a difference factor. For example, CPU seconds remaining is compared to thirty. If it is less than thirty a 300_8 increment is added to the execution priority. Central memory in use (FL) is compared to 150100_8. If it is less than 150100_8, the execution queue priority is advanced by the absolute value of the difference between FL and 150100_8, divided by 2^{-7}. The difference factor 2^{-7} in this case is always represented as a power of two - i.e., a PPU shift operation.

Job Scheduler, Queue Priority Categories

The resulting queue priorities - first-pass, execution, interactive, etc. - can be divided into ranges by virtue of the nature of their computation. Execution values are in the range 100_8-5777_8. First-pass values are in the range 6000_8-6377_8. The scheduling strategies of the Dual MACE System are expressed in terms of these ranges. The job scheduler category tables specify these strategies.

The category tables define a set of queue priority ranges and the manner in which the PPU program scheduler, ISJ, is to consider those ranges. The tables include control point limits, field length limits and candidate tables. ISJ maintains corresponding counts in the tables (stored in the installation area) for display purposes - the DSD "W" display.

The tables can be viewed as a nine by seven array. The first column contains the lower queue priority of the defined range. The next three define the number of jobs of the category to be scheduled to control points at one time, the total field length to be allocated to the jobs, and a candidate limit. The remaining three columns of the array are filled by ISJ to indicate the current usage of each of the first three values.

The candidate limit value of the category tables is a mechanism designed to prevent a given category from overloading the internal tables of ISJ. ISJ places each candidate in a PPU memory table, ranked by queue priority. Currently the table can contain 213 entries. The candidate limits restrict the number of table entries available to a queue priority range, thus preventing a range from filling the table. In practice, the limits are set such that table space is allocated according to the probable number of jobs of the range in the system at any one time. Entries of the weight table can be added, deleted or modified by console commands should job load require it.

The current job scheduler category table has the following entries:

Queue Priority Range	Control Point Limit	Field Length Limit	Candidate Limit
7702_8-7777_8	4	300000_8	4
7000_8-7701_8	1	300000_8	10
6700_8-6777_8	1	300000_8	4
6330_8-6677_8	1	150000_8	8
6300_8-6327_8	1	100000_8	17
6000_8-6277_8	2	300000_8	50
100_8-5777_8	63	300000_8	120

When the table is not defined in central memory, ISJ copies its default, assembled values to central memory. A flag in the table area (bits 48-59 of the first word) indicates the table status. When the table is defined in central memory (and possibly modified by console commands) ISJ reads the table into PPU memory in place of the assembled values.

The table is interpreted in the following fashion. Consider, for example, the entry 6300_8-6327_8/1/100000_8/17. This entry specifies that no more than one job, of field length 100000_8 words (32,768) or less with a queue priority at least 6300_8 and not more than 6327_8 is to be scheduled to a control point at one time. In selecting that job, ISJ is to limit itself to the top 17 jobs, in order of queue priority.

In selecting and scheduling jobs in response to job queue priorities and according to the constraints of the category weights and available system

resources, ISJ builds a number of internal tables. These include a table of candidates, category usage tables, and a memory map. From these tables, the limits of the category tables,and available resources ISJ constructs a new memory map. This map may indicate no change in job assignment, or it may indicate the necessity to roll jobs out, roll jobs in, or to begin the execution of new jobs. The actual processes required are performed by other programs (e.g., CPUMTR, IRO, etc.). ISJ makes the decisions and initiates the processes.

Job Movement

Job movement processes are initiated by the scheduler with the CPUMTR function SJAM (set job activity). In response to this function CPUMTR sets the requested, pending activity flag in word JCIW (22_8) of the control point. When the control point has no activity of a given level, as determined by the pending activity, CPUMTR activates the required process. Thus, for example, if ISJ asks for a rollout via the SJAM request, CPUMTR sets the rollout pending flag. When the control point has no PPU's active (the PPU count in bits 48-53 of STSW (20_8) is zero), the rollout in progress flag is set and CPUMTR assigns a PPU to the rollout program, IRO.

In addition to the constraints imposed by the category tables, ISJ recognizes a number of other constraints. These include field length availability, control point and scheduler activity flags, tape unit usage, etc. In field length processing, ISJ not only recognizes the machine limits, it services requests for additional field length and obtains the specified field length required to initiate a new job or roll in an existing one.

Control point and scheduler activities are regulated with a number of flag bits. In word four of control point zero (JSCL), bit twelve, if set, inhibits job rollout/rollin; bit thirteen, if set, inhibits all scheduling. In each control point area, word 60_8 (CPCW) contains two scheduling control flags. Bit 59, if set, inhibits all job scheduling activity. This bit is set by DSD when it is modifying job parameters (e.g., time limit). Bit 48 is set to indicate that the control point may be used for processing. This bit is called the "idle" or "next" flag. When it is set and no job is being processed at the control point, the control point job name will be "NEXT" to indicate that the control point is available for processing.

The job scheduler regulates tape unit usage by examining bits 12-17 of the rollout file FST entry. A value of 4x indicates that the job must have x tape units in order to continue execution. If the units are available, ISJ assigns x device request blocks (DRB's) to the job before rolling it in. Since queue priority is the main basis on which jobs are chosen for execution, an available unit is generally allocated to the waiting job which has the largest queue priority.

Beginning a Job - 2BJ

When 1SJ sets a pending job activity with an SJAM function call to CPUMTR, the type of activity set can be initialization, rollin or rollout. If a job is being scheduled from the input queue, initialization is set. CPUMTR sets this activity in JCIW, and, since no processing activity is present, activates the job advancement PPU program, 1AJ.

1AJ processes all job advancement. It is activated by CPUMTR whenever a control point has a job present but has no CPU or PPU activity pending or present, and no I/O stack entries. For job initialization 1SJ sets the control point in such a status that 1AJ can recognize that it is being activated for the first time. 1AJ, in turn, alters those statuses so that it will recognize an initialized job on subsequent calls.

When 1AJ recognizes that it has been called to begin a new job, it executes its begin-job PPU program segment, 2BJ. 2BJ sets up various control point fields - e.g., time limit, I/O unit limit, accounting header, control card pointers, etc. It also modifies the INPUT, file FST entry to point to the beginning of the second record. The location of the first record is stored in word CSPW (67_8) of the control point area. When 2BJ finishes it returns to 1AJ in such a manner that 1AJ can continue as though it had been called by CPUMTR to advance an existing job.

Job Rollin - 1RI

When 1SJ has decided to roll a job in at an available control point, the SJAM function call made to CPUMTR results in the assignment of the PPU program 1RI to an available PPU. The PPU is assigned to the control point, and 1SJ indicates the rollout file address in bits 48-59 of word CSPW (67_8).

1RI reverses job rollout by reading the suspended job status data from the rollout file. Job file entries are moved from the FNT/FST record to the FNT/FST table. The INPUT file entry replaces the rollout file entry. The control point area, the dayfile buffer, the job field length, and the I/O stack areas are re-loaded from the rollout file. PPU programs in delayed recall before rollout are re-entered with zero delay time - the "R" status PPU call is placed in RA + 1. The CPU is restarted, if in use at rollout.

During the reloading of these values from the rollout file, 1RI checks their validity in a number of ways. The FNT/FST entries must represent valid files in name, type and equipment parameters. Each area must be of the expected length. If any tests fail, 1RI stops the system with an "-ERR RI", top-line stop.

At the end of a successful rollin, 1RI processes any job modification commands entered while the job was rolled out. These commands are stored in the dayfile pointer sector of the rollout file. They are processed by the 1RI segment 2RI. The commands are entered in the rollout file in response to console keyboard

entries. For example, the operator can enter a message, set an error flag, raise the CPU priority, etc., of a rolled-out job. When the job rolls in 2RI completes the processing of these operations.

Job Rollout - 1RO

1SJ also initiates job rollout with an SJAM function call to CPUMTR. As soon as no PPU activity is present at the control point, CPUMTR activates the job rollout PPU program, 1RO. Note that 1RO can be activated while the CPU is assigned, PPU's are in the delay stack and while I/O stack entries are pending. CPUMTR blocks further processing (e.g., new PPU calls or recalls) because the rollout activity flag (bit 43 of STSW (20_8)) is set.

1RO writes the rollout file in the format already described. When the rollout file is completed, 1RO replaces the INPUT file FNT/FST entry with the rollout file entry, releases the field length assigned to the control point, and clears the control point area. Further use of the control point is determined by the status of the system and control point activity flags, and the processing fules of 1SJ.

In one unusual case, 1RO may leave the job assigned to the control point. This case arises when an equipment (e.g., a line printer) is directly assigned to the control point. This case is unusual, since the mechanisms of device request blocks and disk pack assignment have been provided for linking equipment to jobs. Certain special programs - e.g., a line printer test can, however, attach equipment directly to the control point, provided they have proper authorization. Control points having such programs active cannot be fully rolled out without damaging the equipment-control point linkage. When 1RO processes such a control point it performs all normal rollout activities except the clearing and releasing of the control point.

Job Execution and Advancement

Once a job has been assigned to a control point, it makes use of CPU and PPU resources with control card calls. These calls originate in the coded images of the first record of the INPUT file. In addition, procedures can be invoked, whose control cards are stored on a file which may be joined to the INPUT file as an alternate INPUT file.

The job advancement PPU program, 1AJ, processes all control card calls [15]. 1AJ is activated by CPUMTR whenever a job has no processing active. Except as noted in job initialization, each call to 1AJ consists of two types of processing: error processing; and control card advancement. Normally, error processing is followed by control card advancement. Where specified, 1AJ can return control to a CPU program without control card advancement.

Control Card Advancement - 1AJ/2TS

Control card advancement is performed by 1AJ using the control statement buffer (CSBW) of words 70_8 - 177_8 of the control point area and the INPUT (or alternate INPUT) disk file. Pointers on word CSPW (67_8) control the position in the INPUT file and within the control statement buffer. An additional parameter word defines the status of the alternate INPUT file in word ASPW (57_8).

Using the values in ASPW and CSPW, and the PPU program segment 2TS (translate statement), 1AJ obtains the next coded image from the INPUT or alternate INPUT file. When only a part of the image is in the statement buffer, 1AJ moves the partial image to words 70_8 - 77_8 of the buffer, reads the next control card sector and appends it to the partial image. Once an image is obtained it is separated into two fields: a statement name and statement parameters. The rules for formation of these fields are given in detail in [15].

1AJ interprets the statement name as a program call. 1AJ searches for the named program in the following order:

1. 1AJ local processes - e.g. EXIT from errors.
2. Local files - user and compiler supplied CPU programs or procedures.
3. User supplied virtual system directory.
4. System CPU program library directory (SLD).
5. PPU program library directory (PLD)
6. System Procedure Library

Note that, since the search is ordered, a local file may have the name of a system library program (e.g., the FORTRAN compiler) and it will be selected for execution rather than the system program.

For the most part the 1AJ local processes are related to error exit processing. They define an error resumption point in the control card stream. They may also define dump operations. When they define dump operations, 1AJ writes a small portion of the control point field length to a local file, and loads a dump, CPU program into that area for further dump option processing.

When a control statement names a local file, processing depends on the file contents. Relocatable or overlay program files are passed to the CPU program loader, LINK. Absolute CPU programs are loaded by copying them to the central memory field length. Text records, defining procedures, are passed to the procedure execution CPU program, XEQ. XEQ, in turn makes use of the alternate input file facility to effect execution of the control card statements of the procedure.

When the statement does not name a 1AJ process or a local file, 1AJ searches for a local file named "VSIND". This file can supply an index to a user supplied system of CPU programs in absolute format. The programs themselves are contained on other, local, random files. When an index is supplied, and the statement name is referenced, 1AJ uses the "VSIND" references to locate and load the required CPU program.

If none of the first three searches yield the named program, 1AJ searches the system library directory of central memory resident, the SLD. This library indexes the CPU programs loaded from the dead-start tape to the system disk by STL. The library contains compilers, assemblers, editors, utility programs, etc. Programs of the STL are sometimes called "Control card processors".

When the SLD search also fails, 1AJ tests the statement name for a possible PPU call. The name must be exactly three characters in length. It must be one of an authorized list (or the job must have the proper authorization) and it must be defined in the peripheral library directory (PLD). When all conditions are met, 1AJ rewrites its input register word with the PPU program name and exits to PPU resident to effect the loading and execution of the PPU program.

Finally, if all previous searches fail, 1AJ loads the procedure execution CPU program, XEQ. XEQ searches a standard system procedure library, stored in the common file SPL. If XEQ locates the procedure, it is processed via the alternate INPUT file facility. If it cannot be located, XEQ issues a dayfile error messages and aborts. Control eventually returns to 1AJ from CPUMTR.

Once 1AJ has located the named program it passes the statement parameters, enters the control card in the job and master dayfiles and requests CPU assignment with the RCPM, CPUMTR function. CPU assignment is not required for PPU program execution. Two parameters may be supplied for a PPU call. Parameter one is stored in bits 0-17 of the PPU call and parameter two is stored in bits 18-35.

The control card statement may supply as many as 49 parameters to a CPU program. The parameters are stored in RA+2 through RA+63_8. The list is terminated with a zero word and the parameter count is stored in bits 0-17 of RA+64_8. Bits 18-59 of RA+64_8 carry the statement name, and the entire statement is also written to RA+70_8 through RA+77_8. The registers of the exchange area are zeroed, and the control point field length is stored in A0 of the exchange area.

Each parameter may contain seven or less, non-blank characters. Blanks are simply skipped. The period or the right parenthesis terminate the list. Separator characters are the non-blank, non-alphanumerics - e.g., commas, equal signs, etc. When the comma is used as a separator, bits 0-17 of the parameter word it follows are set zero; all other separators are transmitted in bits 0-17.

Error Processing - 1AJ/2EF

1AJ can also be activated to process errors. Errors are indicated by a non-zero value in bits 36-48 of STSW (20_8) of the control point area. They may be set by a CPU program ("ABT" in RA + 1), by PPU programs, CEFM (change error flag) function calls to CPUMTR, or by various error processes of CPUMTR and MTR (e.g., exit mode control, PPU call errors, etc.). The setting of an error flag stops most control point processes - CPU, delay stack exit, etc. As soon as all processes are stopped, CPUMTR activates 1AJ.

When 1AJ is activated, and the error flag is set, 1AJ, in turn, activates its error processing segment, 2EF. This segment performs all error processing in-

cluding message handling, interrupt control, and advancement to error processing control statements.

There are two main paths through 2EF, defined by the existence of interrupt control data in word INTW (47_8) of the control point area. If non-zero, INTW specifies the types of errors to be processed, the address in the field length of the CPU program routine which processes interrupts, and the address of an interrupt status area.

The error type control includes an "in-progress" flag, to indicate that the current error occurred while the CPU program was processing a previous error. This prevents the occurrence of a loop condition in error processing. The error type is matched with the error flag. If the type defines processing, the exchange area and other error status information are copied to the defined status area, as required. The CPU P register is set to the value indicated in INTW, the "in progress" flag is set, and the CPU is restarted, presumably in the defined interrupt program segment. These interrupt activities are effected by 2EF using the set interrupt (SINM) CPUMTR function. The interrupt control word itself is initially defined by a CPU program call to CPM, the control point manager PPU program.

If the error type does not select error processing, or if no interrupt processing is enabled, 2EF issues an error message and skips to the next 1AJ error exit control card - EXIT, PROCEED, etc. This card may invoke dump processing. Eventually it results in the calling of 1AJ to process the next control card statement.

Timer Interrupt

One other special interrupt operation is also available to the control point, CPU job - the timer interrupt. In conjunction with the flags of the interrupt control word, a CPU program may specify a time interval at whose expiration control is to pass to the CPU program interrupt routine. The interrupt request is specified in bit 50 of INTW (47_8). The time value is stored in word TICW (56_8) of the control point area.

The time value is stored by CPM, the control point manager PPU program, as the value of the real time clock at which the interrupt is required. At rollout the remaining time is computed, and a new interrupt time is computed when the job rolls in. PPU monitor, MTR, scans the control point timer interrupt words on a periodic basis. When MTR locates a control point ready for a timer interrupt, it effects the interrupt by issuing an MTRM (set interrupt) CPUMTR function. This timer interrupt facility enables the terminal system control program, MESA, for example, to maintain time slice control over executing terminal processors.

Special 1AJ Processes

Each time 1AJ is called it performs a number of special processes. These include the completion of job output files, the destruction of sensitive data, and the releasing of restricted files.

Job output files include OUTPUT, PUNCH, PUNCHB, etc. 1AJ searches the parameter list area (RA+2 through RA+63_8) for these names. CPU programs which desire job output file completion place the file names in this area (bits 42-59) together with pointers to the file ervironment tables (FET's) in the job field length (bits 0-17 contain the FET address). If 1AJ locates a valid FET in its search, it calls CIO to copy the remaining circular buffer data to the file.

A CPU program can indicate the presence of sensitive field length data by setting bit 25 of control point word SNSW (26_8). If 1AJ detects this flag, it zeroes all the job field length. This prevents the dumping of sensitive data by subsequent control cards.

1AJ also searches for special files still assigned to the control point. Specifically, these include the permanent files of the PFILES subsystem to which access is highly restricted. Any such files are released by 1AJ using the location-free (zero-level) overlay ØDF (drop file).

Job Termination - 1AJ/1CJ

At some point 1AJ will be activated and no control statements will remain - the INPUT and alternate INPUT files are at the end of the control card records. When 1AJ detects this state, it sets job termination activity with an SJAM function call to CPUMTR and releases the PPU. CPUMTR then activates the job completion PPU program 1CJ.

The job completion activities performed by 1CJ cover two main areas: (1) the releasing of job resources; and (2) the queueing of job output files. Among the resources released by 1CJ are equipments, disk space, central memory, terminal control blocks and the control point itself. In releasing these resources, 1CJ scans the system resource lists - device request blocks, EST table, FNT/FST tables, etc. - and uses CPUMTR function calls and the drop file PPU overlay, ØDF, to return resources allocated to the job.

Job output, punch and print, is released to the respective queues under the job name. The ID code field of the FST entry is set from the corresponding field of the INPUT file FST entry, thus propogating the ID code entered when the job was placed in the input queue, or the code inserted during execution.

With the help of the CPUMTR function ADFM (accounting dayfile function), 1CJ enters job usage data into the job, accounting and master dayfiles. The ADFM function processor computes the usage data and stores it in messages located in the control point, control statement buffer (70_8 through 177_8). 1CJ reads the messages from the buffer and issues them to the dayfiles using the DFM entry of PPU resident.

When the job dayfile has been completed, 1CJ copies it from disk (if necessary) and from the central memory buffer to the end of the output (print) file. The position of the job dayfile (track and sector) is stored in bytes two and three of the output file FST entry. 1CJ then releases the output file to its queue.

All of the job output released by 1CJ, print and punch, carries a queue priority based on the file volume. This volume is derived from two control point words at LCTW and LCTW+1 (52_8 and 53_8). The queue priority describes the volume in complement arithmetic - the larger the queue priority is, the smaller is the volume it represents.

The releasing of output files to the print and punch queues may be suppressed at the direction of the job. This is effected with the DISABLE (OUTPUT) control card. When 1CJ recognizes the flag set by this control card in bit 27 of control point word SNSW (26_8) it simply releases all output files. Jobs whose successful completion can be detected by control cards, and whose output is stored in another form - e.g., a permanent file - may choose to have its printed and punched output disposed of in this fashion.

When 1CJ has finished releasing all files and equipment, it clears the control point area, readying it for use by another job. The control point "idle" flag in bit zero of CPCW (60_8) and other job scheduling factors determine subsequent control point usage.

Special Job Termination - 1RB

In certain unusual cases, the console operator may wish to terminate a job, with or without restarting job execution. This process is initiated with a console entry to DSD. DSD, in turn, calls the PPU program 1RB, roll-back job, to perform the requested action.

1RB functions in much the same manner as 1CJ. There are, however, some important differences. 1RB releases no files to the output queue. Finally, if the job being processed is rolled out, 1RB must read the rollout file in order to be able to release all resources assigned to the job.

To a limited extent the job may control the activities of 1RB. The DISABLE (RERUN) and ENABLE (RERUN) control cards [12] set and clear bit 26 of control point word SNSW (26_8). When this bit is set, 1RB will perform

all termination activities except the releasing of the job to the input queue for rerun. Instead the job dayfile is given a message indicating that the rerun attempt was aborted. Job execution is terminated. A job may use this option when it is, for example, updating files in such a way that a rerun of the update process might generate erroneous results. The no-rerun flag in SNSW is also honored by REC during dead-start recoveries. If REC locates a job at a control point with rerun disabled, it enters the rerun attempt aborted message and sends the job to the output queue, rather than sending it to the input queue for rerun.

Job Limit Processing

During job execution, the utilization of computing resources is controlled by limits defined by the job card, the accounting file default limits, and their translation into the job accounting header. These limits constrain CPU time usage, I/O transfer count usage, disk tracks allocated, lines printed and cards punched. The limits are initially stored in the accounting header of the input file. The header is moved to control point words ACTW through ACTW+5 (41_8 - 46_8) by 2BJ. 2BJ then uses the values to set up special, limit control words for each resource. In addition, the resource words special, resource reserves may be manipulated during job execution.

CPU Time Control

CPU time is managed through two control point words, CPTW (23_8) and CPLW (24_8). CPTW is used to contain the amount of CPU time used in milliseconds times four (for display purposes, byte three of CPTW is used as CPU seconds, although it is actually milliseconds divided by 1,024). CPTW is updated by CPUMTR whenever the CPU is removed from the control point, and at the specific stimulation of PPU monitor through the CCPM (check CPU) function call to CPUMTR.

Word CPLW contains the CPU time limit, also in milliseconds times four. If bit 59 of CPLW is set, all job limits - CPU time, I/O transfer unit tracks, lines and cards are considered unlimited. If bits 48-58 are also set, the unlimited resource usage permission applies to the entire execution of the job. If bits 48-58 are clear IAJ clears bit 59 each time it advances to a new control card. This 4000_8 value in bits 48-59 is used by special control card programs which wish to avoid limit error exits during their execution - e.g., PFILES.

When CPUMTR detects that the CPU time used has exceeded the CPU time limit, it sets the CPU time limit error flag. Eventually 1AJ is activated, and 1AJ activates 2EF. 2EF restores any reserved CPU time by adding the value found in bits 36-47 of control point word RRSW (62_8) to the CPU time limit word. 2EF then issues an error message and advances the job or enters the CPU interrupt routine, as directed by control point word INTW (47_8).

I/O Transfer Unit Control

I/O transfer units are managed with the control point word IOUW (40_8). Bits 36-59 of this word contain the limit times 10000_8. Bits 0-35 contain the amount used. CPUMTR updates **and** checks the values in IOUW when processing the ATCM (advance I/O transfer count) function. The caller specifies equipment, transferred units (sectors, lines, etc.) and file type in various combinations. The ATCM processor converts these parameters into a character count, which is added to bits 0-35. The accumulation is then compared to the limit, unless bit 59 of word CPLW is set.

When the limit comparison indicates that the accumulated I/O transfer units exceed the specified limit, CPUMTR sets the I/O unit error flag. 1AJ and 2EF are eventually activated. 2EF finds the I/O transfer unit reserve, divided by 10000_8, in bits 18-35 of RRSW (62_8). This value is added to the limit field of IOUW. 2EF then continues in the standard fashion.

Disk Track Control

Disk track utilization is managed by the CPUMTR track function processors DTKM (drop), RMSM (reserve mass storage space) and RTKM (reserve). These processors manipulate track counts in control point word TKLW (65_8). This word has the format:

Bits	Value
59-42	Maximum tracks used
41-24	Track limit
23- 0	Sector count

Since the various system devices have a differing number of sectors per logical track, CPUMTR actually counts sectors. The sector count is converted to a standardized, 64 sector track.

The CPUMTR track function processors update the TKLW word as tracks are reserved or dropped. Byte one of the second TRT pointer word defines the sectors per track value. A parameter of the function calls defines

the type of file for which the track function has been issued. Using this file type, the CPUMTR function processors can avoid track processing for special files such as permanent or rollout files.

Comparison of the track count to the specified limit is performed by CPUMTR in monitor mode by the pre-processor routine of the RTKM function. If this pre-processor finds that the track usage is within limit (or if bit 59 of CPLW is set) it passes the call to the RTKM problem mode processor. When the track limit is exceeded, the RTKM processor changes the function call to a CEFM (change error flag) call to set the track limit error flag TKET (15_8). The output register response to this function is all zero, so the response appears to the RTKM caller as an indication that no tracks are available. Upon checking the error flag, the caller detects the TKET value, and exits the PPU program. 1AJ and 2EF are activated. The track reserve value divided by eight, is located in byte zero of RRSW (62_8) and added to the track count. 2EF processing then proceeds normally.

The additional, maximum usage field is carried in TKLW in order to provide the job user with an estimation value. This is important in view of the wide fluctuation in track count that occurs during job execution, caused by programs such as compilers, assemblers, etc., which write and return scratch files.

Line and Card Limits

Line and card limits are managed with the two control point words LCTW and LCTW + 1 (52_8 and 53_8) and the CPUMTR, CTLM (count lines) function by the various output file PPU programs - CIO/2WD, DMP, etc. Each time one of these PPU programs records data on an output file the program counts lines - (DMP) or requests that CPUMTR count the lines (CIO/2WD) of the file, central memory circular buffer. The line and card limit words have the format:

Word	Bit	Value
LCTW (52_8)	59-30	Print line limit
	29- 0	Line limit minus the number of lines written in the OUTPUT file
LCTW + 1 (53_8)	59-30	Punch card limit
	29- 0	Card limit minus the number of cards written in the PUNCH, PUNCHB, and PUNCH8 or P8 files.

CTLM counts a line or coded (PUNCH) card as one coded image or 14 words, whichever is detected first. A binary (PUNCHB) card is considered 15 words; column image (PUNCH8), 16 words.

When the value in bits 0 - 29 of the appropriate count word has been reduced to zero, the count is exhausted. The CTLM caller (e.g. CIO/2WD) or the PPU program which is reducing the count itself (e.g., DMP) is responsible for setting the control point error flag when the count is exhausted. 1AJ and 2EF are activated. 2EF adds 100_8 times the line reserve value in bits 0 - 17 of control point word RRSW (62_8) and proceeds normally. There is no card reserve value.

At job termination all the count values - CPU time, I/O transfer units, tracks, lines and cards - are translated into job accounting messages by the ADFM function processor of CPUMTR. 1CJ sends these messages to the job and accounting dayfiles. Analysis programs are later run to read the accounting files for billing functions.

Security and Authorization

The central and peripheral processors are used in Dual MACE at a number of different levels - system executive, job management, job directed processing, etc. For system security purposes, the levels are divided into a system level and a user level. Outside the standard interface areas - RA + 1 calls, XJ calls, control cards - a user level process must have special authorization in order to gain access to system level processes.

The most obvious restriction of this kind which has already been mentioned is the PPU program naming and calling convention. A user level process cannot call PPU programs whose name begins with a number (e.g. 1AJ) with an RA + 1 or equivalent XJ call. Thus most of the special system PPU program names begin with a numeric - 1AJ, 1SJ, 1RO, 1TF, etc. User callable programs begin with an alphabetic - CIO, CPM, etc.

Beyond this elementary level of control there are several other levels which permit varying degrees of access to system level processes. These accesses are controlled with a number of special flags.

1. The DEBUG flag is a one in bit zero of control point zero word six.
2. The job origin code is a two character, display code value in bits 48 - 59 of control point word ACTW + 3 (44_8).
3. The job eighth name character is stored in bits 12 - 17 of control point word JNMW (21_8).
4. The control card loaded flag is stored in bit 24 of control point word SNSW (26_8).
5. The sensitive data flag is stored in bit 25 of control point word SNSW (26_8).
6. Job permission flag bits are stored in the job accounting header, ACTW - ACTW + 5 (41_8 - 46_8).

Special access requests are granted to user level processes only if some or all of these authorization flags have proper values. The flags tested depend on the access requested. A location-free (zero-level) PPU overlay, ØCA,

is available to PPU programs for testing the flags and taking appropriate actions. CPU programs can access the control point area flags of their own control points with XJ, "RELREAD" calls to CPUMTR.

The DEBUG Flag

The DEBUG flag is a global, system test status flag. In general when it is on, all restricted processes are available. This flag permits the performance of system edits, absolute memory reading via XJ calls, etc. It is usually turned off when the system is processing user jobs. The DEBUG flag status is changeable only via console keyboard commands to DSD.

Job Origin Code

The two character job origin code describes the mode of origin of a job. It is stored in the accounting header of the job input file when it is written. The job origin code may assume the following values:

Code	Significance
"B"	Batch - e.g., local card reader
"C"	Console - via a DSD command
"G"	Generated
"P"	PROCSY - remote keyboard console
"X"	Remote batch devices

A job obtains console origin if it was created with a console command entry to DSD. Jobs of this origin generally have access to all system processes. Other job origin codes carry no authoization privileges.

Job, Eighth Name Character

Special system control point processors are often identified to the system with the setting of a value in bits 12 - 17 of control point word JNMW (21_8). Since these bits adjoin the seven character job name, they are called the eighth job name character. Some special values are defined:

Value	Definition
01	BATCHIO
03	Temporary
20_8	System
60_8	PROCSY 2.0 (MESA)
77_8	Remote batch

In general, a non-zero eighth name character authorizes access to most system level processes. It also prevents the job scheduler, ISJ, from rolling the control point out, hence its use for the "spooling" processors such as BATCHIO, PROCSY 2.0 or the remote batch executive.

Control Card Loaded Flag

When IAJ loads a CPU program from the system CPU program library, it sets the control card loaded flag. Presence of this flag gives the system CPU program access to many system level processes. Two special cases in the use of this flag involve the relocatable program loader and the procedure execution program. Each clears the control card loaded flag before giving control to the user level process.

Sensitive Data Flag

Processes which store sensitive data in the central memory assigned to a control point can set the sensitive data flag in the control point area. When IAJ detects that this flag is set, it clears (zeroes) all of the central memory before advancing to the next control card. This feature is used, for example, by the PFILES permanent file processor when it has transferred portions of the system accounting file to a central memory buffer. The sensitive data flag effects the destruction of the data before, for example, a subsequent control card can dump memory.

Permission Flags

The job accounting header provided by TJC contains a set of single bit access or permission flags. TJC obtains them from the system accounting files. They describe processes which are permitted or denied to the account code user - e.g., usage of permanent files, plotters, graphics devices, etc. The individual permission bits are tested by the PPU and CPU programs using the resources regulated by the bits.

Access Flag Management

These special access flag values are managed at the system process level. The DEBUG flag and console origin both depend upon console keyboard entries. Other flags are set by system processors - IAJ sets the control card loaded flag. In addition, each time IAJ advances a control card it manipulates the authorization flags. It clears memory if the sensitive data flag is set. It also clears the sensitive data flag, the control card loaded flag, the temporary indefinite limits flag and the temporary eighth name character.

Special Authorization Usages

The following system accesses are some of those controlled by the system access flags:

1. System PPU and CPU program library edits require that the DEBUG flag be on. If it is not, the console operator must authorize the edit with a command entry.
2. Sensitive file accesses are regulated. These files include the system dayfiles, the accounting files, and all permanent files stored in directories whose names begin with the characters 5, 6, 7, 8 or 9. The authorization depends on the access mode:
   a. FILES, ATTACH, etc. control cards require console origin or operator authorization.
   b. Control card loaded permission or the DEBUG flag authorize all other requests.
3. CPUMTR requires console origin, control card loaded or DEBUG flag authorization for absolute memory read, XJ ("ABSREAD") requests.
4. Numerous other, CPU program callable PPU programs test the authorization flags, either globally or function by function. These include:
   CPM - the control point manager - selected functions
   LFM - the local file manager - selected functions
   SFP - the system function processor - all functions
   In general, these programs require console origin, control card loaded, or eighth name character authorization.
5. With operator authorization, a CPU program can obtain many of the special authorization permissions - e.g., the origin code can be changed to console, a non-zero eighth name character can be set, etc.

Console Security

Some commands which can be entered at the system console have the potential to severely damage system operation. For example, one can alter central memory contents, roll-back all rollout files to the input queue, checkpoint, etc., with single commands. In order to prevent the accidental entry of these commands, the system display driver operates in two modes - locked and unlocked.

When DSD operates in locked mode, the console operator cannot even enter prohibited commands. The command syntax definition itself is unavailable. When operating in unlocked mode, DSD will accept all command entries. Among commands legal only in the unlocked mode are those which manipulate other security access flags - e.g., it is not possible to turn on the DEBUG flag if DSD is in locked mode.

Special Job Execution Facilities

A number of other special job execution facilities enhance the processing and sequencing of jobs. Among these facilities not already discussed are conditional statement control, special control card advancement, exchange package management, and display buffer processing.

Conditional Statement Control

During the execution of a job it is sometimes desirable that the sequence of control card processing be altered by the results of key control card processes. Error dumps are not required, for example, when the error resulted from compiler detected mistakes. Two facilities of Dual MACE effect conditional control card usage.

The first is the "if-conditional" control register word (IFCW or 51_8) of the control point area. This word contains six registers of the following format:

Bits	Register	Value
59-54	E	Last error flag
53-48	SØ	Processor error count
47-42	S1	Six bit register
41-36	S2	Six bit register
35-24	R0	Twelve bit register
23-12	R1	Twelve bit register
11- 0	R2	Twelve bit register

Using appropriate calls to CPM, the PPU program control point manager, CPU programs can store values in these registers for subsequent programs. Those programs can read the register with calls to CPM or the "RELREAD" XJ function to CPUMTR. The special register E always contains the last error flag value. It is set by 1AJ/2EF. Most compilers and assemblers report their error count in SØ.

CPU programs may use these registers by directly storing, reading and testing them. In addition, a control card language processor, MCL (MACE Conditional Language) [18] is available for manipulating the registers. MCL processes the GOTO, "*-label", SET (register) and IF control cards. They permit testing and control sequencing operations. The IF control card also permits the testing of values other than the registers - the DEBUG flag, the time or date, line count, etc.

Special Control Card Advancement

MCL makes use of a special CPM function which permits a CPU program to advance through its control card statements in much the same manner as 1AJ advances through them. Thus MCL can skip to the "*-label" cards as directed by GOTO statements. Another CPM function permits MCL to send a control card statement to 1AJ for execution. Thus, for example, the IF control card may also contain a statement to be executed if the tested condition is true, in much the same manner that the FORTRAN, IF statement functions. For example, a job wishes to execute a compiled program on file LGO if no compiler errors were detected. The following control card effects that operation:

IF(SØ.EQ.0) LGO.

The control card advance function of CPM is also used by CPU programs which process a number of different control card calls. For example, the system CPU program UTILITY, contains entry points for the processing of forty-seven different control cards (REWIND, RETURN, COPY, FILES, ATTACH, etc.). By reading the next control card with CPM, UTILITY can process a series of its own calls in a much more rapid fashion than can be achieved with individual reloads of UTILITY by 1AJ. System overhead is also reduced.

Exchange Package Management

A number of the XJ function calls (Appendix D) processed by CPUMTR for control point CPU programs permit the manipulation and management of exchange packages. These operations are useful in effecting register dumps, subroutine linkage, interrupt processing, etc. The exchange package can be stored in an area, swapped with an area (a pseudo-exchange jump) or swapped with the insertion of a new value for the instruction counter, P.

A word of the control point area, PRPW (55_8) also enables CPUMTR to maintain exchange package stacks in the calling program field length. PRPW contains three, eighteen bit values which define the beginning (bits 36-53), end (bits 0-17) and current position (bits 18-35) of the stack. Three XJ functions are available for defining the stack, putting exchange packages on the stack, or removing them from the stack. These functions enable recursive processors to save the twenty four operating registers from level to level.

The Control Point Display Buffer

CPU programs can communicate with the system console operator with two message lines of the control point area - MS1W (30_8-34_8) and MS2W (35_8-37_8). Special system utility programs often require a larger communication area. This area is provided by the DSD, control point display buffer feature.

Control point word DBAW (66_8) defines three display buffer control addresses. One address defines the beginning of a right screen buffer (bits 18-35); another, a left screen buffer (bits 0-17); and the third, a keyboard entry buffer (bits 36-53). With appropriate console command entries (the "K" and "L" display commands) DSD will display the contents of the defined buffers on the right and left screens of the system console, and will direct operator entries to the addressed keyboard entry buffer.

Display buffer control words select character size (64 or 32 per line) and mode (vector or character). The CPU program which uses this display facility can enter data in the buffer directly and scan the keyboard entry buffer for incoming data. Optionally, the program can use a PROCSY interface which defines one special TCB as a console display buffer terminal. When the program uses the facility it communicates with the buffers in its field length with CIO calls. CIO uses the 2DS overlay to perform data transfers. While this process is somewhat cumbersome, it does permit the standard, MESA terminal processors to communicate with the system console in exactly the same fashion as a remote device. The PROCSY master terminal is assigned to the system console in this fashion.

CPU programs can communicate with the system consc
two message lines of the control point area - MSW (30
(35$_8$-37$_8$). Special system utility programs often requ
nication area. This area is provided by the DSD, consti
buffer feature.

Control points word DSAW (66$_8$) defines three displ
addresses. One address defines the beginning of a righ
(bits 18-35); another, a left screen buffer (bits 0-17)
a keyboard entry buffer (bits 36-53). With appropriate
entries (the "K" and "L" display commands) DSD will di
of the defined buffers on the right and left screens o
role, and will direct operator entries to the addressed
buffer.

Display buffer control words select character size
and mode (vector or character). The CPU program which
facility can enter data in the buffer directly and sca
buffer for incoming data. Optionally, the program can
face which defines one special TCB as a console displa
When the program uses the facility it communicates wit
field length with CIO calls. CIO uses the ZBS overlay
fers. While this process is somewhat cumbersome, it a
dard, MCSA terminal processors to communicate with the
exactly the same fashion as a remote device. The PROC
is assigned to the system console in this fashion.

Remote Devices - PROCSY 2.0

A large share of the work performed by the Dual MACE system depends upon communication with remote input/output devices. These range from slow (10 characters per second) teletypes to very fast (90,000 characters per second) tape drives. All of this remote communications is managed and directed through a multi-level system called the Purdue Remote On-line Console System, PROCSY [17]. Currently version 2.0 is in use.

The PROCSY system is divided into two main hardware levels. The 6400 and 6500 systems, controlled by Dual MACE, are responsible for the computing, on-line storage, and active control functions. Three front end computers perform the input-output communications tasks. These include an IBM 7094 which communicates with as many as 64 low speed (10 to 30 characters per second) keyboard devices. A second 7094 communicates with a high speed batch station at 20,000 characters per second, a small number of keyboard devices, and four 90,000 characters per second 729 IV tape drives. Third, a Modular Computer Systems MODCOMP III communicates with a number of high speed, 1,000 character per second devices (display consoles, line printers, card readers) a number of low speed keyboard devices and with remotely located computer systems (DEC PDP 11/45, IBM 360/22) over both synchronous and asynchronous communications facilities.

The three front end computers are coupled to one channel of the 6500 with channel-to-channel couplers. The I/O on this channel is serviced by the inter-machine communications stack processor PPU program, IIM. The control programs of each of the three front end machines present a standard software interface to IIM.

Two software mechanisms in the Dual MACE system control the activities of the remote devices: The terminal control block (TCB) table, contains, one, three word entry for each remote device. The words of this TCB entry provide for device status and function activities, processor scheduling (swapping), interrupts, and device reservation. The processing of the TCB's, for the most part, is strictly controlled by the terminal systems CPU program, MESA, assigned to one or more dedicated control points. In certain special cases - the 729 IV tapes, for example - TCB control is managed by IIM and other CPU programs. A special CPUMTR, XJ function supports MESA swapping activities (Appendix E).

Terminal System Control - MESA

The MESA system runs at one or more control points that must remain dedicated so long as there is any associated terminal activity. 15000_8 words of main memory are allocated to a MESA control point. About 2000_8 words make up the MESA resident executive routines. Another 4000_8 words are used for MESA processor program overlays that are normally loaded from ECS. The remaining 7000_8 words make up the user page area. The page area contains information that is specific to an individual user. Information in the area may be data or user program or a combination of program and data. The user page also contains any tables or

other information specific to the individual user that are created by either the resident executive or the processor programs. The user page is the area that must be swapped when the user is temporarily interrupted, and that must be reloaded when that user's processing is resumed.

Because of the small size of the MESA main memory, the processors that operate in the MESA system are carefully designed and structured so as to consist of sequences of relatively small overlays. An active MESA processor program is stored as a set of segments in a segment library in ECS (the MLD). All processor program code is reentrant.

Since ECS storage is limited and system performance is directly related to the speed of swapping, the amount that is to be swapped is kept to a minimum. Only that part of the user area that is actually in use (i.e., the user page) is swapped out when a program is interrupted. Within the user page there is a list of the names of the segments that were in the program area at the time of interruption. Since the program segments are all pure procedure segments, the program area can be reloaded from ECS when the interrupted program is to be resumed and there is no need to swap out the contents of the program area.

File Table

A program that is running at a MESA control point may have up to 24 local files. In order to speed up swapping to and from ECS, a fixed block of 24 entries in the File Name Table (FNT/FST) is permanently allocated to each MESA control point. These entries are always associated with the job's active data page. The local file block is swapped out along with the user page when the job is interrupted, and is swapped back in when it is resumed.

LOGON

Every terminal that is logged on in the system has an active data page. The active data page is initiallialized by the LOGON processor. The LOGON processor first performs validity checks on name, account number and optional password, using a call to the job card translator PPU program TJC. If the user's access is valid, it places the user name and account number information in the data page along with the address of the user's primary file storage block. It also initializes to zero the cells in which the user's processor time and I/O utilization will be accumulated. This LOGON and accounting information area remains with the user's active page throughout his terminal session until he logs off.

After the LOGON processor has finished its initialization of the data page, it asks the user for a processor name, verifies that it is a valid one and proceeds to load it and to give it control. LOGON links to the processor with a procedure called PLINK.