DVA

P800M PROGRAMMER'S GUIDE 3

VOLUME I : MULTI APPLICATION MONITOR

A Publication of :
Philips Data Systems,
Department SSS, T & D
P.O. Box 245, 7300 AE Apeldoorn, The Netherlands.

Publication Number 5122 991 28375

October 1983

This Manual, Volume I of a set, describes the user interface with the Multi-Application Monitor; it has been written for use by system managers, programmers and operators. It should be read together with the other volumes in the set:

Vol. II:	Instruction Set
Vol. III:	Software Processors
Vol. IV:	Trouble Shooting Guide

the last of which describes the internal workings of the Monitor. A pocket-sized summary, the MAS Reference Data booklet, is also available.

Since the last edition, this book has been completely altered in its layout, with the object of making it easier for the user to find information he requires. The Manual is now divided into Parts:

Part 1:	General Description
Part 2:	The System Machine
Part 3:	The Background Machine
Part 4:	Foreground Machines
Appendices.	

Reference to the Table of Contents should enable the user to locate his required information quickly and easily. The text has also been extensively revised to remove numerous minor errors, mostly typographical or caused by changes in a new Release.

While every care has been taken in the preparation of this book, some errors may remain. Should the reader find an error or omission, or have any other comment to make, he is invited to contact:

SSS, Training and Documentation,

at the address on the opposite page. A form is provided at the end of the book, for the user's convenience.

Page

Page

Page

Page

PART 1 GENERAL DESCRIPTION

MAS is a powerful operating system, combining the features of the Batch processing and real time processing in one central machine. MAS has been designed on a hardware feature called Memory Management unit (MMU), which provides the ability to use relative addressing. Its working is described in the following chapter (General Principles).

In MAS a virtual machine concept is used. In one physical machine - one P800 configuration - several virtual machines can be defined. The MAS monitor itself runs in a real time processing entity, known as the System Machine . This System Machine can support one Background Machine and one or more Foreground Machines .

THe System Machine is defined at system generation time; the other machines can be defined at any time by the user. The machines are described in Parts of the same name, which appear later in this manual.
MAS is particularly suited to:

- Multi-tasking applications. The large memory size allows several simultaneously memory-resident programs and data areas, with good response time and overall performance.
- Data communication applications. The large memory size allows a large number of buffers and tables to be memory-resident for data entry, verification and collection.
- Foreground/background applications. Batch, real-time, on-line and process control activities can be performed concurrently.

Allocation of peripheral and processor resources is controlled by a system of hardware interrupts and software priorities. These determine which application within the foreground or background machine shall have a particular resource. This allocation is dynamic, and a re-allocation of resources will occur when a greater need is determined by MAS.

Communication between all the programs in one machine is possible; this is achieved by the provision of common areas of memory. Foreground and background machines are directed by the user by use of System Command Language (SCL), Foreground Command Language (FCL) and Background Command Language (BCL) respectively; SCL and FCL commands are processed within the system machine and BCL commands within the background machine, if present. These commands may be catalogued, and as such they are known as Catalogued Procedures. The following Chapter (General Principles) outlines the parts of this manual of particular interest to the individual reader. System managers, programmers and operators may each find specific information necessary for their successful use of MAS.

HARDWARE CONFIGURATION

MAS requires the following configuration in order to run:

- a P857, P858, P859, P854 or P876 CPU with Memory Management Unit and at least 64K words of memory, up to a maximum of 128 Kwords for a P857 or P858 configuration and 512 Kwords for a P859, P854 or P876 configuration.
- a console typewriter for the system manager and system operator;
- one or more discs, other than flexible discs;
- a line printer;
- magnetic tapes, cassette drives, card reader, video displays, etc., as required by the application programs to be run.

MACHINES

MAS consists of a number of virtual machines, each of which is a separate processing environment. A P800 configuration under MAS supports a System Machine, one or more Foreground Machines and optionally a Background Machine.

Applications are designed as either foreground or background applications. Programs which are of real time nature are placed in a Foreground Machine. Batch programs can be placed in the Batch Machine (background processing) or in a Foreground Machine (middleground processing).

During a MAS session each foreground application to be executed requires a foreground machine. All background applications are run one after the other (as determined by the systems manager) in the background machine, or in a foreground machine, when middleground processing is used.

MAS controls the simultaneous performance of, and provides services for, all the foreground applications and the background application (if any). MAS performs its activities in the system machine. Each machine is defined as required at the start of or during each session. A machine consists basically of:
- A set of programs to perform the application.
  An area of memory reserved exclusively for the application.
- A Filecode Table which defines the files and the I/O devices required by the application. (Filecodes are described in Chapter 3.)

The I/O devices are available to all machines. As will be explained later, the filecode tables (one for each machine) are used to identify the devices to be used for the files required by the application.

Each user machine (foreground or background) is controlled by a user with control commands entered from an input device. The system machine is controlled by the systems manager with control commands entered from an input device. Additionally, MAS is controlled through an interactive device, by means of which control commands can be given to MAS by the operator and reply messages given to the operator by MAS.

The System Machine

Each user machine is controlled by the system machine, which is an independent machine within which MAS is executed. As well as controlling the user machines it provides service functions which can be requested by any user program via LKM (Link to Monitor) requests.

An operator console is assigned to the system machine; a wide range of operator commands is available for controlling user applications. MAS also uses the operator console to inform the operator of all unusual conditions, such as a device requiring operator intervention or repair. User programs may send messages to the operator (to mount or dismount tapes, for example) and may specify routines to be started when the operator enters an operator command.

A service program in the system machine obeys SCL (System Command Language) commands submitted by the systems manager from an interactive device. They are processed sequentially; if one is found to be invalid, then facilities are available for it to be corrected interactively. SCL commands are used to:

- Define memory, devices and DAD's for user machines, prior to starting user applications.
- Initialise the system date and real-time clock values.
- Perform system on-line debugging.

The systems manager may, during the session, obtain a map showing the resources used by each application being performed, and may instruct MAS as to the I/O devices and memory pages which may be used.

These facilities may also be used in the case of memory pages and devices which become inoperable, or where the system has been generated to support a larger number of devices than is presently installed. MAS may also be informed that floating point instructions are not used by particular programs; floating point registers will not be saved when these tasks are interrupted, and this will increase throughput.

The SCL service program is loaded into the system machine automatically at the start of a session. It can be deactivated (to free the devices it uses) by the SCL BYE command, and it can be subsequently reloaded with the SM operator command, or by a BYE SYSTEM command from a foreground machine.

In addition to the SCL control program, each user machine has its own control program providing control services for its own user.

The Background Machine

The background machine executes user applications where the input entities can be batched into a file or files, which are then submitted to the computer for batch processing. The programs in a background application control the input; data is only input when a program is ready for it and has issued an instruction to read it.

As well as user batch applications, activities such as program compiling, program link-editing and file copying may be regarded as background applications.

Foreground Machines

Foreground machines execute user applications which must process input immediately as it arrives at the computer. The input may arrive as a series of isolated entities (for example, a signal from a piece of equipment used in a process control application, or a message from a VDU or teletype used in an on-line application). The input may arrive at predictable intervals (for example, equipment sending a signal to the computer every two seconds) or unpredictable (for example, a photo-electric cell sending a signal every time a vehicle approaches a traffic light). The user application may be real-time, in that the computer reacts to the sending signals which cause some external effect (for example, an application which corrects deviation from the planned flight path of a rocket). It may be conversational, in the sense that the input and output entities are in the form of sentences in a human language. It may be interactive, in the sense that humans are kept informed of the computer's activities and may continuously adjust, correct or stop them.

The main feature of foreground applications is that each input entity is received as an isolated unit by the computer and must be processed immediately, interrupting if necessary whatever tasks are being executed. Sometimes the processing that must take place is merely recording the input entities for subsequent batch processing.

Each foreground machine supports one or more applications, each of which may contain several programs running concurrently.

DADs (direct access devices)

Just like a physical machine is divided into virtual machines, the discs are divided into virtual devices, called DAD's. A DAD is the largest entity on a disc, that a user can access. It consists of an integer number of cylinders and it is accessed via filecodes. These filecodes must have a value ranging from /F0 to /FF. Filecodes are described Chapter 3.

MACHINE INDEPENDENCE

Memory consists of up to 256 pages. Some pages (depending on the options specified at system generation time) are reserved exclusively for the system machine. Each of the remaining pages may be reserved during any one session exclusively for one user machine (foreground or background), or it may be placed in a pool of pages which may be used as required by any user machine. In the latter case, MAS ensures that no page is allocated to more than one machine at a time.

Devices and disc areas may theoretically be shared between machines merely by assigning filecodes in several machines to the same device or disc area. It is possible to envisage disc areas being used for inter-machine communication. In practice it is easier to control the machines if no sequential devices (apart from consoles and possibly the line printer) are shared between machines. This remains true even if the operator, the systems manager and all the foreground and background users are in fact one person.

LINK TO MONITOR

Programs in foreground and background machines may request monitor services by issuing LKM instructions. The steps necessary to issue an LKM depend on the LKM involved and, moreover, whether it is issued by a foreground or background machine. The sequence of steps required to issue a particular LKM is described in Appendix C.

To summarise, the processing of LKMs involves an LKM sequence which must include the LKM instruction and in most cases one or more of the following:
- Registers A7 and A8 to be loaded with values.
- A parameter value.
- An associated Scheduled label. (Scheduled labels are described later in this Chapter.)

Scheduled labels may only be associated with some LKMs, and must not be associated with others. With some LKMs, particularly when issued from the background machine, scheduled labels are irrelevant or their use is not meaningful, e.g. LKM 3 (Exit) and LKM 46 (Abort).

The Chapters on foreground (10,11) and background (9) machines and Appendix C give details for each LKM call, describing the required values of Registers A7 and A8, scheduled labels and parameters.

HARDWARE LEVELS

The P800 series of computers utilises a 64 level hardware interrupt structure, these levels being numbered 0 to 63 inclusive. The lowest levels (highest priority) are reserved for hardware functions. Level 62 is reserved for the monitor, and the highest level, level 63, is available for user programs.

HARDWARE INTERRUPTS

A hardware interrupt represents a hardware level, so that if an interrupt occurs before a previous one has been completely processed, the one with the lower hardware level (higher priority) is processed first. The addresses of routines which process hardware levels are specified at system generation time. These addresses are contained in the first 61 words of the system machine memory area.

SOFTWARE LEVELS

Up to 240 software levels but with a multiple of 30, may be defined at system generation time. The software level that a particular program occupies is determined by the system or systems manager. The factors to be considered are described in Part 2 (The System Machine).The highest level (lowest priority) is always occupied by the Idle Task, the next highest by the idle task I/O routine (if any).

Idle Task

This is a simple instruction loop which is executed when no other programs need to be executed. Execution of the idle task is always discontinued in the event of any hardware interrupt or software priority. The user may rewrite the idle task to include instructions which accumulate a count of idle time for use by another application, for example for accounting and statistical purposes.

Background

The background machine, if present, usually occupies the level below the two levels dedicated to the idle task, although it may occupy a lower level.

System

The lowest software levels, i.e. the highest priorities, are reserved for the system machine at system generation time. Usually it is levels 0 through 15 which are occupied by system tasks.

Foreground

The remaining levels are available to foreground machines, and foreground and middleground programs.

Allocation of Software Levels

Each program running under MAS occupies its own software level. The background machine runs only one program at a time, so background programs always occupy the same software level. The systems manager must determine the importance of foreground programs, because a foreground program with a low software level will run before a foreground program with a high software level. Middleground programs are connected by the system to the highest free software level.

DISPATCHER

This is a monitor module which distributes central processor time by starting programs according to their priority.The dispatcher cannot be entered directly by the user, but only from an interrupt routine, e.g. the I/O interrupt and monitor request handlers.

STANDARD PROCESSORS

Standard processors are utility programs provided under MAS to run as batch programs in the background machine, or as middleground programs. They are:

ASM	Assembler
FRT	Fortran IV
RTL	RTL/2 Language Processor
LKE	Overlay Linkage Editor
UPD	Sequential Disc File Update
LIB	Librarian
MAC	Macro Processor
EDF	EDFM/TDFM Processor
SRT	Sort processor

They are called into execution by giving the appropriate processor calls to MAS. A full description of each standard processor is given in the P800 Programmer's Guide 3, Vol. III: Software Processors, or in separate manuals.

REGISTERS

The following terms which appear in this manual are basic to the P800M and should be understood by the reader.

P-Register (A0)

This is a single 16-bit register containing the relative address of the next instruction to be executed in a program.

Registers A1 to A14

General purpose 16-bit registers available to user programs.

Registers A7 and A8

These are single 16-bit registers which are loaded with data such as addresses and numeric values by the user prior to issuing various LKMs. In most cases, the contents of register A7 indicate to the user successful or unsuccessful completion of the LKM. If an LKM has not been completed successfully, A7 usually contains a status value indicating the nature of the failure.

SYSTEM INTERRUPT STACK (A15)

When an interrupt occurs, certain information about the interrupted program or routine must be saved before the interrupt can be serviced. This is done in the system stack, the address of which is held in register A15. The start address of this stack is defined at system generation time. The stack extends downwards in memory, i.e. towards the lower addresses. Register A15 always points to the first free location in the stack. Normally the highest address in the stack is address /2FE and the lowest address is /100.

During an interrupt, the Program Status Word (PSW) and the P-register (A0) are always saved by the hardware. Additionally a number of registers (normally 8) and other values can be saved by the interrupt routine. The interrupt routine resets all saved registers and values by itself, except when it returns via the dispatcher. The dispatcher expects 8 registers (A1-A8) the PSW and the P-register in the stack and resets it. See Fig 2.1

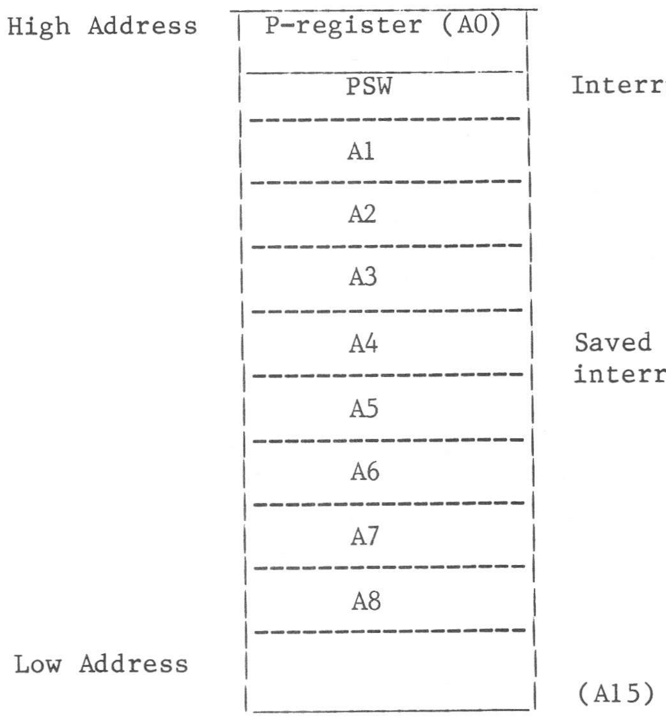


Fig. 2.1 Stack

When the stack pointer (A15) reaches or becomes lower than /100 (the last location before the stack overflow area), an interrupt occurs and the machine halts.

PROGRAM STATUS WORD

This is a hardware 16-bit register, containing the status of the currently active program running under MAS. It is copied to the user program register save area on interrupt. The layout is as follows:-

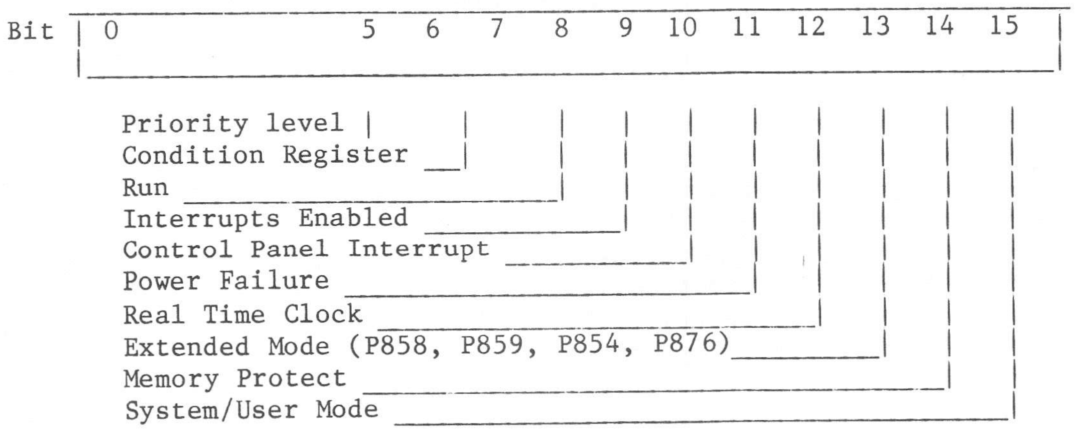


Fig. 2.2 Program Status Word

MEMORY

The basic unit of memory is the 16-bit word. The numbering convention for bits within a word is 0-15, counting from left to right. There are two bytes of 8 bits in each word; either bits 0-7 (the left-hand byte) or bits 8-15 (the right-hand byte).

Memory Allocation

Memory may be reserved exclusively for a machine or shared between machines.

Exclusive Use

For the foreground machine a set of programs may be loaded into this reserved area called segment and will remain there until the machine, segment or program is killed, or until the next IPL.

For the background machine the Batch Control Processor (BCP), Standard Processors and user batch programs will be loaded one at a time into this reserved area within the background machine.

Shared Use

Memory may be shared between user machines. Foreground programs (and optionally the active batch program) will be loaded into this shared area (known as the Dynamic Loading Area).

DYNAMIC LOADING AREA

The Dynamic Loading Area (DLA) is the area in which user disc-resident programs are executed. It is variable in size, since it consists of the area remaining after the user machines have been declared (see Fig. 2.3, Memory Layout).

A program occupying pages in the Dynamic Loading Area is copied to disc, when its state switches from active into waiting or when it consumed its Minimum Core Resident Time. It is not copied to disc, when no other program is waiting for pages in the Dynamic Loading Area. MAS automatically keeps disc accesses required for this procedure to a minimum by copying only the modified pages of a program from memory to disc.

MEMORY LAYOUT

For foreground machines memory is divided into segments. Segment 0 is addressable by all other segments in the same foreground machine. The maximum size of a segment other than Segment 0 is 32K words less the number of words in Segment 0. Within one segment, programs may address each other directly, but programs in different segments of the same machine must communicate through Segment 0.

Schematically, an example of a memory layout would appear thus:

```
          0    ---------------------------------------------
              |            Interrupt Locations              |
              |                                             |
        /7C   |---------------------------------------------|
        /7E   |                    Trap                     |
              |---------------------------------------------|
        /80   |                  Reserved                   |
              |---------------------------------------------|
        /82   |   Communication Vector Table (CVT) address  |
              |---------------------------------------------|
              =                                             =
       /100   |---------------------------------------------|
              |                 A15 Stack                   |
       /300   |---------------------------------------------|
              |          Monitor Transient Area (2040 words)|
      /12F0   |---------------------------------------------|
              =                                             =
              |---------------------------------------------|
              |               System Modules                |
              |---------------------------------------------|
              |                                             |
              |---------------------------------------------|
              |          System Dynamic Area (SDA)          |
 Page boundary|---------------------------------------------|
              |                           | )               |
 -------------|                           | )               |
       <32K   |  Core Resident Segment 2  | )               |
              |---------------------------| )               |
              |  Core Res. Seg. 1         | )  Foreground   |
              |---------------------------| )               |
            ( |  Dynamic Area             | )  Machine 1    |
  Segment 0 ( |---------------------------| )               |
            ( |  Public Library           | )               |
            ( |---------------------------|-----------------|
              |  C.R. Seg. 2              | )               |
              |---------------------------| )               |
              |  C.R. Seg. 1              | )  Foreground   |
              |---------------------------| )               |
            ( |  Dynamic Area             | )  Machine 2    |
  Segment 0 ( |---------------------------| )               |
            ( |  Public Library           | )               |
            ( |---------------------------|-----------------|
              |  Batch Machine                              |
              |---------------------------------------------|
              |  Dynamic Loading Area (DLA)                 |
              |---------------------------------------------|
```

Fig. 2.3 Memory Layout

MEMORY MANAGEMENT UNIT (MMU)

Memory management is effected by means of the hardware Memory Management Unit, which in turn is software controlled by the systems manager using SCL commands submitted during the definition of user machines, although, as previously mentioned, memory can also be dynamically allocated and released by user programs using LKM requests. The MMU allows a paging system to be used, which provides user programs with the facility of virtual addressing within 16 bits, although the absolute machine address of the program page may exceed this.

Physical Addresses

A physical byte address is 16 bits long, giving an address range of 0-65535 bytes. Bits 0-14 of the physical address identify the word (0-32K) and bit 15 identifies the byte (0 for the left byte and 1 for the right).

The MMU allows a program to be loaded anywhere in the physical memory. Without it, programs would always have to be loaded into memory module 0 and memory modules 1 upwards would remain inaccessible. The MMU translates the relative addresses within programs into physical addresses within particular memory modules.

Paging

The total memory area is divided into pages of 2K words. The maximum numbers of pages are 64 for P857 and P858 and 256 pages for P859, P854 and P876 (the P854is intended to support 1024 pages). When loaded into the machine, a program needs not to occupy contiguous pages; the system loads each 2Kw block of program into a free page and enters the page address into the program's page table. Each relative program address must now be regarded as having the following format:

Bits 0-3:	Contain the relative page number in this program containing the byte to be addressed. Up to 16 pages are possible, giving a maximum size of 32 KW per program.
Bits 4-15:	Contain the displacement of the byte within the page.

The Page Table

The Page Table is the link between the relative addresses in a program and the absolute address of the memory pages. Whenever a program is declared, a page table of 16 MMU registers is created for it. When the program is loaded, the MMU registers are filled. Each MMU register has the following format:

Bits	Contents
0-5:	For P857/P858 the pagenumber, for P859/P854/P876 the least significant bits of the pagenumber.
6:	Set to 1 when the page is not connected to the program. An access of a program to such a page gives a page interrupt.
7:	Set to 1 when the page is read only. A write of a program to such a page gives a page interrupt.
8:	Set to 1, when the contents of the page have been modified, since it was loaded.
9:	Not used.
10-13:	For P858/P859/P876 not used, for P854 in future these bits, together with the bits 14 and 15 will be the most significant bits of the page number.
14-15:	Most significant bits of the pagenumber for P859/P854/P876.

For the P857, the bits 10-15 contain the time the page was loaded.

Using a program with a length of 3 pages and 22 bytes as an example, MAS could construct the following table:

virtual page nr	page table entry									
bit	0	1				5	6	7	14	15
0	1	1	0	0	1	0	0		0	0
1	1	1	0	0	1	1	0		0	0
2	1	1	1	1	0	0	0		0	0
3	0	0	0	0	0	1	0		0	1
4	x	x	x	x	x	x	1		x	x

Fig. 2.4 Page Table

In this example, the program consists of the pages /32, /33, /3C and /41 (the last page is not possible in case of P857/P858). A relative program address /201C will be translated into the absolute address /3C01C. The program address /5500 will give a page interrupt and the program will, accessing this address, be aborted with memory protect.

Absolute Addresses

The 18- or 20-bit address used by the MMU is known as an absolute address. It can range from 0 to 128KW or 512KW respectivily and identifies one byte in the bare machine. Absolute addresses are used in the parameter fields of the following operator commands:

WM	(Write Memory);
CR	(Patch Memory);
DM	(Dump Memory);

and also in the following SCL commands:

WRM	(Write Memory);
DUM	(Dump Memory).

They are entered as 5 hexadecimal digits with maximum value /FFFFF, of which bits 0-7 contain the page number and bits 8-19 the byte number within the 2K word page.

System Mode

Running in System Mode, the MMU is not used; instead 16 bits absolute addresses are used. System Mode can only be used in the first 16 pages of the memory. The MAS system runs completely in system mode, except MAS release 8 for P859/P854/P876 which runs partly in Extended Mode. In System Mode, priviliged instructions like CIO, INH and A15 using instructions can be used.

Extended Mode

The Extended Mode is a System Mode running under MMU. In this mode, priviliged instructions can be used. MAS release 8 for P859/P854/P876 runs partly in Extended Mode to overcome the 16 pages limit for the monitor.

Common Areas

Communication between programs within the same foreground machine is possible via the common `communication area' in Segment 0 of that machine. The allocation of these areas is described in Part 4 Chapter 11 (Foreground Machines).

PROGRAMS

Types

User program characteristics depend on whether they belong to the foreground or background.

Foreground Programs

The way a foreground program is declared determines whether it is Disc Resident or Memory Resident.

Disc Resident Programs

These are declared by the following commands:
- SWP which declares a Swappable program.
- RON which declares a Read-only program.

A third type of disc-resident program may exist. This is a non-declared program, known as a Middleground program.
A swappable or a non-declared program may be Overlaid. Disc-resident programs may be written as Re-entrant, . In that case they can be declared as Read-Only-Reentrant. Disc-resident programs are loaded into the Dynamic Loading Area.

Swappable programs

A swappable program must be declared and connected to a software level before activation; it can modify code or data within its own area during execution. The core image is swapped out when its minimum core-resident time is exceeded or when it turns into inactive state and another program needs memory space in which to run. Only disc-resident programs may be swapped out.

Disc Space Allocation for Swappable Programs

The unit of allocation is the granule, where 1 granule = 1 program page. A system DAD, D:CI, is allocated at system generation time to accommodate all declared disc-resident programs. This DAD consists of two parts:

1) An area provided for the initial image of swappable and read-only programs;
2) A save area into which the core images of swappable and middleground programs can be written when swapped out. They can be reloaded from this area when they reach the head of the queue of programs waiting to run.

When an overlaid program is declared swappable, only the root is written to the D:CI DAD. The overlay segments are still read in from the original load module. When the overlaid program is swapped out, the root and the current overlay tree are written to the D:CI DAD.

Program Swapping Mechanism

The decision to interrupt and swap-out a program is based on the Minimum Core-Resident Time (MCRT) for that program. When this time has been exceeded and another program requiring memory space is eligible to run, the system program X:SWIO, which initiates the swapping process, is activated.

The minimum core-resident time is defined at the time of:

- System Generation, by adapting the CVT (Communication Vector Table)
- Program Declaration. The default value is 3 seconds.

The conditions under which a program will be swapped out can be summarised thus:

1) The MCRT has elapsed and another program is waiting for memory space. When there is no other program waiting for memory space, the program remains in core and the MCRT is set to its initial value.

2) The MCRT has not elapsed, but the program issued an LKM 2 (wait for event) with an odd value in A8 (ECB address). The program is swapped out, regardless whether there are programs waiting for memory space. The program is not swapped, when event waited for, modifies the pages to be swapped out.

3) The MCRT has not elapsed, but the program issued an LKM 2 with an even value in A8. The program is only swapped out when another program is waiting for memory space. The program is only swapped when the event does not modify pages to be swapped out.

4) When the program issued an LKM 6 (pause).

5) Programs, which have issued an LKM 3 (exit) in the main sequence are not swapped out, but are discarded from memory.

Bits 7 and 8 of the page table entries are used by MAS to control the swapping operation. If a program is to be swapped, its page table is located via the Program Control Table (PCT) for this program and the relevant entries are examined. If bit 7 is 1, the page does not need to be swapped before being overwritten, since the core image is identical to the image stored on disc in the Core Image File. If bit 7 is 0 and bit 8 is also 0 (i.e. the page is unmodified), again there is no need to swap-out the page.

Swappable Batch Programs

These are swapped in and out via the MCRT in the same way as Foreground programs.

Read-only Programs

A read-only program may occupy memory required by another program.In this case it is overwritten. When able to resume, it is reloaded from its original core image file. Its pages are set read-only; it may therefore not modify its own area, and may only contain instructions and constants. It may modify variables in segment 0. Read-only programs are indicated by Bit 7 of their page table entry being set to one. A read-only program cannot be overlaid.

Non-declared Programs (Middleground Programs)

Non-declared programs are identical to disc-resident swappable programs, except that they are not declared by the user and they are connected by MAS itself, not by the user. They must be catalogued in the user program library, and aretransferred to the DAD D:CI whenever their execution is requested. They are connected by the system to a low priority software level.

Memory Resident Programs

These are declared by the following commands:
- REP which declares a Reentrant program.
- LOD which declares a Non Reentrant program.

Reentrant

A reentrant program may be active for several tasks simultaneously in one foreground machine. In effect, whilst a re-entrant program is still being performed for one task another task may be using it, and so on. Re-entrant programs cannot be overlaid.

Background Programs

Since the background machine may contain only one program at a time, the way the background machine is declared determines the characteristics of any background program it may contain.

Disc-Resident Programs

These exist in a background machine when the machine is declared by the SCL DCB command with no number of memory pages given.

Memory-Resident Programs

These exist in a background machine when the machine is declared by the SCL DCB command with the number of memory pages given as greater than zero. Programs that run in the background machine may be overlaid.

OVERLAYING

A program with an overlay structure may be executed with only a part of the program resident in memory. It is the user's responsibility to decide which programs should be overlaid, and the overlay structure. Programs which occupy over 16 pages of memory must be overlaid, as this is the maximum size allowed by MAS. Programs under 16 pages in length could be overlaid if the functions of the program can be clearly distinguished, and where a large number of other programs must also be resident at the same time.

There are two types of overlay segment available for constructing overlaid programs:

a) Disc-resident overlays;
b) Memory-resident overlays (secondary load modules).

A program utilising disc-resident overlays consists of a root segment and one or more disc-resident overlay segments. The root segment remains in memory for the duration of the program's residency, while the overlays are loaded into the dynamic loading area of the machine, one at a time, as they are needed. These overlay segments may themselves be overlaid.

A program constructed of memory-resident overlays consists of a primary load module and one or more memory-resident secondary load modules. The primary load module and the secondary load modules are loaded before the program is activated, and remain in memory throughout the execution of the program. Secondary load modules may be declared as read-only and may be shared between more than one primary module.

These two methods of overlaying allow the user to make a reasonable compromise between the demands for memory space, on the one hand, and the need to increase throughput by cutting down on the high disc access time associated with programs which utilise disc-resident overlay segments, on the other. The user can save memory space by putting his overlays on disc, or increase his throughput by making his overlays memory-resident. These two methods can be combined if desired, the most frequently used overlays being made memory-resident and those which are used only occasionally being made disc-resident.

Generating Programs Utilising Disc resident Overlays

The loading of these overlays during execution is effected by means of LKM 27 requests, embedded in the user's program by the Linkage Editor. This is done during the processing of the NOD statements, which the programmer uses to structure his overlay 'tree' and which are output to the object temporary file (filecode /D5) during the compilation process.

A simple example illustrates this; S1, S2 and S3 are all overlay segments of a root module 'MAIN', and are to be loaded at the same address defined by the node EX1, thus:

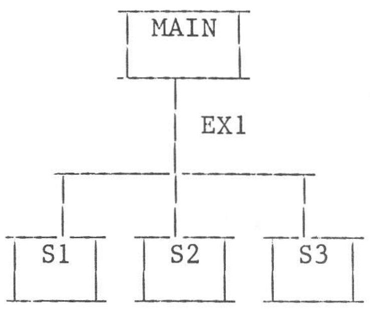
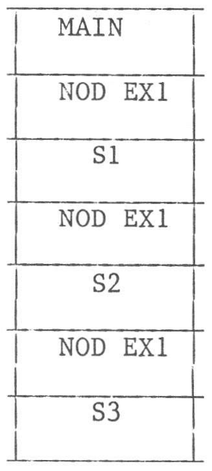


Fig. 2.5 Overlay Structure Example

The following set of commands could be used:

```
ASM
OPT  PROG=MAIN
NOD  EX1
ASM
OPT  PROG=S1
NOD  EX1
ASM
OPT  PROG=S2
NOD  EX1
ASM
OPT   PROG=S3
LKE
OPT MAP=YES,CREF=YES,CATL=MAIN
-
-
```

The temporary object module file (/D5 or /O) would appear thus:



and the temporary load module file (/D6 or /L), after linkage edit, appears as:

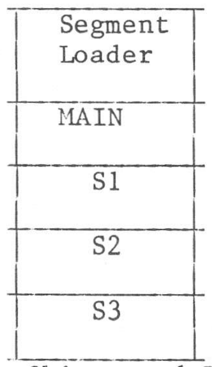


Fig. 2.6 Temporary Object and Load Module Layout Examples

Memory resident Overlaying

The memory-resident segments (secondary load modules) of an overlaid program are loaded into memory, prior to activation, by means of the FCL LSM command (Load a Secondary Module) for Foreground programs and by SCL LSM command for Background programs. Once loaded, they remain in memory until deleted; although the total memory space occupied can exceed 32K, the longest overylay path cannot.

The program's 32K 'window' is altered by modifying the MMU page table to include each module as required (excluding those which are no longer required) by means of LKM 57 requests (Connect/Disconnect a Secondary Load Module) which have been embedded in the user program during the linkage edit process. For e.g. suppose that a primary module R1 is loaded into a machine together with secondary load modules S1, S2, and S3, and that these are linked by the node EX2, thus: #

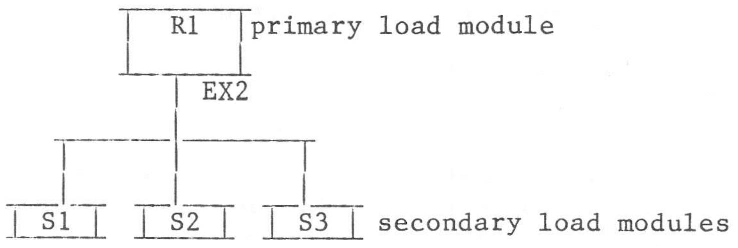


Fig. 2.7 Memory Resident Overlay Structure Example

R1-S1 may be a program path, as defined by the MMU page table, and this may use the whole of the available 32K `window', but suitable connectrequests could change this to R1-S3 and the other modules S1 and S2 would then become `invisible'. N.B. For secondary load modules, only one node is allowed for each primary load module.

Such a structure could be generated by the following run stream:

```
ASM
OPT  PROG=R1
NOD  EX2,SLM1
ASM
OPT  PROG=S1
NOD  EX2,SLM2
ASM
OPT   PROG=S2
NOD   EX2,SLM3
ASM
OPT  PROG=S3
LKE  SIZE=MAX
OPT  MAP=YES,CREF=YES,CROV=(SLM1,SLM2,SLM3),CATL=R1
-
-
```

Fig. 2.8 Secondary Load Module Generation

The user should study the descriptions of the NOD statement in Chapter 9 of this manual, the LKE OPT statement in the Software Processors manual and the FCL LSM command in Chapter 11 of this manual.

If it is intended that secondary load modules be shared between more than one primary load module, then absolute addresses should be used in the NOD and LKE OPT statements, and the modules should be declared read-only.

The secondary load modules must be loaded at page boundaries (see the description of the FCL LSM command).

Only disc-resident or non-re-entrant memory-resident programs may be overlaid.

EXECUTION

The execution of programs may be started and controlled by MAS control language (FCL or BCL) commands in two basic ways:

- Programs in foreground or background machines may be `run'. This means that execution will begin at the program's entry point, and that no further FCL or BCL commands affecting the program will be accepted until the program reaches normal or abnormal completion. (Abnormal completion means a severe or fatal error has occurred.)
- Programs in foreground machines may be `activated'. This means the same as `run' above, except that FCL commands will be accepted both before and during execution of the program.
- Programs in any one foreground machine may also be activated by other programsin the same machine.

Further details about the control language (BCL or FCL) commands can be found in Part 3 (The Background Machine) and Part 4 (Foreground Machines). LKM requests are described in Appendix C.

COMMANDS

MAS is controlled by four basic types of commands; they are given by the:-
Systems Manager (System Command Language, SCL);
- Background user (Background Command Language, BCL);
- Foreground users (Foreground Command Language, FCL);
- Operator (Operator Commands).

SCL, BCL and FCL Commands may be made into Catalogued Procedures when they are intended to be repeated in the same, or nearly the same, format. Catalogued procedures are described later in this Chapter, and are referred to in the other Chapters of this manual. The syntax rules for typing in these commands are described in Appendix B.

SCHEDULED LABELS

A scheduled label is an address of a routine which is specified in an LKM request sequence. The routine addressed by a scheduled label is known as a scheduled label routine.

Most of the scheduled label routines are executed immediality after the LKM. Some LKMs cause continuation of the main program and its interruption on completion of the procress started by the LKM. They are:

LKM -1 (I/O, without implicit wait);
LKM -12 (Activate);
LKM -25 (Read Unsolicited Key-in);
LKM -30 (Queue Handling);
LKM -77 (Accept an Attention Key).

Scheduled label routines relating to LKM requests containing the above LKMs are not processed immediately. That is to say, a scheduled label routine will never be executed until the associated LKM is complete. In effect, use is made of the time delay between any of the above LKMs being initiated and being completed.

A LKM -12 request (Activate) indicates to MAS that a program is in a state that it may be run, but the scheduled label routine associated with it will not be executed until the program concerned terminates.

An LKM -30 request (Queue Handling) refers to a queue which may be empty. In this case the scheduled label routine associated with it will not be executed until there is an entry in the queue.
An LKM -1 request (I/O) indicates to MAS that an I/O operation is required, but there may be a delay before it can be completed. In this case the scheduled label routine associated with it will not be executed until the I/O operation is complete.

An LKM -25 request does not cause program suspension awaiting the operator Key-In, but the scheduled label routine is entered when this occurs.

The main purpose in associating a scheduled-label routine with the above four LKM requests is that the routines in each case will set an event bit in an Event Control Block, i.e., after any of the above LKMs have been initiated, the related ECB can be examined. This is done by the LKM 2 request (Wait for an Event). When an LKM 2 is encountered, nothing further will be processed if the event bit has not yet been set. When the event has occurred, it is set and processing continues. This use of ECBs is described in more detail later in this chapter.

If an LKM request sequence with a scheduled label is not one of the above types, control will pass to the scheduled label routine when the LKM is exectued. After the scheduled label routine has been executed, control returns to the first sequential instruction following the LKM request sequence. For example:

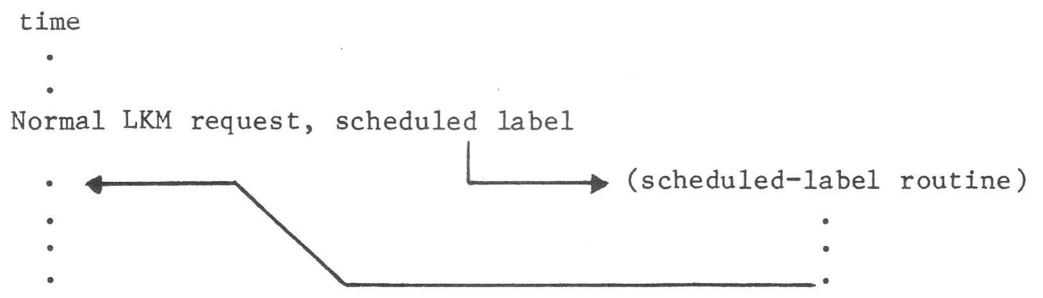

time
Normal LKM request, scheduled label
(scheduled-label routine)

Fig. 2.9 Scheduled Label Example 1

If the LKM request is one of the above, control will pass to the scheduled label routine only when the LKM request is finally complete. Where an I/O operation is involved, an interrupt will result and the time spent waiting for the I/O to be completed will be used by passing control to the next sequential instruction following the LKM request sequence. Consequently, if the user has instructions following the I/O LKM which refer to the I/O he must code a 'wait-for-event' LKM request sequence, to ensure that the I/O operation is complete before these specific instructions are executed.

In fact the scheduled label routine may be used simply to set the event bit in an ECB. After the scheduled label routine has been executed control returns to the next unexecuted sequential instruction following the LKM request sequence. The event referred to by this ECB can be the one that the user's 'wait-for-event' LKM request waits for. For example:

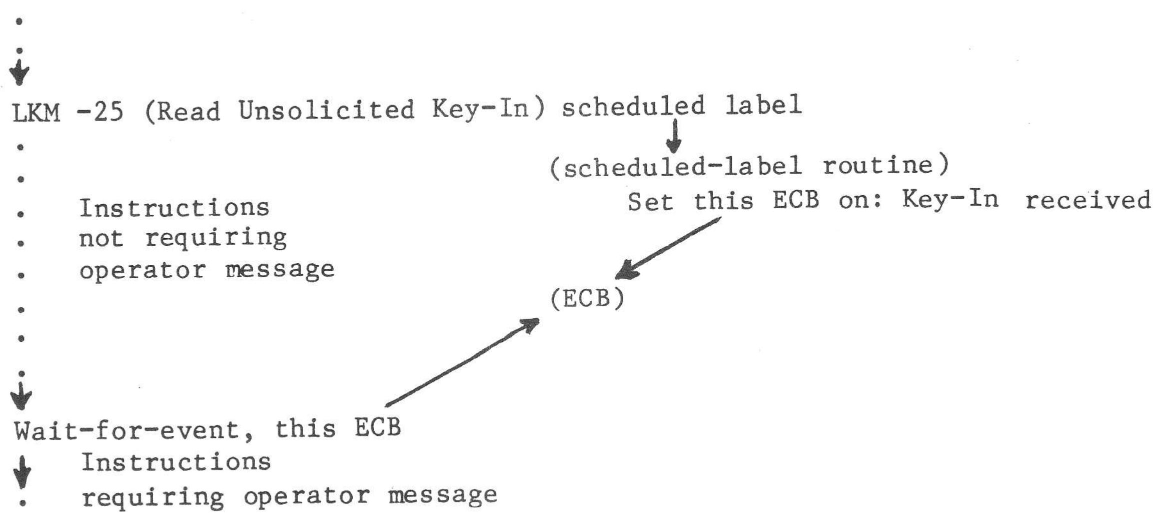

LKM -25 (Read Unsolicited Key-In) scheduled label
(scheduled-label routine)
Set this ECB on: Key-In received
Instructions
not requiring
operator message
(ECB)
Wait-for-event, this ECB
Instructions
requiring operator message

Fig. 2.10 Scheduled Label Example 2

Scheduled label routines may be nested. A scheduled label routine may contain an LKM sequence itself addressing a scheduled label routine, and so on.

Scheduled label routines must exit by an LKM 3 (Exit) or LKM 46 (Abort).

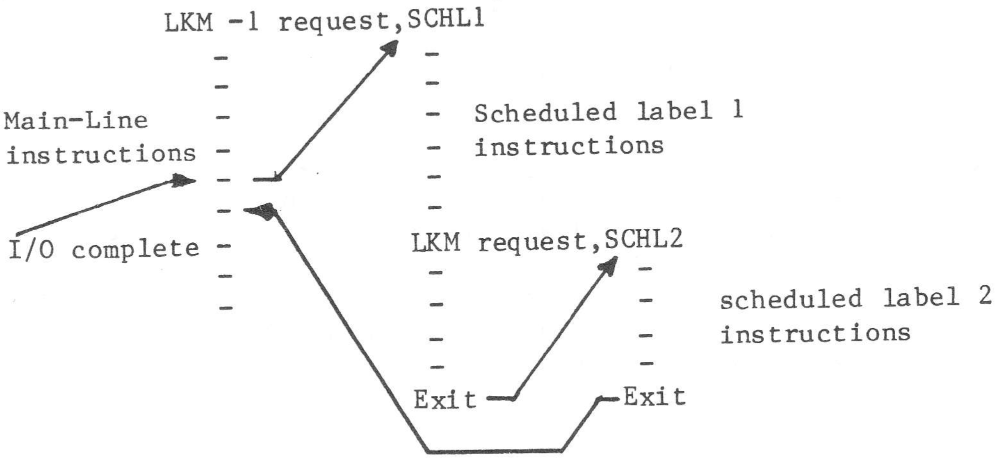


Fig. 2.11 Example of Nested Scheduled Label Routines

The presence of a scheduled label in an LKM request sequence is indicated by coding the LKM with a negative number, followed by the scheduled label.

The following example uses the DATA instruction, which codes data into object words and always follows the LKM instruction to indicate the type of LKM.

```
LKM
DATA 1
```

This results in LKM 1 being output in object code, without a scheduled label.

```
LKM
DATA -1,SCHLAB
```

This results in LKM 1 being output in object code, with the scheduled label SCHLAB.

Scheduled label routines used with LKM 3 (Exit) or LKM 46 (Abort) are ignored.

CATALOGUED PROCEDURES

Catalogued Procedures may be constructed when a number of commands are intended to be repeated in the same, or nearly the same, format. Any SCL, BCL or FCL commands can be set up as a catalogued procedure, to be invoked by a procedure call.

Catalogued procedures are similar in concept to macro definitions, and readers who have studied the P800M Macro Processor Manual will notice certain similarities in construction and operation.

In its simplest form a string of commands may be repeated sequentially, from beginning to end. When set up, the string is given a procedure name. When this is referred to by this name in a procedure call, the entire string will be processed sequentially as if it had been input manually through the console, or some other device.

In its advanced form a string of commands may contain items which are to be specified at run time by the procedure call parameters. These items may or may not be given default values. When set up, the string is given a procedure name; when this is referred to by name the string will be processed according to any modifications caused by the procedure call parameters.

In this advanced form a catalogued procedure can be used any number of times and also with any required variations.

For example, an application may be run at the end of every week. However, month end and year end may have special significance, in as much as extra programs may be required, different parameter values provided or new files created, etc.

Therefore a single string of commands can be constructed as a single catalogued procedure for all occasions. If this is done, then each time the application is run the catalogued procedure call need only specify that the run is `end-of-week', `end-of-month' or `end-of-year'.

File Identities

SCL Commands

Catalogued procedures for the system machine must be contained in a file identified by:-

DAD filecode	= /F6 of the system machine
USERID	= first user (normally MASUP)
Filename	= S:PROC
File type	= UF
Version	= 0

The DAD code /F6 is automatically included in the filecode table of the system at System Generation time.

FCL Commands

These can be reserved for the exclusive use of one machine user, or made available to all foreground users.

Catalogued procedures for the exclusive use of one foreground user must be contained in a file identified by:-

DAD filecode	= /F0 of the foreground machine
USERID	= first user
Filename	= F:PROC
File type	= UF
Version	= 0

Those which are to be available to all foreground machines should be set up in a file identified by:-

DAD filecode	= /F6 of the system machine
USERID	= first user (normally MASUP)
Filename	= S:PROC
File type	= UF
Version	= 0

BCL Commands

Catalogue procedures intended for the exclusive use of one background user should be contained in a file identified thus:-

DAD filecode	- as specified on the :JOB command
USERID	- as specified on the :JOB command
Filename	= B:PROC
File type	= UF
Version	= 0

Those intended for public use should be contained in a file identified thus:

DAD filecode	= /F0
USERID	= first user (normally MASUP)
Filename	= B:PROC
File type	= UF
Version	= 0

Procedure Set Up

The first record of every catalogued procedure must contain %% (two percent-signs) followed immediately by the procedure name of between one and six characters, not using blanks. The %% must begin in the first character position. The last record must contain PEND as the first four characters. There must be at least one Data Record between the %% and PEND record.

Data Records

Data records consist of ASCII characters, and may contain any number of Replacement Strings.
A command may be coded on more than one data record. Data records which are continued must include the continuation character ';' (semi-colon) as the last character. The second example shows the use of the continuation character.

Replacement Strings

Replacement strings have the following format:

```
@{d | n}[.v | ?]
```

d	is a number, 1 to 9999. The replacement string is replaced by a positional parameter in the procedure call.
n	is a name of 1 to 4 alphanumeric characters, the first of which must be alphabetic. The replacement string is replaced by a keyword parameter.
v	is a valid default value for the procedure call parameter.
?	indicates that no default exists, and that it is intended that a value must be provided in the procedure call, otherwise the entire data record is ignored.

If the options v and ? are both omitted, then if the corresponding parameter in the procedure call is empty the replacement string is ignored, but the rest of the data record will be recognised. For maximum lengths, see Appendix B.

Here is an example of each of the six types of replacement string:

@4	(positional parameter)
@4.10	(positional parameter with default)
@4?	(positional parameter, if ommitted, no output)
@FC1	(keyword parameter)
@FC1./30	(keyword parameter with default)
@FC1?	(keyword parameter, if ommitted, no output)

Nesting

Catalogued procedures may be nested; that is, when a catalogued procedure has been invoked by a procedure call its execution may encounter another call which invokes another catalogued procedure, and so on. Execution of the calling procedure is suspended until that of the called procedure is complete.

A catalogued procedure must never contain a call to itself or, if nested, a call to another procedure which calls the first (or calls another procedure which calls the first). A loop would obviously result if this situation occurred. A catalogued procedure may _not_ contain a :JOB, :EOJ or :EOB command.

If an error results in the above situation, MAS outputs an error message, sets the status code and passes control to the command following the erroneous procedure call.

The following illustrates the nesting of catalogued procedures:

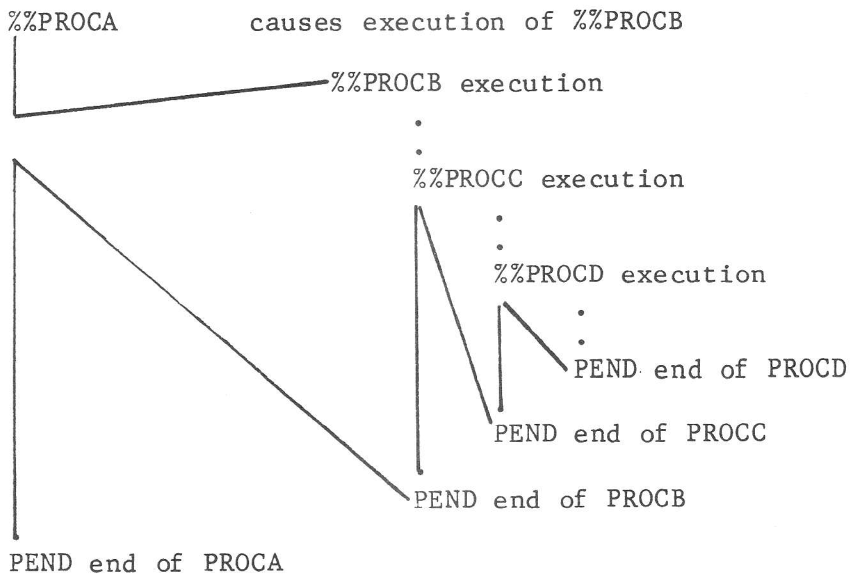


Fig. 2.12 Nesting of Catalogued Procedures

Cataloguing Procedures

Catalogued procedures may be set up in public or private libraries, by means of the standard processors Librarian (LIB) or Update Processor (UPD), described in Volume III: Software Processors.

In practice it is easier to catalogue a procedure which is already working than one which is not yet tested, and may need to be amended several times to get it right. The string of commands may be first operated manually. When they are complete, any required replacement strings may be noted on the command input device log by hand, and the procedure may then be catalogued using LIB and UPD. Some corrections may still be needed, but the likelihood will be reduced.

Procedure Calls

A procedure call refers to a previously catalogued procedure, and:
- identifies the catalogued procedure;
- specifies parameter values to be given to replacement strings in the data records of the procedure.

Procedure calls have the following format:

```
%%procedure-name [[param-1][,param-2]...]
```

procedure-name	is the procedure name, in the first record of the catalogued procedure.
param-1, etc.	are positional or keyword parameters, defining values to replace the replacement strings.

When the procedure call is executed, the specified catalogued procedure is located and its replacement strings are replaced by the parameters (if any) specified in the call.

Procedure calls are input through filecode /E0. If entered in the system machine, MAS searches the file named S:PROC. If entered in a foreground or the background machine, MAS searches F:PROC or B:PROC, respectively; if not found in this file, MAS then searches S:PROC or the system's user B:PROC.

If the procedure is found, MAS creates a work-file on disc, with filecode /ED, and writes the procedure into it, replacing the expanded replacement strings. The filecode /EE is assigned to filecode /ED and the commands are read back and executed. If a nested call to another procedure is encountered, the procedure is located, its replacement strings are processed and it is written to /ED. When the PEND statement is read, the command processor continues reading from /EE. Upon completion, control passes to the command following the first procedure call.

Call Parameters

Parameters in procedure calls are of one of two types:

- Positional	a string of zero or more ASCII characters, except blank, comma, semi-colon and equals sign.
- Keyword	a string of one to four alphanumeric characters, the first of which must be alphabetic, followed by an equals sign and a parameter value of one or more characters.

Positional parameters replace replacement strings according to their relative positions in the procedure call. That is, the first positional replacement string in the procedure is replaced with the first positional parameter in the procedure call, the second string with the second parameter, and so on.

Keyword parameters may be coded in any order within the procedure call. Each keyword parameter is uniquely identified by a key-name, of one to four characters. Every occurrence of a key-name in a catalogued procedure is replaced with the value of the corresponding key-name in the procedure call.

Replacement strings in a catalogued procedure which are not supplied with a corresponding value by the procedure call are handled as follows:

- If there is a default value, this will be used.
- If there is no default value and the replacement string contains ? (query), the entire data record of the procedure is ignored.
- If there is no default value and no ? (query), only the replacement string is ignored.

Two consecutive commas in the parameter list of a procedure call imply that the parameter is 'empty', i.e. no value is supplied. The parameter is assumed to be positional. Thus, if two commas were coded after the eighth parameter in a call, then it would be assumed that the ninth parameter was empty. The next positional parameter after this one would be positional parameter number 10, and so on.

When all replacement strings have been replaced, the resulting data record must not exceed 80 characters.

Errors in a procedure call result in an error message being output; control then returns to the command input routine of the relevant Command Processor. The next command is then input through filecode /E0 or /EE.

Examples

The examples show catalogued procedures and their related procedure calls and parameters. The first shows procedure to declare a Batch machine.

```
%%DCB
DCB @1
FCD /C0
FCD /F0,/C0,SUPERV
FCD /F2,/C0,@DAD1?
FCD @2?
FCD /F2,@2?,@D2.DAD2
FCD 1,TY10
FCD 2,@FC2.LP07
FCD /E0,TY10
DEN
PEND
```

Fig 2.13 Catalogued Procedures Example 1.

There are two positional replacement strings:

@1	This is the first one, with no default value. With default value the declaration would be e.g. @1.16.
@2	This is the second one, with no default value. If no value is supplied the two records containing @2 will be ignored.

Keyword replacement strings occur in three positions:

@D1?	The presence of the ? indicates that in this position a value is necessary; if none is given, the whole data record is ignored.
@D2.DAD2	If no other value is supplied for D2, the default is DAD2. Note that the record is still ignored if @2 is not supplied.
@FC2.LP07	If FC2 is not supplied, LP07 will be the default.

A simple call for this procedure is:

```
    %%DCB 16
```

resulting in:

```
    DCB 16
    FCD /C0
    FCD /F0,/C0,SUPERV
    FCD 1,TY10
    FCD 2,LP07
    FCD /E0,TY10
    DEN
```

A more extended call is:

```
    %%DCB 16,/C2,D1=DAD7,D2=DAD8,FC2=TY10
```

resulting in:

```
    DCB 16
    FCD /C0
    FCD /F0,/C0,SUPERV
    FCD /F1,/C0,DAD7
    FCD /C2
    FCD /F2,/C2,DAD8
    FCD 1,TY10
    FCD 2,TY10
    FCD /E0,TY10
    DEN
```

<u>Example 2</u>

The following example concerns file assignments (ASG commands).The procedure has the name PROC, and is called thus:

```
    %%PROC ALFA,,KEY1=4,BETA,KEY2=XYZ,,5,/70
```

MAS sets up the following internal table of replacement strings and their corresponding values:

STRING	VALUE
1	ALFA
2	(empty)
KEY1	4
3	BETA
KEY2	XYZ
4	(empty)
5	5
6	/70 (hexadecimal).

The catalogued procedure PROC was defined as:

```
    %%PROC
    ASG FCOD=@6,ECOD=@2./A0
    ASG FCOD=@4?,ECOD=@2./A0
    ASG FCOD=@7./60,DAD=@D./F0;
         FNAM=@KEY2,USID=@3?
    ASG FCOD=/35,DAD=/F4,FNAME=@1,VERS=@KEY1.0
    ASG FCOD=/D9,DAD=/F3,NBGR=@5?
    PEND
```

Consequently, when the catalogued procedure is invoked by the call above, the following are recognised and executed:

```
    ASG FCOD=/70,ECOD=/A0
    ASG FCOD=/60,DAD=/F0,FNAM=XYZ,USID=BETA
    ASG FCOD=/35,DAD=/F4,FNAM=ALFA,VERS=4
    ASG FCOD=/D9,DAD=/F3,NBGR=5
```

Error Messages

a) From BCP

No error message is output if a catalogued procedure data record exceeds 80 characters after replacing replacement strings by parameters.

UNKNOWN PROCEDURE
A procedure call contains %%name, but name cannot be found in any of the procedure libraries B:PROC in the :JOB userid and DAD, or B:PROC on the first userid of DAD /F0.

PROCEDURE NAME MISSING
A procedure call contains %% followed by a blank.

PARAM xxxx MISSING
A procedure call contains a keyword parameter without a value (KEY= followed by a blank, comma or semi-colon).

PARAM xxxx REDUNDANT
A procedure call contains more than one occurrence of the same keyword parameter.

KEY PARAM TOO LONG
A procedure call contains a value (for a keyword parameter) which is too long.

abc NOT ALLOWED IN A CATAL PRO
The string abc indicates one of :EOB, :EOJ, :JOB.

SYNTAX ERR:?
A catalogued procedure data record contains a replacement string which cannot be analysed.

BAD ASSIGN FILE /xx STATUS=yy
The string xx indicates a filecode (/E0, /EE or /EC) which could not be assigned while processing the catalogued procedure. The string yy gives the ECB status code (see LKM 23: Assign)

I/O ERROR FILE /xx STATUS=yy
The values of xx and yy are as in the previous message.

LINE=nnn(p) PARAM NOT PROVIDED IN THE PROCEDURE CALL
The replacement string @p in the data record nnn of a catalogued procedure has no default value (the string does not contain '.'), and no parameter was specified to replace it.

xxxx WRONG KEYWORD PAR
The keyword paramater contains a character, which is nor a letter, nor a digit.

If any of these error messages is output, the error code is set to 05 and control returns to the next command after the procedure call. These messages are output to filecode /02.

b) <u>SCL and FCL</u>

DYNAMIC AREA OVERFLOW
    There is no place in the monitor area to set up tables for executing the catalogued procedure.

SYNTAX ERROR
    Error in procedure call. The input line contains less than 3 characters, there is no blank after the procedure name or the last character of the procedure call a delimiter, but not a blank.

INVALID PROC NAME
    The procedure name contains zero or more than 6 characters.

ILLEGAL KEYWORD PARAMETER
    A keyword parameter in the procedure call contains zero or more than 4 characters or the value of the parameter contains more than 8 characters.

READ COMMAND ERROR
    An error occurred, reading the procedure call.

INVALID POSITIONAL PARAMETER
    A value of a positional parameter in the procedure call contains more than 8 characters.

ASSIGN ERROR /xx STA=/yy
    An error occurred, assigning the work files for the procedure. For an explanation of the status values, see Appendix C: LKM 23 (assign).

PEND RECORD NOT FOUND
    Reading the procedure file (F:PROC, S:PROC) an :EOF was detected.

I/O ERROR ON /xx STA=/yy
    An I/O error occurred, accessing a procedure work file. For an explanation of the status values, see Appendix C: LKM 1 (I/O).

LINE: nn BUF OVERFLOW.
    Expanding line nn of the procedure, the result exceeded 72 characters.

LINE: nn SYNTAX ERROR
    In the procedure, a line ended with a delimiter and more characters were expected, e.g. ended with a period and no default value was given.

EVENT CONTROL BLOCK (ECB)

An event control block, in its simplest form, is one 16-bit word long. Bit 0 of this first or only word is known as the Event-bit and bit 1 as the Chain-bit. The event-bit is initially set to zero by the user when the ECB is coded.

Event-Bit

When the event-bit is zero the event has not taken place, when it is one the event has taken place. The event-bit is set to one by an LKM 18 (Set Event) request which refers to the particular ECB. Elsewhere the user may code an LKM 2 (Wait for Event) request, referring to the same ECB, which will cause execution to be suspended if the event-bit has not been set to one.

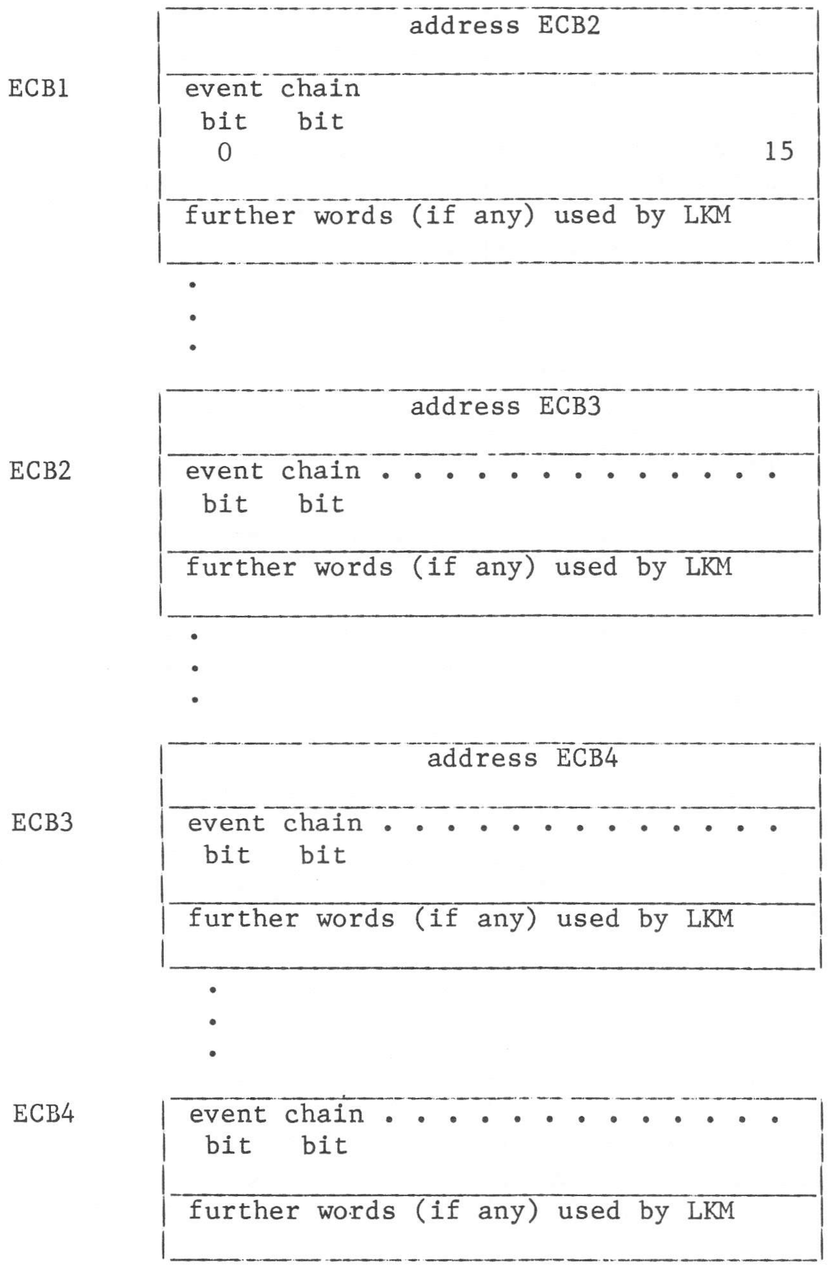
The task and the event it must be synchronised with may be in different user programs within the same machine. That is to say, a task in one program may be synchronised with an event in another program.

Bits 2 to 15 may contain data relevant to individual LKM request sequences which may refer to the ECB. In some cases extra data may be required in an ECB, and this is coded in further consecutive words. The number and contents of these extra words depends on the individual LKM requests, which are fully described in Appendix C.

Where an LKM request sequence refers to an ECB, the contents of register A8 points at the first or only word of the ECB.

Chaining

ECBs may be chained together. The construction of a chain may be illustrated as follows:



ECB1
address ECB2
event chain
bit bit
0
15
further words (if any) used by LKM
ECB2
address ECB3
event chain
bit bit
further words (if any) used by LKM
ECB3
address ECB4
event chain
bit bit
further words (if any) used by LKM
ECB4
event chain
bit bit
further words (if any) used by LKM

Fig 2.14 Chain Bit Example

As can be seen from the above illustration, the word preceding the first ECB (ECB_1) contains the address of the beginning of the second ECB (ECB_2), etc.

The chain-bit of the last ECB in a chain is coded as zero. The chain-bit of all the other ECBs in the chain is coded as one. In other words, when the chain-bit of an ECB is set to one the word preceding the first or only word of the ECB contains the address of the next ECB in the chain.

When an LKM request sequence refers to an ECB in a chain, register A8 points at the second word of the ECB. If the ECB is the last in the chain, then the first word of the ECB will be zero.

MISCELLANEOUS MAS SERVICES

Time and Date

Operator commands are available to initialise or change the date and the time.The time and date can be obtained and set by a program with an LKM (Link to Monitor) request. They can be output by the BCP (Background Command Processor) onto the BCL (Background Command Language) log.

Debugging

The contents of memory words may be printed by LKM requests, operator commands or FCL (Foreground Command Language) and SCL (System Command Language) commands. The contents of memory words may be altered by FCL and SCL commands,and in emergencies by operator commands.

Error Control

MAS traps all severe user errors, such as:
- The P-register containing the address of a word which does not contain an instruction.
- An instruction attempting to reference memory not allocated to its own program.
- A background program exceeding the time limit which the user specified for its execution.

In the event of a severe error, MAS performs actions designed to enable the user to locate the fault quickly. By using LKM 7 (Keep Control on Abort), a program can request MAS not to perform the standard actions in the event of a serious error, but to transfer control to a user routine.

The standard actions which take place depend on whether the erroneous program is in the background or foreground.

For a background program, the contents of the following are dumped to the command logging device through filecode /02:

- Program Status Word (PSW);
- Registers A1 to A15;
- P Register;
- Floating-point Registers;
- Severity-code;
- Background machine (if DUMP=PROG was given on the RUN command);
- Whole memory (if DUMP=ALL was given in the RUN command).

The BCL processor is then reloaded into the background machine, ready to accept further BCL commands.

For a foreground program, the program is placed in an abort state and a message is output on the comand logging device, containing the program name, the P-register and the severity code.

The foreground machine may now be dumped by the user or the operator. It is also possible for the user to print the registers (P-reg, PSW, A1-A15, Floating point registers) at the moment of abortion via the PRG command.

Floating Point Errors

Two LKMs allow the user to modify the standard action taken by the system in the event of a severe error occurring during the execution of a floating point instruction. These are:-

LKM 37 (Branch to User's Error Exit);
LKM 38 (Cancel the Previous LKM 37, and revert to standard error action).

These two monitor requests are described fully in Appendix C.

In case of fatal errors, a jump to a Fatal Error Halt location is performed and register A1 will contain an error code.

SPOOLING

Spooling is a term used to describe a method of processing which simulates several peripherals operating simultaneously. Under MAS the following devices may be spooled:

- The card reader;
- The line printer;
- The paper tape punch;
- The graph plotter.

Spooled input to the card reader must be in the form of a JOB stream, which is written to the card reader spool file and joins a queue of jobs waiting to be processed. When its turn comes to be processed, the job stream is read and started as soon as the :EOJ or :EOB card image is encountered. If the output peripheral has been declared as a spooled device, the output is written to the specified output spool file and queued for output behind any previously queued output. If this queue is empty the output starts immediatily after the output file has been closed.

In the case of output spooling for foreground machines, output is scheduled as soon as the Write :EOF is sent to the output spool file.

Using the Spooling Facility

Devices which are to be used for spooling must be defined as such at System Generation time, and the following DAD's must be set up:

D:SPCR	for the card reader,
D:SPLP	for the line printer,
D:SPPP	for the paper tape punch, and
D:SPPL	for the graph plotter.

Then, after machine declaration, the operator command `start spooling' (SP) can be given for the appropriate device(s).

The system also offers the following supplementary spooling commands:

- Resume spooling following an error (not allowed on the card reader).
- Delete the current spooled file (output only).
- Rewind and restart spooling (output only).
- Rewind to the beginning of the current page (output only).
- Remove the vertical format characters on the current LP output file.
- Unspool output files still present in the spool queue from a previous run.

For a full description of these commands the user should refer to the operator command SP in Chapter 5 of this Part.

TRANSIENT AREA

This is an area of memory within the system machine which is used to contain those parts of MAS which are disc-resident, and consequently are loaded when required. For example, to have all the routines associated with LKM 23(Assign) memory-resident would take up too much memory. Some MAS functions can be declared disc- or core- resident at system generation time. All transient routines run within hardware level 63, at software level 3.

ERROR LOGGING

Errors, detected during disc-, tape-, or cassette-access are retried up to 5 times. Only when an error persisted 5 times, Mas reports the user that there occurred an error on the device. Still, it might be interesting to know, whether a device is giving errors now and then, so that a hardware engineer can take preventive actions.
Therefore, Mas gives the possibility to log recovered errors in a file. For a description of this Error Logging facility see Appendix D.

FILECODES

Filecodes are used to assign program files to individual peripheral devices or disc areas. These codes are symbolic; consequently the actual device used can, within reason, be changed at run time.

For example, a program may require input data from a serial device. It may be intended that it should be the card reader; however, should the operator find it desirable, he may re-assign the input file code to the console or the paper tape reader, etc. Likewise, it may be intended that a program may output data to the console; this output file may at run time be re-assigned to the printer or papertape-punch, etc.

Clearly this feature of MAS is not universal. Some complications may possibly result if, say, a papertape-punch is assigned to an input filecode.

Certain filecodes are reserved for MAS and have special significance. A list of Reserved Filecodes appears later in this Chapter.

The link between programs and devices is the Filecode Table. Under MAS each machine contains a filecode table which defines the devices required by the application running within the machine. A maximum of 255 filecodes per machine can be declared. The filecode table consists of 4 word chained entries in the System Dynamic Area.

Reserved Filecodes

The following Filecode Reference List gives the reserved filecodes and indicates within which machine or standard processor they are used.

- Machines are shown as SYS for the system machine, FGR for foreground machines and BGR for the background machine.
- Standard processors are shown as ASM for Assembler, FRT for Fortran, LKE for Linkage-Editor, UPD for Update, LIB for Librarian.

The reserved filecodes are:

/01 and /02
/C0 thru /CF
/D0 thru /D9
/E0 thru /E2
/EC thru /EF
/F0 thru /F6 and /FF

The remainder are available to the user.

MAS FILECODE REFERENCE LIST

		Machines			Standard Processors				
FC	Used For	SYS	FGR	BGR	ASM	FRT	LKE	UPD	LIB
01	Error Messages and Corrections	+	+	+	+	+	+	+	+
02	Command Logging and Print Output	+	+	+	+	+	+	+	+
C*	Physical Disc Drives	+	+	+					+
D0	Work File			+				+	+
D1	Work File							+	+
D2	Work File								+
D3	Work File Type UF					+		+	+
D4	Work File Type SC		+	+	+	+		+	+
D5	Work File Type OB		+	+	+	+	+		
D6	Work File Type LM		+	+			+	+	
D7	Work File						+	+	
D8	Work File						+		
D*	Temp. Files								
E0	Command Input	+	+	+	+	+	+	+	+
E1	ASCII Data Input				+	+			+
E2	Binary Data Input								+
EC	Cat. Proc. Input Library	+	+	+					
ED	Temp. File for Cat. Proc.	+	+	+					
EE	Generated Cat. Proc.	+	+	+	+	+	+	+	+
EF	Operator Commands and Messages	+							
F0	System DAD (SUPERV) or	+		+			+		
	User DAD with F:PROC		+						
F1	D:CI File	+							
F2	DAD D:SPCR	+							
F3	DAD D:SPLP	+							
F4	DAD D:SPPP	+							
F5	DAD D:SPPL	+							
F6	System DAD with S:PROC	+							
F*	Other DAD Codes	+	+	+	+	+	+	+	+
FF	D:MSEG File	+							

CONTROL TABLES

MAS maintains internal tables for each machine and each program within each machine. These are used to ensure that commands and LKM requests are efficiently processed. The construction of these tables ensures that decisons affecting machines and their programs are taken by MAS depending on the resurces they require on the one hand, and their relative importance on the other hand.

DEVICE-TYPE CODES

Device-type codes are two-character mnemonics given to peripheral devices, as:

CR	card reader
DK	disc
FL	Floppy Disc
LP	line printer
MT	magnetic tape
PL	Plotter
PR	high speed tape reader
PP	high speed tape punch
TK	cassette
TY	console/typewriter
NO	Dummy assignment
DD	DAD
DY	Display connected to AMA8.

These mnemonics are used within language and operator commands when it is necessary to indicate a particular type of peripheral device.

DISCS

Types

Five types of disc drives are available to be included in P800 configurations: the X1215 or X1216 disc, the CDC-SMD 40M or CDC-SMD 80M disc,the CDC-CMD discs, the DDC fixed head disc and the 0.25M or 1M Flexible (or Floppy) discs.

Direct Access Device (DAD)

Areas of disc storage are known to MAS as Direct Access Devices (DADs). This term should not be confused with the physical disc unit. In the same way that one physical computer configuration can contain several `machines´, each physical disc device can contain one or more `DADs´.

Transaction-oriented Disc File Management (TDFM)

An extended disc file management system is available under MAS. It is described in the P800 Programmer´s Guide 3, Vol. III: Software Processors, Part 7.

LOGICAL DISC (DAD)

The disc is divided into 1 or more sets of consecutive cylinders, each called a logical disc or DAD. A DAD can be added or deleted by the LIB Standard Processor, using the DCD or DLD commands.

Each user machine may use the disc space occupied by the DADs assigned to it. The System machine uses the System DAD´s, which are automatically assigned to it by IPL.

A VTOC (Volume Table of Contents) at the front of the disc describes its DADs. Each DAD has its sectors organised in a particular (not necessarily identical) manner, as defined by two values, specified by the user when the DAD is created:
- The number of sectors per granule.
- The interlace number.

These values are specified:
- As operator answers to the following console messages output by Premark:
  ```
  # OF SEC./GRAN. OF XXXXXX:
  # OF INT. OF XXXXXX:
  ```
  XXXXXX being a just defined DAD name.
- As values specified by the background machine user on the DCD (Declare DAD) command to LIB for the parameters:
  ```
  NSPG=
  NINT=
  ```

Before an explanation can be given of what these parameters mean, and what factors the user should bear in mind when choosing values for them, some further concepts need to be introduced.

Userids

Each DAD is divided into one or more user areas called userids. The purpose of this is to allow each background machine user a separate disc area. For foreground and system machines this is not relevant, since each has only one user and he uses the disc area of the first (or only) Userid in the DADs assigned to his machine. A catalogue at the beginning of each DAD describes its location.

Files

Each Userid area contains zero or more files of user records. A Directory at the beginning of each Userid describes the location of each file belonging to this user.

Granule

A granule is the unit of disc space allocation for all files in one DAD, as an integral number of sectors. For the first DAD on the disc, the minimum is 8 sectors/granule; for the other DADs, the minimum is 6. Each DAD has a granule size. The maximum sector length for a DAD is 512 bytes (except for some system DADs). The granule sizes can be different for different DADs.

Each set of cylinders (DAD) consists of a set of granules. A granule may occupy several tracks, and does not need to start or end on a track boundary.

The granules of one DAD may be:
- The DAD system granule. This uses 6 sectors of information about the DAD for MAS (including 4 catalogue sectors for this DAD). For the first DAD on the disc, it also contains the IPL, defective track and VTOC and volume label sectors.
- A Directory granule for a Userid in this DAD. Each Userid has one Directory granule, describing the locations of all the files for one user.
- A granule allocated to a file. Each file consists of one or more granules. A granule cannot be allocated to more than one file.
- Unused granules. These are available for use by future files or for extension of existing files (if their granule organisation allows this) or for directories of future Userids.

A BITTAB in the DAD system granule indicates, for each granule in the DAD, whether it is used or unused.

Granule Organisation

When creating a new disc file, the user specifies whether the granules to be allocated to it must be consecutive or not.

If the granules are to be consecutive, the user must specify how many are to be permanently reserved for the file before writing any record to the file

If the granules are to be non-consecutive, the system will allocate a specified number of granules (default 1) to the file; as soon as the user program has filled this granule with records, another granule will be dynamically allocated, and so on. When allocating a granule the system will choose any unused granule in the DAD. A table at the front of the file is maintained automatically which describes the DAD physical granules allocated to the file: logical granules 0, 1, 2 ... This table allows up to 200 granules to be dynamically allocated to the file, and is called the GRANYB sector.
Some notes on how to decide whether to use consecutive or non-consecutive granules are given later in this chapter.

Record Organisation

The user has two methods available. The I/O drivers for the disc always read or write one sector at a time, except for floppy discs and CDC CMD discs.

Sequential

If the user wishes MAS to calculate the sector number(s) containing the user record to be read or written and the user will always read and write the records sequentially, he should use the sequential record organisation.

- Each record will have a header describing its length, and the last record will be followed by an end of file (:EOF) marker.
- When the user issues a Standard Write LKM 1, the first free characters of the current sector are used to contain the record. If the sector becomes full, it is output and MAS continues if necessary writing the next characters of the record in the next sector. If the end of a granule is reached, MAS continues writing the record in the first sector of the next granule (which will be dynamically obtained if the file has non-consecutive granules).
- When a Standard Read LKM 1 is given, MAS reads as many sectors as required to extract the record into the user record area.

Direct Access

If the user wishes to access a file in which the records are not neccessarily in sequential order, he must use Direct record access. In this case, the required file relative sector number must be specified when using the Direct Read LKM 1.

If a file is to be written by Direct Write LKM 1, the required number of granules must be specified when the filecode is assigned, as no dynamic allocation of non-consecutive granules will be made by MAS.

Sectors per Granule

The number of sectors per granule can be specified to optimise disc efficiency.

For the first DAD on the disc it must be at least 8, since MAS uses sector 7 in the first granule of the first DAD for the VTOC. For every other DAD it must be at least 6, since MAS assumes sector 5 can be used for the catalogue. It must be no less than the number which would cause any file in this DAD to have more than 200 non-consecutive granules. For example, a file with non-consecutive granules,
which will require 3200 sectors, necessitates at least 16 sectors per granule. The larger the granule size, the larger will be the user directories, and so the greater the maximum number of files which one user can create in this DAD. The larger the granule size, the bigger the amount of disc space allocated every time a file with non-consecutive granules requires another granule, and therefore, the more disc space will be wasted by files which do not completely fill their last granule.

If a file with non-consecutive granules is read sequentially, then the larger the number of sectors per granule, the fewer the disc access arm movements required to retrieve the sectors. This is of course assuming that the disc arm is not repositioned by an I/O operation to another disc file, during the sequential read of the sectors for this file.

For Direct Access files with non-consecutive granules, normally one sector will be identical to one user record. If, however, the user has designed a complex file where one user record is contained in several logically consecutive sectors, so that every user record will require one LKM 1 per sector to access it, then to minimise the retrieval time for one record the granule size should be as many sectors as is necessary to contain the record length (not less than 8 for the first DAD and 6 for subsequent DADs).

For example, on the X1215/16 the MAS Swapping Dad (D:CI) has 11 sectors per granule, so that each granule contains one 2K core image page. The 11 sectors can be retrieved in a loop which requires at most one access arm movement.

DISC FILES

Users may create new disc files in the following ways:

- By assigning a filecode to a disc temporary file (a set of consecutive or non-consecutive granules in a DAD), using this filecode to output records via LKM 1 in a user program and then giving an LKM 40 (Keep File), or a KPF (Keep File) command, when the file has been created.
- By the LIB commands SVU, CSF, CDF and LTO.
- By the UPD commands !!OU or !!KF.
- By giving KPF or KOM commands to LIB for filecodes assigned, by Standard Processors or user programs earlier in the same :JOB to disc temporary files, for background users and earlier in the session for foreground users.

Note that users may replace a catalogued disc file:

- By specifying the catalogued file on the CSF or CDF commands to LIB.
- By specifying the catalogued file on the !!OU command to UPD.

Catalogued disc files are identified by:

DAD FILECODE	(File code /F0-/FF).
USERID	(Up to 8 ASCII characters);
FILENAME	(Up to 6 ASCII characters).
TYPE	(A 2-character code).
VERSION	(A digit from 0 to the value specified on the SMV command to LIB for this Userid, with a maximum of 7 and a default 0).

The attributes of a file are described by 4 flags:

- System flag
- Invisible flag
- Write Protect flag
- Shared flag.

These are maintained by the LIB Standard Processor, and an explanation of the meaning of these flags is given in the sections for the LIB commands SSH, RSH etc. LKMs 32 and 40 also set these flags.

The format of a file (record length, record organisation etc.) is not described in the Directory, but in the records themselves.

File Granules

Granule Allocation

Files are always allocated on disc in granules and the user specifies, whether these are to be consecutive or not. A granule cannot be allocated to more than one file. If a granule is erroneously allocated twice, the system halts on dectection with a fatal error (/1B). Unused sectors in allocated granules are not usable by other files.

The granules of a DAD are described in the Bittab (sector 0) of the DAD. Each bit in the Bittab describes a granule, set to one means the granule is not used, set to zero means that the granule is allocated to a file..

For non-consecutive files, the largest number of granules, that can be allocated to a file is: (DAD sectorlength-10)/2. So for the most used sector length of 410 bytes, the maximum number of granules in a file is 200.

For consecutive files, the largest number of granules that can be allocated to a file is dependent of the length of the DAD where it resides. The maximum length of a DAD is 32K sectors. So the maximum length of a consecutive file is: number of granules in the DAD minus 2 (1 granule for DAD information and one granule for userid information).

The first two sectors of each file are not available for the user. So, when the user addresses sector 0 of the file he gets the 3rd sector of the first granule.

File Header Sector

This is always logical sector 0 of file logical granule 0, and is unused at present by MAS.

GRANTB Sector

This is always logical sector 1 of file logical granule 0. It only contains information for files with non-consecutive granules. Otherwise it is an empty sector. The lay-out for a GRANTB in a DAD with a 410 bytes sectorlength is:

WORD	CONTENTS
0	cylinder number (only used for X1215/X1216).
1	length used (in characters) excluding words 0-1.
2-201	DAD physical granule number of file logical granule 0-199. A word containing zero means that there is no granule address in this word of the GRANTB. For OB files, file logical sectors 0-n and m-1599 are used; the file logical granules between n and m are unused, and will have zero entries in the GRANTB sector.
202-204	0

Record Sectors

A record is a string of data which will be input to, or output by, a program with one (normally) LKM 1 I/O Monitor Request.

Records of a file are contained in logical sectors 2 onwards of file logical granule 0, and the remaining sectors in any other granules allocated to the file. The granules of a file may be consecutive or non-consecutive, specified by the user when he creates the file by assigning a filecode to a disc temporary file. The four methods of doing this are by FCL ASG command, by BCL ASG command, and by LKM 23 or 33; whichever method is used, when assigning a filecode to a disc temporary file there is a parameter to specify whether the granules are to be consecutive or not.
Note that if a file is to be created by Direct Write LKM 1, the number of granules required must be specified when the filecode is assigned, and no dynamic allocation of non-consecutive granules will be made by MAS.

Choosing the Granule Organisation Method

- with consecutive granules:

  All granules must be allocated when the file is defined by assigning a filecode to a disc temporary file. The user must know how many granules are required. If too many granules are requested, unused granules are not freed when the EOF (End of File) mark is written by the user. If too few granules are requested, MAS will not allow extra granules to be dynamically allocated, but will return EOM (End of Media) status when the user tries to write beyond the last granule allocated.

  If the DAD has sufficient granules to satisfy the requested allocation number, but they are not consecutive, the assignment is rejected.

  Reading the granules sequentially will involve little disc head movement (if no other task is sharing the disc) and therefore be faster.

- with non-consecutive granules:

  The file may contain as many granules as can be accomodated in the GRANTB sector.

  The user does not need to know how many granules to request (unless the file is to be created by Direct Write LKM 1).

  When a filecode is assigned to a disc temporary file and non-consecutive granules are specified, MAS will allocate at least one granule and initialise logical sector 0 as the file header and logical sector 1 as GRANTB. Subsequent logical sectors are filled by the user via LKM 1. Whenever a granule is full, another granule is dynamically allocated (using the BITTAB sector of the DAD, which is kept in memory) and the GRANTB sector (on disc) is updated. If a Keep File (LKM 40) is given, the BITTAB is updated on disc.

  An assignment is rejected, when the number of required granules is not available on the DAD. However, it is also possible that a subsequent LKM 1 output operation could fail because a new granule cannot be dynamically allocated.

  Loading the file, and subsequently reading the granules in the order they were created, may involve considerable disc access arm movement, especially if the granules are split between cylinders.

Disc File Access

Disc files can be read by assigning a filecode to them (i.e. Assign a Filecode to a Catalogued Disc File) and then using LKM 1. They may be deleted by LKM 41, by the LIB Standard Processor or by the SCL DLF Command. The recommended procedure to create a new version of a catalogued file is as follows:

Assign filecode _A_ to a disc catalogued file
Assign filecode _B_ to a disc temporary file
Read filecode _A_ with LKM 1
Modify records as necessary
Write filecode _B_ with LKM 1 (catalogue it with LKM 40 or KPF).

If the filename and filetype specified in the last Keep File already exist in the Directory for the DAD and Userid, then the file whose version number equals the value specified on the SMV command to LIB for this Userid is deleted and its granules are freed in the DAD BITTAB. Other versions have their version numbers incremented by 1, and the kept file becomes version 0.

To replace a version of a catalogued file, the following methods are available:
- Assign a filecode to it and use LKM 1 Output operations to this filecode.
- Specify the catalogued file on !!OU or !!KF command to UPD.
- Specify the catalogued file on the CSF or CDF command to LIB.

LKM 1 is thus used for all input and output, A7 containing the type of I/O to be performed. Depending on the type in A7, MAS starts one of two Access Methods. MAS does not pass user disc I/O requests direct to the disc drivers. It is the Access Methods which communicate with the I/O drivers.

The two Access Methods are:
- Direct Access
- Sequential Access.

Direct Access

The Direct Access method only deals with sectors. It can read a sector or write a sector. In either case the user must specify (in the ECB pointed at by A8 when the LKM 1 is given) the file relative sector number. For Read, the numbers are independent of each other. A program can thus request to Read the 74th file relative sector, and then request to Read the 5th file relative sector. For Write, the user can write sectors in a non-sequential order, providing the filecode defines:
- either a disc catalogued file
- or a disc temporary file, and the number of granules was reserved when the filecode was assigned.

Note that if the file has non-consecutive granules, a new granule is not dynamically allocated if a Direct Write is given to a sector beyond the last granule.

The Direct Access method does not examine or convert sectors, but merely passes them between the disc and the user program. Direct access sectors thus have the format:

WORD	CONTENTS
0	cylinder number (only for X1215/X1216, for other discs not used).
1-end	user data

It is the user`s responsibility to identify what is a record and which sector(s) contain it.

For consecutive granules, to read a specified sector, the Direct Access method converts the file relative sector number to a disc physical sector by extracting:
- The disc physical sector number of the file header sector for the file, from the Directory.
- The interlace number for the DAD, from the DAD BITTAB sector.

For non-consecutive granules, to read a specified file relative sector, it is converted to a file logical granule number. This granule`s DAD physical granule number is found from the GRANTB sector for the file, and the disc physical sector can be calculated from it. For example, the user requests to Read the 10th record sector on the file (file relative sector number 9). Suppose the VTOC for the disc says that the relevant DAD has 8 sectors per granule. The Directory for the relevant Userid is then located from the DAD catalogue pointed to by the VTOC. The file is found in the Directory and its GRANTB sector is read. The address of the second file logical granule is found from the GRANTB. The required sector is the fourth logical sector in this granule.

Sequential access

The Sequential Access method is used if the user:
- will always want to read records in the same order they were created, starting with the first record. -Requires the system to identify the sector(s) containing the next record. For a Read, the system will extract the next user record from the sector in the blocking buffer (and, if the next user record is not entirely in the current sector, it will read as many subsequent sectors as required) and place it in a user record area. For a write, the system will take the next record from a user record area and format it into as many sectors or parts of sectors as necessary.

The sequential access method requires every user record to be written to contain a header field which describes the length of the record. Other header and trailer information will be added by the Sequential Access method itself, and stored on the disc.

If consecutive granules are used, they are read and written sequentially, sector by sector.

If non-consecutive granules are used, they are allocated and written dynamically. They are read in the order they were written by using the GRANTB sector.

The recording areas of sectors contain:
- Records. A record can be contained in several sectors, or a sector may contain several records. A record may even be contained in more than 1 granule. The maximum record length is 4095 characters, since this is the size of the buffer.
- An EOS entry. This is output by ASM at the end of each object module (since object modules must always start in a new sector). It causes the Sequential access method to write the next user record at the start of the next sector, even if the current sector is not full. The facility is also available for users.
- An EOF entry. Written at the end of the file by LKM 1. It is always written at the start of a new sector.

The sector header (in word 1 of each sector) contains:

BITS	CONTENTS
0	1 if the sector has been deleted
1	1 if the sector contains an EOS
2	1 if the sector contains an EOF
3-15	the length in characters used in this sector (excluding the 4 characters of word 0 and word 1).

All entries in the recording area (i.e. Records, EOS and EOF) have the format:

WORD	CONTENTS
0	bit 0 is not used. Bit 1 is 1 if the record is an EOS. Bit 2 is 1 if the record is an EOF. Bits 3-15 contain the entry length in characters, excluding this word and the next one.
1	original record length, in characters. (The system will have removed trailing blanks from the user record, and added 1 character if it then had an odd number of characters.)
2 - (n-2)	record data.
(n-1)	file relative sector number of the sector containing the first word of the record (usually designated `S').
n	displacement of the first character of the record, within the sector defined by the previous word. It is 4 if the record is at the start of the sector recording area (usually designated `D').

EOS entry is /4004 followed by /0000 S D.
EOF entry is /2004 followed by /0000 S D.

A blank card would be loaded to disc as 4 words containing:

4
80
S
D

For a detailed information about the structure of DFM files, see volume V : Trouble Shooting Guide.

INTRODUCTION

This Chapter contains a brief description of the operation of the machine from the initial load to the point at which the monitor is activated. Procedures for running diagnostic routines, and de-bugging programs, are dealt with in the P800 Trouble Shooting Guide. The Stand-Alone Dump, however, is also described here.

THE CONTROL PANEL

The Control Panel of the P857/P858 displays the contents and addresses of locations up to 128K of main memory. While the computer is running the instructions and their addresses are continuously displayed, so that when a machine HALT occurs the address and value of the next instruction to be executed can be read off.

The Control panel of the P859/P854/P876 consists of two displays, When the machine is running, one display contains 'run' and the second contains an eventually set preset address. When the machine halts, one display contains the address of the next instruction to be executed and the other display contains that instruction itself.
Operator push buttons are provided for normal manual operations, including load and read facilities, for both registers and main memory. In addition, a bootstrap, held within a ROM located on the CPU board, is automatically loaded into memory when the IPL button is pressed, and allows the loading of any initial program load routine.

LOADING THE BOOTSTRAP

The bootstrap is a basic program used to load more sophisticated loader programs, such as the IPL (Initial Program Loader). In the P800 systems,the bootstrap is automatically loaded and started, when the IPL button is pressed.

When the control panel IPL button is pressed, a bootstrap is copied from a 64-(P857) or 256-(other machines) word ROM into memory, and this loads the IPL from the device and channel specified by the settings of the data switches (P857/P858) or by the pushings of the applicable buttons (P859/P854/P876).

For the various devices, the contents of the IPL data should be as follows:

Magnetic Tape:	0000 0010 10xx xxxx	(e.g. /0284)	
Cassette Tape:	0000 0111 10xx xxxx	(e.g. /0785)	IOP connected.
X1215/X1216 :	0110 yyyy 11xx xxxx	(e.g. /65F2)	
CDC-SMD BIGD :	0000 0001 00xx xxxx	(e.g. /0116)	
CDC-SMD BIGD2:	0010 0000 00xx xxxx	(e.g. /2016)	
DDC FHD :	0100 0000 00xx xxxx	(e.g. /400F)	
CDC-CMD :	0010 0000 01xx xxxx	(e.g. /2056)	fixed part.
CDC CMD :	0010 0000 00xx xxxx	(e.g. /2016)	cardridge.
Flex 0.25M :	0100 0000 10xx xxxx	(e.g. /4083)	IOP connected.
Flex 1M :	0110 0000 00xx xxxx	(e.g. /6083)	

Where xxxxxx is the Device Address and yyyy is the interlace factor (X1215/X1216).

The operation of the automatic loading facility consists of 4 main steps:

Step 1	The bootstrap is copied from the ROM into the first words of memory
Step 2	The value set on the data switches or input via the buttons is copied into register A15
Step 3	The CPU is put into INHIBIT INTERRUPT state
Step 4	The P register is loaded with zero and the CPU is started.

RUNNING THE IPL

With IPL, from disc a program is loaded, which determines what monitor is wanted and where it is to be loaded.
The program outputs the following questions:

MONITOR? reply the name of a load module containing a monitor, residing in the first userid of the first DAD of the disc, or a question mark (?). A question mark results in a print out of all load modules residing in the first userid of the first DAD.

LOAD ADDRESS? reply the address where the monitor is to be loaded. Default address (so <CR>) is /0000.

With the IPL program, it is impossible to load a MAS 8 monitor with Extended Mode. Therefore, first a program has to be run to load the MAS 8 monitor. The name of that program is LDMASR , so for an Extended Mode monitor the first question of the IPL program should be answered with: LDMASR.
The LDMASR program on its turn asks for the name of the MAS 8 monitor to be loaded by asking the MONITOR? question again. The reply must be the name of a load module containing an Extended Mode monitor. The default (so <CR>) is: MASR

STARTING THE SYSTEM

When the IPL program has finished loading MAS, it branches to an initialisation routine, which performs a number of checks:

- it checks the filecode of the disc from which the IPL was made. If this disc was not declared with filecode /C0 at system generation time, but with filecode /Cx, it exchanges the filecodes /Cx and /C0.
- it discovers the total memory size of the bare machine, and places this information in one of the MAS tables.
- it completes the setting-up of the system machine tables
- it asks for the DATE (reply: yy,mm,dd) and the TIME (reply hh,mm[,ss] or <CR>). This is only done in Extended Mode systems..
- finally, it simulates the operator command:

  SM SYSTEM

  and exits. The system machine is started and the system manager may now enter SCL commands to define and start the foreground machines to be used in this session, plus the background machine if required.

CHANGING A DISC PACK

A removable disc pack may, unless it is on the drive from which the IPL was done - which is automatically assigned to system machine filecode /C0 - be removed at any time during a session, and replaced by another removable disc pack.

When this happens, MAS checks the volume number of the mounted pack against the one of the dismounted pack. If equal, MAS assumes that the same disc is mounted again and no action is taken. When the volume numbers differ, the following actions are performed:
- The device tables are updated.
- All filecodes assigned to the dismounted disc are deleted.
- The volume label of the mounted disc is printed on filecode /EF of the system machine.

Although a disc can be changed at any time, care should be taken, with dismounting. It can abort programs, which accessed the dismounted pack.

There is no need for the operator to inform MAS (by an operator command or by an SCL command) that a disc pack is to be changed. The system will know this automatically as soon as the drive becomes ready.
All messages are output to the filecode /EF of the system machine. If an error occurs, one of the following messages is output:

```
    DISC UNIT xx: VOLAB READ ERROR.
```
The Volume label could not be read.
```
    DISC UNIT xx: VTOC READ ERROR.
```
The Volume Table Of Contents could not be read.
```
    DISC UNIT xx: BAD TRACK READ ERROR.
```
The Bad Track sector of the mounted disc could not be read.
```
    DISK UNIT xx: AN EXTENDED FILE USES THE OLD DISK.
```
On the dismounted disc, an Extended File was assigned. As this file was not yet closed, still tables belonging to that file reside in core. These tables have to be written onto the dismounted disc, because else the Extended File will be inconsistent. So remount the dismounted disc, close the Extended File and then change discs.
```
    DISK UNIT xx: INVALID DISC TYPE.
```
In the Volume label, a Disc Type has been recorded, which is not known to the system. Legal Disc Types are:

1215	X1215 2.5M disc
1216	X1216 5M disc
40M	CDC SMD 40M disc
80M	CDC SMD 80M disc
FHD	DDC Fixed Head disc
6875	X1215 disc created by FTS
6876	X1216 disc created by FTS
6877	CDC SMD 80M disc created by FTS
X1	X1215 disc created by DOS P800
X2	X1216 disc created by DOS P800
16M4	CDC CMD 16M disc removable part
16M2	CDC CMD 16M disc fixed part
48M2	CDC CMD 48M disc fixed part
80M2	CDC CMD 80M disc fixed part
8M	BASF 6171 8M disc
24M	BASF 6172 24M disc
FLD2	Floppy disc type F3
FLD1	Floppy disc type F1

If the disc has not been premarked, the following error message is output:
```
    DISC UNIT xx: VOLUME SERIAL NUMBER ERROR.
```

In all messages, xx means the device address of the disc.

If the message giving the volume label is not printed, it means there was not enough space in the System Dynamic Area to allocate memory to read the volume label. MAS has not in this case updated its tables, and the operator should re-ready the drive to ensure that this is done. Not printing the volume label can also mean that a pack with the same volume number was mounted.

Warning

If the installation has two replacable discs with the same volume serial number, and one is dismounted and the other is mounted in its place, MAS does not re-initialise its tables. Using the disc may then cause considerable damage to the data on the second disc, especially if the two discs have different DAD, Userid or file layouts. It is therefore strongly recommended that volume serial numbers should be unique within an installation.The same can occur, dismounting a disc, changing its lay-out on another installation and mounting it again on the same unit, while no other disc was mounted on that unit. Also in this case the disc can be damaged.

STAND ALONE DUMP

The Stand-Alone Dump is incorporated into MAS, but uses none of its facilities; thus it may be used as a diagnostic tool when MAS itself has aborted with gross errors. It has two entry points, one to dump the entire memory space plus all registers (including the MMU registers), the other to dump selected portions of memory only, plus all registers.

To run the dump program, proceed as follows:

1) Press the INST button to stop the machine.
2) Read the contents of registers A0 (P register) and note it down for later use in debugging the dump output.
3) For a full dump, load A0 with /12F0 and press RUN. The dump is now output to the line printer.
4) For a partial dump, load A0 with /12FC; then press RUN. The system will halt again and now put in A1 and A2 the start address of the area to be dumped and in A3 and A4 the end address and press RUN again. Now the partial dump is output on the lineprinter. After any partial or full dump, another dump can be made.

Notes:

a) If the line printer is off-line or faulty, the dump routine loops. Rectify the fault, put the printer on-line and the dump proceeds.
b) When entering addresses in Step 4, the least significant bits of A1 (or A3) contain the most significant bits of the absolute address; A2 (or A4) contains always the 16 least significant bits of the address.

METHOD OF ENTRY

Operator commands are given on filecode /EF of the System machine. This filecode should be assigned to an interactive device (such as a console), since it is also used by MAS to send messages to the operator (such as to reload paper into the printer).

LMK 25 (Read Operator Key-In) may be issued by user programs if it is required to receive messages from the operator.

Operator commands are entered by the following procedure:

1. Press the control panel interrupt button.
2. Enter the command on filecode /EF after the 'M:' which is output immediatily on that filecode.

The left arrow or the Backspace (BS) key is used to delete the last input character currently in the buffer. It may be pressed several times consecutively. If pressed _n_ times, a logical backspace of _n_ characters is performed. If there are no input characters currently in the buffer, it is ignored.

The CAN key or the CNTL<D> is used to delete all input characters currently in the buffer. Also all characters are ignored until a termination character <CR> is detected.

OPERATOR COMMAND SYNTAX

Each operator command must obey the following syntax rules:

- It must be terminated by a CR (carriage return);
- It must be no more than 1 physical line (for a keyboard, for example, a maximum of 74 keys may be pressed consecutively, excluding Backspace and Can keys and the characters which these remove from the input buffer).

Each command has the format:
- A two character mnemonic code.
- One blank.
- Zero or more positional parameters.

Each positional parameter consists of:
- Zero or more non-blank ASCII characters (zero is only allowed for optional positional parameters). Hexadecimal values, when required, are not preceded by a / symbol.
- A comma, if it is not the last positional parameter for this operator command. Otherwise the CR key.

COMMAND DESCRIPTION

All operator commands are actioned by routines loaded into the transient area of the system machine. Therefore, other services which require the transient area cannot be performed simultaneously. So care must be taken that one operator command does not give rise to an operator message (such as the PU message). If this happens, the operator command is ignored. An example of this is the case where a `DM´ command is given, but the device attached to filecode /02 of the system machine (the device on which the dump would appear) has not been made ready. The usual `PU´ message requesting the operator to ready the device cannot be output, because the transient area is already in use.

If the operator command is rejected, an error message will be printed on the device assigned to the system machine filecode /EF.

FORMAT 1

AB Abort the background program.

The program in the background machine is aborted (unless it is the BCP).

If the background program issued an LKM 7 (Keep Control on Abort), control is given to the user abort label specified, and an abort code of 06 isplaced in the Abort Control Block. It is then the user's responsibility whether to abort or not by issuing LKM 3 with an exit code and postmortem dump flag in A7.

Otherwise, the postmortem dump flag is set on by MAS, and a dump will be performed according to the DUMP parameter on the BCL RUN or Processor Call command which activated this program.

(Note that for systems without Extended mode postmortem dumps are a SYSGEN option, and if this option is not selected, a postmortem dump is never performed.)

If there was a previous BCL :STP command in this JOB and the ABCD parameter was specified on it, MAS will set the severity code to the specified value. Otherwise the severity code is set to /7F.

The BCP is then reloaded, and processes the next BCL command from the filecode /E0 (unless a catalogued procedure is in use) of the background machine. It will be processed or flushed according to whether it is a step terminator (:STP, :JOB :EOJ, :EOB); if not, whether the severity code exceeds the :STP value allowed.

One of the following error messages will be output if the AB command for the background machine is rejected:

```
MACHINE UNKNOWN (no Batch machine declared)
PROGR INACTIVE (no SB operator command given or BCL :EOB has been read).
BCP PROCESSOR! (the BCP processor was active)
PROG ALRDY ABORTED
```

FORMAT 2

AB machineid, programid   Abort a foreground program.

machineid	a foreground machine name on a previous SCL DCF command (i.e. SYSTEM or BATCH not allowed).
programid	a program name on a previous FCL command (LOD, REP, RON or SWP) for the same foreground machine, or a Middleground program.

The foreground program is aborted. A message is sent to filecode /01 of that foreground machine, to allow the user to request a dump for core resident programs or to print out the registers. The program is placed in an abort state, which means:

- The current task can never exit. (For a re-entrant program, all tasks are placed in the abort state.)

- The tasks in its activation queues will never be started.

- No new activation queue entries can be created by FCL CNT commands, or by LKM 10 (connect a program to a timer) or 12 (activate).

- New activations by previous LKM 10 will not be made.

- LKM 12 will return status code -6.

- Disc resident programs are swapped out.

The user may reactivate the program by an FCL RUN or FCL ACT, but not by an FCL CNT command.

An FCL ACT command resets the abort flag in the PCT for the program. Future FCL CNT or LKM activations will then be accepted. The FCL RAB command can also be used to reset the abort flag in a PCT.

One of the following error messages will be output if an AB operator command for a foreground program is rejected:

```
PROG NAME MISSING (only one parameter was given)
WHAT IS THE 3RD PARAMETER (after the 2nd parameter a comma was detected)
MACHINE UNKNOWN
PROGR UNKNOWN
PROGR INACTIVE
SYSTEM PROG!
PROG ALREADY ABORTED
```

Note: The abort of a program in the wait state can only be completed when the event waited for has occurred.

FORMAT

```
AS fc,dndd
```

fc	a filecode of one or two hexadecimal digits, without a preceding / character.
dn	a non-disc device name of two characters; see Chapter 3.
dd	device address; two hexadecimal digits.

The AS operator command is used to assign a system machine filecode to a non-disc device. When the device name is 'NO', the dd parameter must be omitted and the filecode will be assigned to a dummy device.

One of the following error messages will be output on system machine filecode /EF if an AS operator command is rejected:

PARAM TOO LONG
The fc parameter contains more than 2 or the dndd parameter more than 4 characters.

INVALID FILECODE
The fc parameter contains non-hexadecimal characters.

INVALID DEVICE ADDRESS
The dd parameter contains a device address that has not been generated in the system.

SYST. DYN. AREA OVERFLOW
There is no place in the System Dynamic Area to create the File Code Table.

DEVICE UNKNOWN
The dn parameter contains a device name that does not exist or has not been generated in the system.

FORMAT

```
CR fc
```

fc a system machine filecode. It must already be assigned (by IPL or by an SCL ASG command), and the device must be ready. If not, nothing will happen.

The memory will be updated from a set of sequential records (read from fc). Each record except the last has the syntax:

```
a,v0[,v1]...
```

where a, v0, v1, etc. have the same meanings as for the WM operator command. The last record is an EOF record (the format varies according to the device) for the sequential access method. The transient area is blocked until this is read.

One of the following error messages will be output if a CR operator command is rejected:

```
I/O ERROR
F.C. ERROR
```

One of the error messages shown for WM will be output if a record is rejected.

Dump the Background Machine

FORMAT

```
DB [from[,to]]
```

from a relative address (up to 4 characters) defining the first word of the back-ground machine to be dumped. Default is the first word of the first page allocated to the background machine (relative address /0000).

to a relative address, greater than "from" and up to 4 characters, defining the last word of the background machine to be dumped. Is ignored if "from" was omitted (so, when the comma is given). The default is calculated so that only one line of dump will be printed, but if "from" is also omitted, the entire background machine will be dumped.

Note:

For a disc-resident background machine, the maximum permitted relative address varies during the session, as the pages are dynamically allocated from the common dynamic loading area whenever a program starts. The maximum possible allocation is 16 pages. For a memory-resident background machine, in addition to the rules above, both "from" and "to" must not exceed the maximum relative address implied by the first parameter of the SCL DCB command.
The specified (or defaulted) memory locations will be dumped to filecode /02 of the batch machine. DB does not block the system machine transient area, since it uses the core resident dump routine.

The output generated by the DB command starts with:

```
    DB FROM xxxx TO yyyy
```

followed by the dump.
One of the following error messages will be output if a DB operator command is rejected:

```
    MACHINE UNKNOWN
```

The Batch machine was not declared.

```
    INV. ADDR
```

The second parameter was not greater than the first one.

```
    ADDR. FORBIDDEN
```

Both or one of the specified addresses were outside the area of the Batch machine.

```
    DYNAMIC AREA OVFL
```

There was no place in the System Dynamic Area to allocate a buffer for generating a dump line.

```
    PARAM ERROR
```

The first or second parameter did not contain up to 4 hexadecimal characters.

FORMAT

DM a1,a2

a1	a number up to 5 hexadecimal digits 0-FFFFF giving the absolute address of the first byte to be dumped.
a2	a number up to 5 hexadecimal digits, not less than a1 and not more than FFFFF, giving the absolute address of the last byte to be dumped.

The DM operator command can be used when the system manager wants to give an SCL DUM command but unfortunately has already given the SCL BYE command, or it may be used when a foreground user wishes to give the FCL DUM command but has already given the FCL BYE command. Alternatively, it may be used to dump a memory area which is not wholly allocated to one machine.

To dump memory in the background machine the DB operator command should be used, as this does not use a transient routine.

The DM command is an emergency facility (performed by a transient routine with software level 3) which will dump the specified memory to filecode /02 of the System machine. The transient area is blocked while this is done.

If nothing happens, it means the following:

Filecode /02 of the system machine defines a device which requires operator intervention at this moment. The system does not bother about that, and sends the dump to a dummy device, until the dump has been output or the device has been readied.

One of the following error messages will be output if a DM operator command is rejected:

PARAM MISSING (not 2 parameters in the command)
PARAM ADDRESS TOO LONG (a1 or a2 have more than 5 hexadecimal digits)
ADDRESS VALUE NOT HEXA (a1 or a2 do not contain hexadecimal digits)

FORMAT 1 Background Machine

```
KI BATCH,s,m
```

FORMAT 2 Foreground Machines

```
KI u,p,s,m
```

u	a foreground machine name specified on an SCL DCF command.
p	a foreground program name. This should not be a re-entrant program.
s	two characters, identifying for which request the message applies as definied in the previously given LKM 25.
m	the message to be transmitted to the task's LKM 25 buffer. The message is terminated by the first blank, comma or CR, providing this occurs before the requested length is reached.

The KI operator command is used to enter a message to a task which has issued an LKM 25 request.

One of the following error messages will be output on system machine filecode /EF if a KI operator command is rejected:

PARAM MISSINGThe command does not contain the predefined number of parameters.

MAC UNKNOWN
The specified machine name does not exist or contains more than 6 characters.

PROG UNKNOWN
The specified program name does not exist in the machine or contains more than 6 characters.

SP CH UNKNOWN
Although the program is expected some KI command, the specified special characters do not belong to any LKM 25 given. The message can also mean that more than two special characters have been given..

MESS TOO LONG
The message given in the KI command contains more characters than expected in the program.

KEY IN NOT EXPECTED
The progam is not expecting any KI command.

Notes:

- Two special characters that need not to be equal have to be specified. Blank special characters are accepted, but in that case a blank is not a separator character, so that another blank has be be given as separator.
- When the mnemonic for an operator command is unknown, the system assumes the typed characters are special characters for a KI command. It searches all programs in all machines to check if there is a program expecting a KI command with special characters equal to the typed mnemonic. So defining special characters unequal to any operator command and unique in the system, the layout of the KI command can be:
  s m (s: special characters, m: message)

If no program has an outstanding KI for this special character, the error message: UNKNOWN COMMAND
Meaning, that the mnemonic is not equal to any operator command, nor is any program in any machine expecting a KI with the typed special characters.

FORMAT

```
OF da
```

da	two hexadecimal digits without a preceding / character, representing the 6-bit device address of a non-disc device.

OF is used to inform the system that a non-disc device may not be allocated to user or system programs.

If, for example, a program issues an LKM 23 type 0 (Assign a Filecode to a Physical Device) where there are several devices of the required device type available, but the program does not specify which device is to be assigned, then an inoperable device will not be chosen for the assignment. Similarly, a BCL REQ command or LKM 54 will not result in an OF device being allocated.

One of the following error messages will be output to system machine filecode /EF if an OF operator command is rejected:

```
PARAM MISSING
PARAM TOO LONG
INVALID DEVICE ADDRESS
UNKNOWN DEVICE ADDRESS
DISK DEVICE CANNOT BE PUT ON/OFF
```

Note:
'OF' for a device which is already 'OF' is ignored.

FORMAT

ON da

da	two hexadecimal digits without a preceding / character, representing the 6-bit device address of a non-disc device which has been specified on an OF operator command given previously in this session.

ON is used to inform the system that a non-disc device, previously declared unusable, may now be used by MAS.

One of the following error messages will be output to system machine filecode /EF if an ON operator command is rejected:

```
PARAM MISSING
PARAM TOO LONG
INVALID DEVICE ADDRESS
UNKNOWN DEVICE ADDRESS
DISK DEVICE CANNOT BE PUT ON/OFF
```

Note:

'ON' for a device which is already 'ON' is ignored.

FORMAT

```
PK cn[,{V | W}]
```

cn the disc file code of the physical disc drive containing the disc pack to be premarked (Cl-CF).

V the disc has already been premarked. Only the VTOC is to be reinitialised.

W all home addresses and identifiers are to be written, but are not to be checked.

If neither V nor W is supplied, all home addresses and identifiers are written and checked.

1) X1215/16/FHD

Premark performs the following operations:
- Writes and reads all sectors of all cylinders of the whole physical disc.On bad spot detection it declares the corresponding track bad and records that track in the bad track sector in the first granule of the first DAD (sector 6).
- Formats each track to contain 16 (for X1215/16) or 43 (for FHD) physical sectors of 205 words, and initialises the first word of each sector to be its logical cylinder number.
- Allocates cylinders to the first DAD on the physical disc, according to operator specifications. As well as specifying the number of cylinders to be assigned to the DAD, the operator specifies how the cylinders are to be formatted into granules (i.e. the number of sectors per granule, and the interlace factor).
- Initialises the first 2 granules of the physical disc to contain information about the disc pack and the first DAD, namely:
  - Granule 0:
    - BITTAB sector (sector 0);
    - IPL sector (1);
    - Catalogue sectors for the first DAD (2-5);
    - Bad Track sector (6);
    - VTOC sector (7).
  - Granule 1:
    - Directory sectors for the first DAD.

Premark is performed in the transient area of the system machine. Premark is started by the Operator Command PK entered on the system machine operator console.

Premark outputs a series of messages on the Operator console, to which the operator must reply. If a reply to a question is invalid, the question is repeated. The operator can terminate a PK operation by replying AB to any question. Each reply must be terminated by CR.

DK TYPE (1215, 1216, FHD1, 2, , ,8):	4 ASCII characters defining the disc type. For FHD 8 disc types, FHD1, FHD2, -- FHD8 can be specified depending on the disc capacity.
LABEL:	1 to 16 ASCII characters for the disc volume label (not beginning with ‘AB’).

PACK NBR: 1 to 4 hexadecimal characters without a preceding / for the disc pack number. This must be unique for each disc pack.

DAD NAME: 1 to 6 ASCII characters for the name of the first DAD (not beginning with `AB').

OF CYL OF XXXXXX :
A number from 1, specifying how many cylinders are to be reserved for the first DAD (whose name is XXXXXX). Cylinders will be allocated consecutively from cylinder 0 onwards.

OF INT OF XXXXXX :
An odd number from 3 to 15, defining the interlace factor of the sectors for the first DAD (whose name is XXXXXX).

OF SEC./ GRAN OF XXXXXX :
A number (minimum 8), specifying the number of sectors per granule for the first DAD (whose name is XXXXXX). It must not exceed the number of sectors per cylinder (32).

SYST. USERID:
1 to 8 ASCII characters, giving the name of the first Userid for the first DAD. A Directory for this Userid will be created.

PASSWORD: 0 to 4 ASCII characters. This is the password associated with the first User of the first DAD. It is used only by User Accounting routines.

ACCOUNT#: 0 to 4 decimal characters, representing a positive value from 0 to 9999. This is the account number associated with the first Userid of the first DAD. It is used only by User Accounting routines.

Premarks ends successfully with the following message:

```
# OF DEF. TRACKS: nnnn
```

Premark ends unsuccessfully with one or more of the following messages:

NR OF INT NOT COMPATIBLE WITH NR OF SECT/TR
The interlace factor and the number of sectors per track have a common divisor, not equal to 1.

NR OF INT NOT LESS THAN NR OF SECT/TRACK
The interlace factor is greater than 16 (for X1215/16) or greater than 43 (for FHD).

TOO MANY DEF. TRACKS: n
(n is a number over 6). The disc pack is unusable.

BAD TRACK IN FIRST CYL.
Cylinder 0 is defective. The disc pack is unusable, since the Defective Track Table sector <u>must</u> be written in Cylinder 0.

THRUPUT ERROR

SEEK ERROR

XPCTD: c_1 READ: c_2

where c_1 and c_2 are the co-ordinates of the expected sector and the sector actually read. The format is 12 hexadecimal digits: ccchhhhrrrr (cylinder No./head No./sector No.). This message indicates that an error was detected while checking identifiers.

For FHD two additional error messages can be output:

SECTOR NOT FOUND

LOCKED OUT SECTOR ADDRESS

After the PK Operator Command, there may occur one of the following messages:

FC MISSING	No filecode was specified on the PK command.
INVAD PARAM	The filecode is invalid.
INV. FILECODE	The filecode is not assigned to a physical device.
FILE CODE NOT ASGN	The filecode is not in the system machine filecode table (initialised at SYSGEN).
INV. DEVICE	The filecode is assigned, but not to a disc unit.
UNKNOWN DEVICE	The reply on the DK TYPE question is not one of the given possibilities.
DVCE NOT OP	The disc to be premarked is not operable. Make it operable and restart the premark process.

2) <u>CDC SMD DISC</u>

This is identical to that for the X1215, except that:

- The number of cylinders for the first DAD is not restricted to the disc capacity but to the Bittab length and the maximum number of sectors in a DAD (32767)
- The number of interlaces is not necessarily a number from 3 to 15.
- The number of sectors per track is not restricted to 16.
- The sector length is not restricted to 205 words.

Three extra questions are sent to the console when a Premark is made of a CDC-SMD disc:

PACK CERTIFIED? (Y OR N):

For CDC disc a utility exists to certify it. This utility, named <u>COPY</u> is described in Appendix E. The Copy utility searches bad tracks on a disc and builds a bad track sector. By answering this question with 'Y', premark takes the existing bad track sector into account.

DK TYPE (40, 80, 150, 300M):

Only 40M and 80M discs are known to Mas at present, so only 40M or 80M should be specified.

OF SEC./TRACK OF XXXXXX:

SEC. LENGTH (IN CHAR) OF XXXXXX:

In the last two questions, the operator should enter a positive non-zero decimal number of 1 to 2 or 4 characters.

If the sector length is too large or the number of sectors per track is too large, the following message is output:

```
SECT/TRACK OVERFLOW
```

If the DAD declared is too large, one of the following error messages is output:

```
DAD MUST NOT EXCEED 32768 SECTORS
DAD TOO BIG BITTAB OVERFLOW
```

If an error occurs that cannot be recovered by premark, the message:

```
                    CERTIFY THE PACK BEFORE PREMARK IT
```

is output.

CDC-CMD discs

The questioning for CDC-CMD discs is similar to that of the CDC-SMD disc. Only the question about the disc type differs:

```
                    DISK TYPE:
                    CDM : 16M4 (REMOVABLE PART),
                    16M2, 48M2, 80M2 (FIXED PARTS),
                    ANSWER:
```

The answer must be one of the mentioned disk types.

FLOPPY disc

The questioning of the floppy disc is similar the that of the X1215/16 discs. Again, the disc type question differs:

```
                    FLOPPY TYPE F1 OR F3
```

The answer must be F1 (DOS compatible floppy with one DAD) or F3 (floppy disc with the possibility to contain more than one DAD). Furthermore, for floppy discs the SECTORLENGTH question is asked and the INTERLACE question may be answered with 1.For the layout of floppy disc, see Appendix E.

Pause a User Program

FORMAT 1 Pause the background program.

    PS

The background machine (i.e. the program currently residing in it - the BCP, a Standard Processor, or a user program) will be placed in the pause state. The pause will be terminated by the RS operator command. The TIME parameter in the RUN command is not overrruled by this command.
When a swappable batch program is put in pause, the program is swapped out.If the background machine has not been defined the following error message is output:

    MACHINE UNKNOWN

If the current program in the background machine is already in a pause state (because the PS operator command has already been given, or the program has issued LKM 6, or a BCL PSE command was the last BCL command processed), the following error message is output:

    PROG ALRDY IN PAUSE

If the SB operator command has not been given, or if the BCL :EOB command was the last BCL command processed, the following error message is output:

    PROGR INACTIVE

FORMAT 2 Pause a foreground program.

    PS machineid,programid

machineid	a foreground machine identifier specified on an SCL DCF command (i.e. SYSTEM or BATCH not allowed).
programid	the name of a program running in the foreground machine.

The program is swapped out if it is disc resident.

A re-entrant foreground program can be put in pause but, since MAS is searching for the first program with the specified name, only the first activated task can be put in pause.

The paused program can be restarted by the RS operator command. One of the following error messages will be output if a PS foreground program operator command is rejected:

    PROG NAME MISSING (only one parameter was specified)
    WHAT IS THE 3RD PARAM. (more than two parameters were specified)
    SYSTEM PROGR! (the system machine was specified)
    MACHINE UNKNOWN
    PROGR INACTIVE
    PROG ALRDY IN PAUSE

Restart a User Program

FORMAT 1 Restart the background program.

```
RS [a7]
```

a7	the value to be contained in A7 on return to the program. The default is the value when the program was paused.

This command terminates the pause state of the background machine. It may be used, for example:
- After a PS operator command specifying the background machine.
- After an LKM 6 or LKM 54 issued by the current background program.
- After a BCL PSE, REQ or ROI command has been processed.
- After an SCL or FCL PSE command has been given for the background machine.

One of the following error messages will be output if the command is rejected (the first 3 messages apply only if a parameter is specified):

```
PARAM TOO LONG
BATCH MACHID UNKNOWN
BAD A7 PARAM VALUE (value to be put in A7 was not hexadecimal)
PRG NOT IN PAUSE
```

FORMAT 2 Restart a foreground program.

```
RS machineid,programid[,a7]
```

machineid	a foreground machine name on a previous SCL DCF command (i.e. SYSTEM or BATCH not allowed).
programid	a program name on a previous FCL command (LOD, RON or SWP) for the same foreground machine.
a7	the value to be contained in A7 on return to the program. The default is the value when the program was paused.

A Middleground program may also be specified.

This command terminates the pause state of a foreground program. It may be used, for example:
- After a PS operator command for the same foreground machine and program.
- After an LKM 6 issued by this foreground program.
- After a FCL PSE command for this foreground machine and program name.

One of the following error messages will be output if the command is rejected:

```
PARAM TOO LONG
MACHID UNKNOWN
PRG NAME UNKNOWN
BAD A7 PARAM VALUE
PRG NOT IN PAUSE
```

Note:
If only one parameter was specified, MAS assumes that the RS was given for the Batch machine with the parameter containing the a7 value.

FORMAT

    SB

This can only be entered after the SCL DEN command has been given, ending the definition of the background machine.

The BCP will be loaded from batch machine filecode /F0 into the background machine. If a memory-resident background machine was defined, the BCP is loaded into the pages reserved by the SCL DCB command. If a swappable background machine was defined, the BCP is loaded into pages allocated from the common dynamic loading area shared by all machines, and its core-image is copied to the swapping file (D:CI) defined by system machine filecode /F1. The BCP will start to read BCL commands from the device defined by Batch filecode /E0.

One of the following error messages will be output if a SB operator command is rejected:

    BATCH MACHINE UNKNOWN

No batch machine was declared

    MACHINE RUNNING

An SB command was already received by the system

    MACHINE IN GENERATION

The system received already a DCB, but not yet a DEN command

    /E0 ASGN TO NO DEVICE

The command input filecode is assigned to a dummy device

    /E0 NOT ASGN

No command input filecode was declared

    BATCH MACHINE /F0 NOT ASGN

The filecode, containing the BCP processor has not been declared

    SYSTEM DYNAMIC AREA OVERFLOW
    READ FILE I/O ERROR

Reading the BCP processor into memory, an I/O error occurred

    BCP PROCESSOR NOT CATALOGUED

No BCP processor resided on filecode /F0

    D:CI DAD F.C. /F1 NOT ASGN

Filecode /F1 was not assigned in the system machine (only for swappable background machine)

    D:CI DAD OVERFLOW

The D:CI DAD contained not enough sectors to contain the BCP processor core images (only for swappable background machine)

    BATCH MACHINE MEMORY OVERFLOW

The number of pages, specified in the DCB command for a core resident background machine were not enough to contain the BCP processor.

FORMAT

```
SC h,m[,s]
```

h	one or two digits from 0 to 23, representing the hour.
m	one or two digits from 0 to 59, representing the minute.
s	one or two digits from 0 to 59, representing the second. Default 00.

The SC operator command is used to initialise or reset the system clock. After a power failure, for example, the System manager may not have noticed it and thus will be unaware of the need to give an SCL CLK command.

One of the following error messages will be output to system machine filecode /EF if an SC operator command is rejected:

```
PARAM MISSING (less than two parameters given)
PARAM TOO LONG
SECONDS VALUE TOO BIG (>59)
MINUTES VALUE TOO BIG (>59)
HOURS VALUE TOO BIG (>23)
```

FORMAT

SD d,m,y

<u>d</u>	one or two digits representing the day. Minimum 1. Maximum 31 if m is 1, 3, 5, 7, 8, 10 or 12; 30 if m is 4, 6, 9 or 11; 29 if m is 2 and 1900+y is divisible by 4; 28 otherwise.
<u>m</u>	one or two digits 1 to 12, representing the month.
<u>y</u>	one or two digits 0 to 99, representing a year from 1900 to 1999.

The SD operator command is used to set the date field in memory.

One of the following error messages will be output to system machine filecode /EF if an SD operator command is rejected:

```
PARAM MISSING
PARAM TOO LONG
WHAT IS THAT YEAR?
WHAT IS THAT MONTH?
WHAT IS THAT DAY?
IT IS NOT A LEAP YEAR
```

FORMAT

SM machineid

machineid	a)	'SYSTEM', to start the SCL processor after it has been stopped with an SCL BYE command.
	b)	the machine name (1 to 6 ASCII characters) specified on a SCL DCF command, to start a foreground machine. The SM must not be given until the SCL DEN command, ending the definition of this foreground machine, has been processed.

The SM command causes the SCL/FCL program to be activated for the specified machine. It will start to read commands from the interactive device defined by the filecode /E0 for that machine. The message FCL: will be output to /E0 whenever it is ready for the next command.

One of the following error messages will be output if an SM operator command is rejected:

```
MACHID UNKNOWN
FCL RUNNING FOR THIS MACHINE (also output when SYSTEM machine is running)
/E0 ASGN. TO NO DEV.
/E0 NOT ASGN
MACHINE IN GENERATION
INPUT/OUTPUT F.C. 01 NOT ASGN
INPUT/OUTPUT F.C. 01 ASGN TO NO DEVICE
F.C. 01 NOT ASGN TO AN INPUT/OUTPUT DEVICE
```

Notes:
- When a foreground machine is started, the filecodes /02 and /01 have been defined by SCL commands. The operator should ensure that these devices are not being used by other tasks in the bare machine, or that, if they are being used, sharing them with this FCL task will not cause any problems.
- SM BATCH is allowed and has the same effect as the SB command.

FORMAT 1

SP dnda

dn	device name (CR, LP, PP or PL)
da	device address

The spooling process, input (when the device name is CR) or output (when another device name is specified), is started.

Input spooling

With input spooling, the system filecode /F2 is assigned to the DAD D:SPCR (input spool DAD). Input spooling consists of submitting jobs to the background machine. These jobs are recorded in the spool DAD and executed afterwards, one by one on FIFO (first in first out) basis. If no jobs are recorded in the spool DAD the message:

JOB QUEUE EMPTY

is output on filecode /EF of the system machine and the spooling process waits for new jobs to be submitted.
There are two kinds of input spooling: submit jobs to the background machine via the cardreader or via LKM 50. Both can be used in the same session.

When input spooling is done via the cardreader, on the SP CRda command jobs are read from the cardreader and copied into a file on the spool DAD. If no cards are present, the cardreader is in not operable state and outputs the message:

PU CRda, 0001, RY

As soon as cards are available, they can be copied from the cardreader to the spool DAD by giving the RY (retry device) command (RY CRda). The cards must contain jobs streams, so must start with a :JOB command and end with a :EOJ (or a :EOB) command. When all cards have been read, the cardreader goes into inoperable state and the jobs read in are executed, at least if the background machine has been started. If not, the jobs are executed, when an SB command is given. New job streams may always be delivered to the cardreader and will be read in as soon as a retry device command (RY) for it is given. If an :EOB (end of batch) card is read, new job streams can be read in by the cardreader, but they are only executed when a new SB command is given.

Submitting a job via LKM 50 can be done from any foreground machine. A file containing a job stream must be specified and will be executed in the background machine. For more information about this LKM see Appendix C of this manual. If only LKM 50 is used for input spooling, it is not necessary that the cardreader is physically present and the PU message can be ignored.

Output spooling

Output spooling applies on the lineprinter (SP LPda), the papertapereader (SP PPda) or the plotter (SP PLda). When spooling is started, a DAD filecode is assigned to a DAD in the system machine. For LP, filecode /F3 is assigned to the DAD D:SPLP, for PP filecode /F4 is assigned to the DAD D:SPPP and for PL filecode /F5 is assigned to the DAD D:SPPL. All writes to a filecode assigned to a spooled output device are translated by MAS into writes to a file on the applicable spool DAD. This is done by writing to an internal filecode, that is /100 more than the original filecode (internally in MAS filecodes of more than 8 bits can be used). A file on an output spool DAD is assigned, when a write is done to a filecode, for which not yet a filecode+/100 exists. The spoolfile is closed when, in foreground, an :EOF (end of file) is written to it or, in background an :EOJ or :EOB is received.

In foreground, the file can be closed in the following way:
- by using the CLS command (see system and foreground machine description).
- by writing an EOF to the spooled filecode via LKM 1 order /22 (see Appendix C).
- by writing an EOF via the Librarian command WEF.
- by reassigning the filecode that was used.
- by using the BYE command (from Mas release 8.50)
- by using LKM 79 (from Mas release 8.50)

As soon as the file is closed, it is ready to be output to the spooled device. Therefore it is put in a spool out queue. A separate task takes files from that queue, outputting the files one by one on FIFO basis. The file is delelted from the spool DAD afterwards.

FORMAT 2

```
SP LPda,N                No page skipping
```

In the current file to be unspooled on the printer, all skips to Top Of Form are removed. A skip to the next page is only done by MAS, when a number of lines has been printed, equal to the value, specified at generation time or via the DLP command, for the number of lines per page for the printer.

FORMAT 3

```
SP dnda,D                Delete
```

The current file to be (un)spooled is deleted, output to the spooled device stops. Input to the spool DAD is flushed until an :EOJ or :EOB is found.

FORMAT 4

```
SP dnda,R                Rewind
```

The current output file is rewound and re-output (only for output devices).

FORMAT 5

```
SP dnda,W                Warm start
```

The output files from a previous run, which have not yet been output, are unspooled onto the specified device.

FORMAT 6

```
SP LPda,B                Backspace
```

Spooling is restarted from the beginning of the current page (LP only).

FORMAT 7

```
SP dnda,C                Correct
```

The spooling is resumed on the specified device, following an error. Also, this command undoes the effect of the SP LPda,N command.

The following error messages may be output:

    PARAM MISSING
Only SP was input without any parameter
    PARAM MISTK
The first parametr (dnda) did not have four, or the second parameter had more than one character.
    INV DEV NAME
The specified device name (dn) was not LP, PP, PL or CR.
    INV DEV ADDR
The device address (da) did not contain 2 hexadecimal characters or was not generated in the system.
    DEVICE ATTACHED
The specified device was attached to another task than the spool task.
    SPOOL TABLE NOT CREATED
The specified device exists in the system, but was not declared as a spool one at sysgen.
    INVALID COMMAND
SP dnda,N or B was given while the device (dn) was not LP, or CR dnda with a parameter unequal to D was given.
    SPOOL NOT START
An SP command with a second paramter (#W) was given without a previous start or warm start spooling command.
    DEV ALREADY SPOOL
A second start or warm start spooling was given.
    UNKNOWN COMMAND
Second parameter was not equal to N, W, B, C, D or R.
    A FC OTHER THAN 02 ASSIGNED TO LP
During an SP command for the lineprinter (except C and W) another fc than 02 was assigned to it.
    ASSIGN MISTK
SP CRda,D given but the file that the spooling process was creating could not be found.
    WRITE DISK MISTK
Writing to the spool DAD an I/O error occurred.
    GET BUF ERROR
Not enough space in System Dynamic area to create tables for the spooling process.
    DAD D:SPCR NOT FOUND
On the system disk (filecode /C0) no DAD D:SPCR could be found.
    ASSG ERROR
Impossible to assign a file on the spool-in DAD, or the filecode, assigned at system generation time to the cardreader has been scratched or has been assigned to another device.
    SPO: DAD NOT FOUND
The spool DAD belonging to the specified spool-out device (D:SPLP, D:SPPP or D:SPPL) could not be found on any of the discs known to the system.
    SPO: ASSIGN ERROR (SPOOLED DEVICE)
The filecode assigned during system generation to the output device has been scratched or has been assigned to another device.
    SPO: I/O ERROR ON SPOOL DAD
I/O error detected during initialisation of the output spool DAD during execution of the SP dnda or the SP dnda,W command.

FORMAT

```
WM a,v0[,v1]...
```

a — 5 hexadecimal digits, giving the absolute address of the first word to be modified.

v0,v1 — the values to be written in the word(s) whose absolute addresses are a+0, a+2, ... a+2n. Each value must be 4 hexadecimal digits.

The WM operator command performs similar functions to the SCL WRM command. It is provided in order to keep the WRM facility, even though the SCL BYE command has been given. The system machine transient area is blocked temporarily.

One of the following error messages will be output if a WM operator command is rejected:

```
PARAM MISSING
PARAM ERROR
```

OPERATOR MESSAGES AND REPLIES

Overview

Messages from MAS to the operator are always sent to the filecode /EF of the system machine. There are two operator messages, PU and DKER. Only the PU operator message requires an operator reply, which may be RY or RD.

Replies are also entered from the device defined by filecode /EF of the system machine. Replies are entered in exactly the same way as operator commands, namely, press the control panel interrupt button, wait for the message M: to be output, and then enter the reply. Backspace and cancel keys may be used to correct erroneous key depressions. Replies must be terminated by CR.Filecode /EF of the system machine should be assigned at SYSGEN to an interactive device such as a console.

As well as operator messages and replies, this filecode is also used for:

Entering operator commands.
Error messages for rejected operator commands.
Messages from LKM 6 (Pause).
Messages from BCL PSE, REQ, ROI, REL and MES commands.
Messages for LKM 25 (Read Unsolicited Key-in).

FORMAT 1

```
DKER da,CYLc,RECr,s
```

da	disc address (2 hexadecimal digits);
c	cylinder number (4 hexadecimal digits);
r	sector number (4 hexadecimal digits): - 2 leftmost digits = track number; - 2 rightmost digits = sector number.
s	hardware status (4 hexadecimal digits); 8000 means the disc was already not operable and since then, no ready interrupt is received.

The engineer should be called to service the disc (unless s = 8000 or 8001). No operator reply is necessary. The system returns to the user program with a status code. The user program decides whether to exit or not.

FORMAT 2

```
DKER da,RDNxx,[FX|RM],CYLyyy,HDzz,SCpp,(RSNqqqqqq),rrrr
```

da	disk address
xx	relative disk number (00 - 03)
FX|RM	fixed part (FX) or removable part (RM)
yyy	cylinder number
zz	head number
pp	sector number
qqqqqq	real sector number
rrrr	status

The Format 2 DKER is output, whenever an error occurred on a CDC-CMD disk. For other disks, the Format 1 DKER is applicable.

Physical Unit Intervention Required

FORMAT

PU dnda,s[,RY]

dn	device type code. See Chapter 3.
da	device address.
s	hardware status code.
RY	If present, the problem can be solved by some operator intervention (e.g. removing a card jammed in the card reader). When this has been done, the operator should give an RY reply. An RD reply should be given if the operator cannot fix the device. If RY is not present, the engineer is required to service the device. No reply is necessary, since MAS will return to the program which issued the LKM, with a status code.

FORMAT

```
RD da
```

da	the device address specified on the previous PU operator message. The PU message must have ended in RY.

The system returns to the user program which issued the I/O request, with a status code in A7 or in the ECB. The user program decides whether to exit or not.

One of the following messages will be output to /EF if a RD reply is rejected:

```
BAD DEVICE ADDR
DEV. ADDR TOO LONG
DEV. ADDR UNKNOWN
DEV. NOT IN RY
```

Retry an I/O Operation

FORMAT

RY da

da	the device address specified on the previous PU operator message. The PU message must have ended with RY, otherwise a retry is not allowed. The operator should perform the required manual intervention on the device before giving the RY reply.

One of the following messages will be output to /EF if a RY reply is rejected:

```
BAD DEVICE ADDR
DEV. ADDR TOO LONG
DEV. ADDR UNKNOWN
DEV. NOT IN RY.
```