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

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

```
                  _________________
High Address     | P-register (A0) |
                 |_________________|
                 |       PSW       |   Interrupted PSW
                 |-----------------|
                 |       A1        |
                 |-----------------|
                 |       A2        |
                 |-----------------|
                 |       A3        |
                 |-----------------|
                 |       A4        |   Saved Registers of
                 |-----------------|   interrupted routine
                 |       A5        |
                 |-----------------|
                 |       A6        |
                 |-----------------|
                 |       A7        |
                 |-----------------|
                 |       A8        |
                 |-----------------|
Low Address      |                 |
                 |_________________|   (A15)
```

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:-

```
     ____________________________________________________________
Bit | 0              5   6   7   8   9   10  11  12  13  14  15  |
    |____________________________________________________________|

   Priority level |     |     |   |   |   |   |   |   |   |
   Condition Register __|     |   |   |   |   |   |   |   |
   Run _______________________|   |   |   |   |   |   |   |
   Interrupts Enabled ____________|   |   |   |   |   |   |
   Control Panel Interrupt ___________|   |   |   |   |   |
   Power Failure _________________________|   |   |   |   |
   Real Time Clock ___________________________|   |   |   |
   Extended Mode (P858, P859, P854, P876)_________|   |   |
   Memory Protect ____________________________________|   |
   System/User Mode ______________________________________|
```

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:

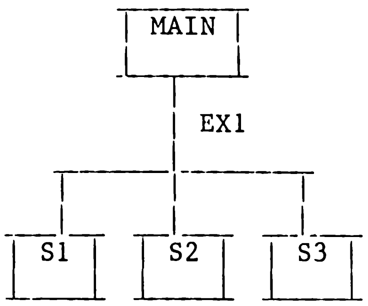
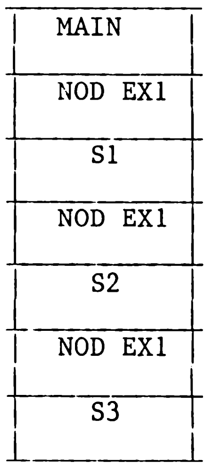


Fig. 2.5 Overlay Structure Example

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:

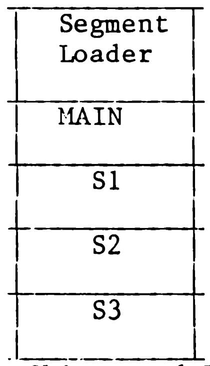


Fig. 2.6 Temporary Object and Load Module Layout Examples

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: #

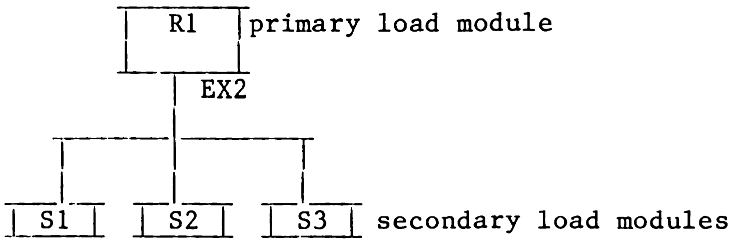


Fig. 2.7 Memory Resident Overlay Structure Example

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:

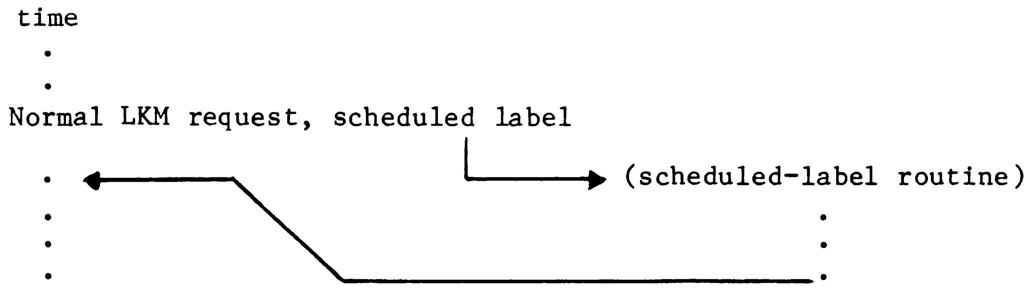

time
Normal LKM request, scheduled label
(scheduled-label routine)

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:

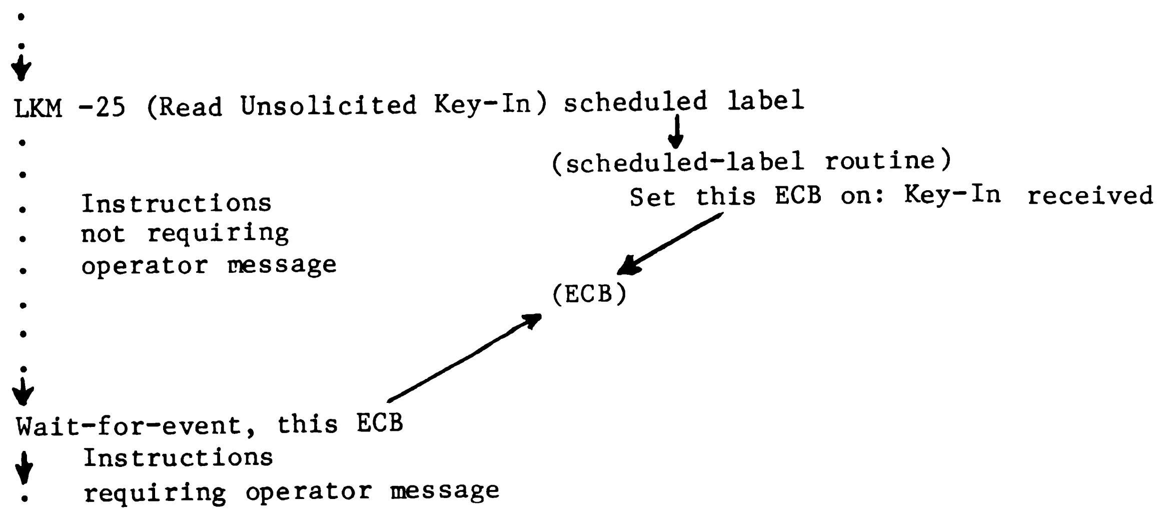

LKM -25 (Read Unsolicited Key-In) scheduled label
(scheduled-label routine)
Set this ECB on: Key-In received
Instructions
not requiring
operator message
(ECB)
Wait-for-event, this ECB
Instructions
requiring operator message

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).

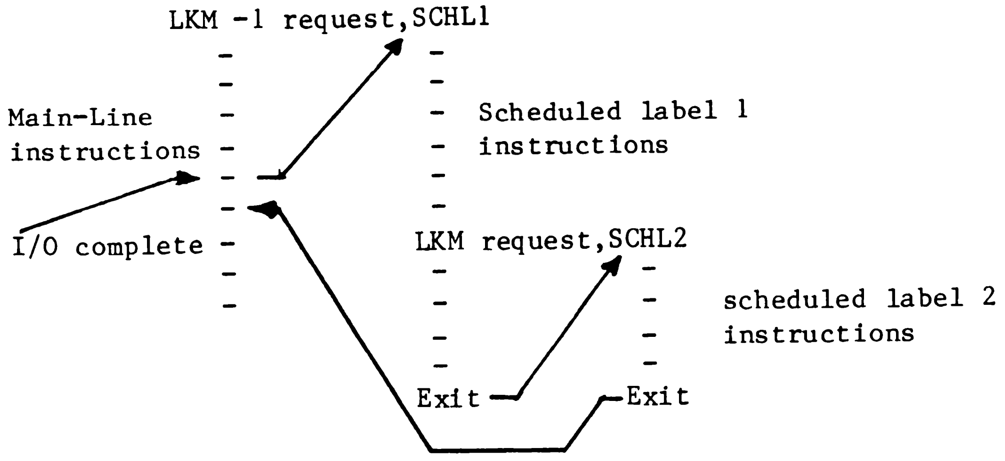


Fig. 2.11 Example of Nested Scheduled Label Routines

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:

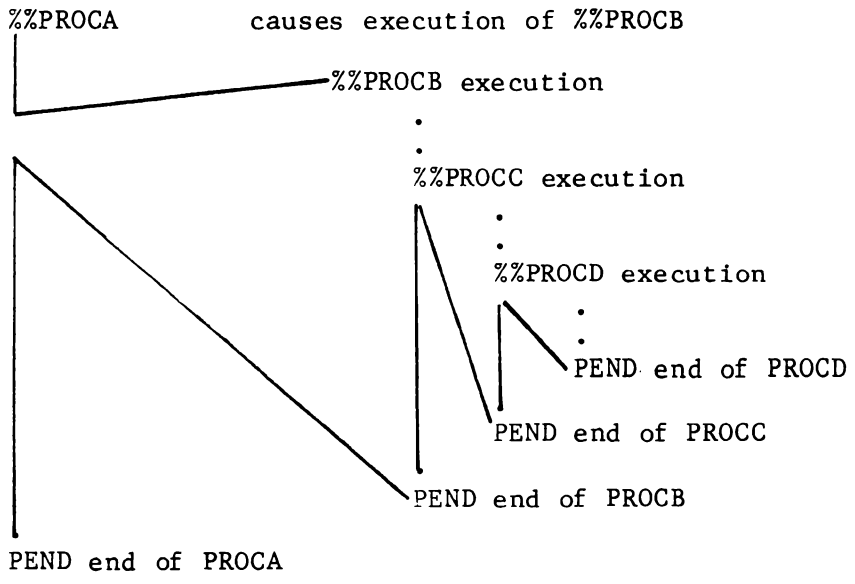


Fig. 2.12 Nesting of Catalogued Procedures

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) SCL and FCL

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:

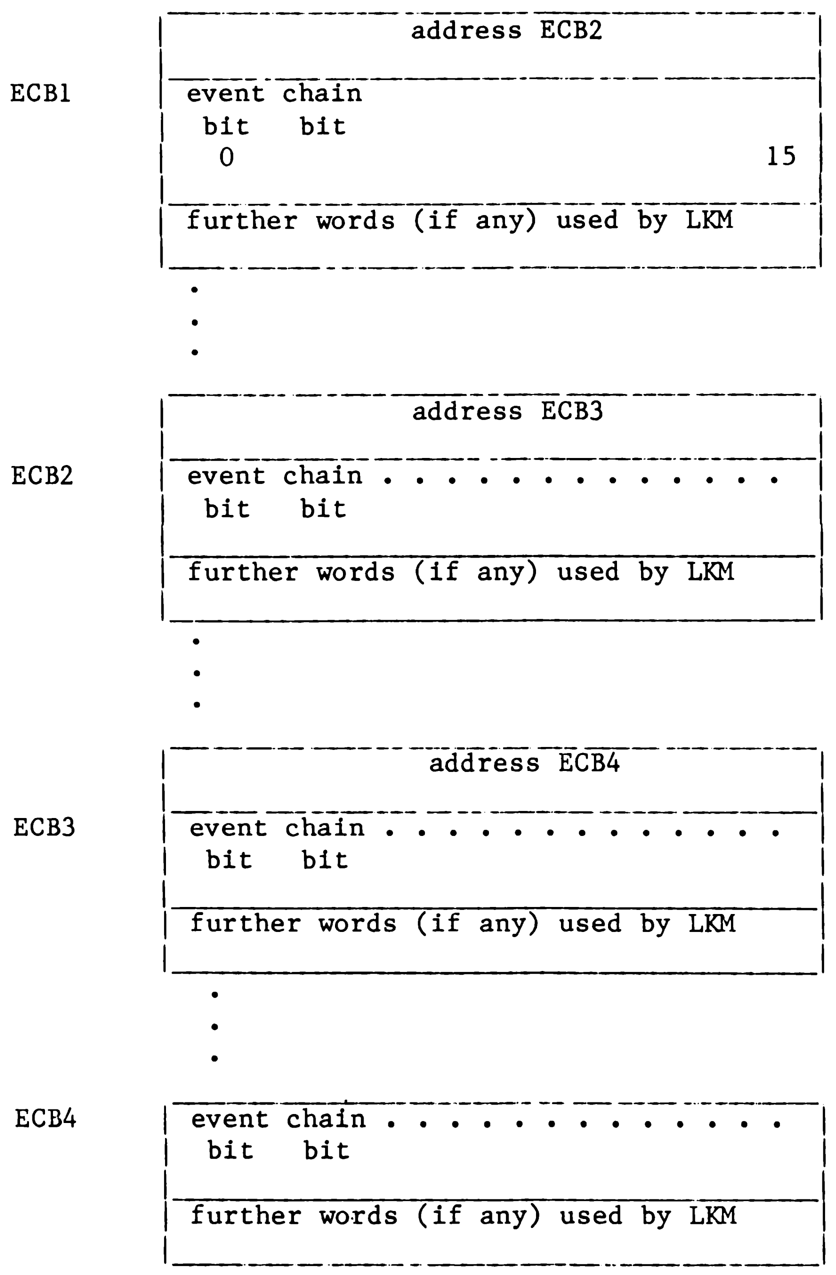

address ECB2
ECB1
event chain
bit bit
0
15
further words (if any) used by LKM
address ECB3
ECB2
event chain
bit bit
further words (if any) used by LKM
address ECB4
ECB3
event chain
bit bit
further words (if any) used by LKM
ECB4
event chain
bit bit
further words (if any) used by LKM

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 (C1-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: cccchhhhrrrr (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.

<u>CDC-CMD discs</u>

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.

<u>FLOPPY disc</u>

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.

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
```

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.

Start the Background Machine

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

d	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.
m	one or two digits 1 to 12, representing the month.
y	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 /CO) 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.

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
```

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.
```

PART 2 THE SYSTEM MACHINE

GENERAL

Each P800M configuration running under MAS, as has been explained previously, contains:

- A system machine;
- Optionally, one or more foreground machines;
- Optionally, a background machine.

MAS itself runs in the system machine and the user's tasks in the other machines.

Prior to the first user session a System Generation must be performed. This is described in Appendix A of this manual.

Prior to every subsequent user session, if MAS was previously deleted, it must be loaded into the bare configuration by a procedure known as Initial Program Load (IPL). This is described in Chapter 4, Operation.

After IPL, MAS initialises its internal tables and makes certain checks on the devices declared at system generation time. Error messages that may be output at this point are listed in Chapter 4.

MAS is now ready to process System Command Language (SCL) commands and Operator Commands, both of which are mentioned in this Chapter.

SCL statements define the characteristics of each machine. Operator commands control the operation of each machine. SCL and operator commands complement the facilities provided by the Foreground Command Language (FCL) and Background Command Language (BCL) commands.

SYSTEM COMMAND LANGUAGE (SCL)

SCL commands define the user applications to be peformed in the current MAS session before the machine is started, and allow re-definition whilst it is being processed.

This Chapter describes the SCL commands related to the system machine and the specific SCL commands for defining foreground or background machines.

MAS receives SCL Commands through the command interactive device, using file-code /E0. Filecodes /02 and /01 are used to log and correct SCL commands, respectively. These file-codes need not be assigned to separate devices.

FILECODES

Filecode /E0 represents any interactive device, such as the console or a VDU. MAS will output:

```
    FCL:
```

whenever it is waiting for the next SCL command. (FCL is output when either SCL or FCL commands may be input, because they are both processed by the same MAS routine.) The system manager is then free to enter another command when required.

Filecode /02 represents a file on which all accepted commands will be logged. It must be a sequential output file, such as a printer, sequential disc file, paper tape punch or console. If it is assigned to the same device as filecode /E0, the commands (if accepted) are logged immediately below the entered command. Filecode /02 may be assigned to NO if the commands are not to be logged.

Filecode /01 represents a device from which erroneous commands may be corrected. This device must be interactive.

CORRECTIONS

An erroneous command results in the following being output to filecode /01:
- The erroneous command;
- An error message;
- The message:
      FCL:

MAS then waits for the command to be re-input from the console, which is assigned to filecode /01. This correction procedure is repeated until the command is accepted as correct. MAS will now continue reading commands through filecode /E0, unless the correction given is BYE, which ends the FCL sessionfor the machine.

It is not always possible to change the type of command during a correction. In some cases the erroneous command may have been partly processed. Commands of this type are denoted as such in the descriptions, in alphabetical order, which appear in Chapter 7.

Error Messages

Two error messages apply to all SCL commands; they are:
- INPUT COMMAND I/O ERROR
  An I/O error occurred when trying to read the next command through file-code /E0 or /01.
- COMMAND UNKNOWN
  The three character mnemonic is not one of the set of permitted commands.

Other error messages are related to one or more SCL commands, and are explained in the descriptions of the SCL commands which appear in Chapter 7.

Syntax

The rules for the construction of all commands are described in Appendix B to this manual.

Processing SCL Commands

The task which processes the SCL or FCL commands relevant to one user machine runs concurrently with other tasks in the bare machine. Whenever the SCL/FCL task is suspended awaiting operator input, these other tasks continue.

Waiting and Stopping

Once MAS has indicated that it is ready to accept a further SCL command by the message:
    FCL:
the SCL task will be held in the wait state. It will be restarted when the next valid command has been entered.

To stop the SCL task, the SCL command 'BYE' is necessary. The area assigned to the task and the devices used for filecodes /E0, /01 and /02 are freed and the task exits. No further SCL commands can be given.

It is necessary to stop the task in hand whenever a task is to be started for another user machine and there are not enough free devices for use through file-codes /E0, /01 and /02 for the new task.

For example, after defining all user machines for this MAS session with SCL commands the BYE command could be given. The SM (start machine) operator command could then be given for one of the defined foreground machines and MAS will be able to re-use the same devices as were used in the definition. When all the commands necessary to start the application in the user machine have been given, the BYE command can be given again. This starting sequence can then be repeated until all required machines are running.

Once a BYE command has been given, the only control over that machine is by SCL or operator commands to the system machine. If the BYE command is given to the system machine, the only control is by operator commands.

COMMANDS

SCL commands allow the system manager:
- To set the time and date;
- To obtain the time and date;
- To define user machines;
- To indicate that certain devices and memory pages are not available to MAS;
- To dump memory and disc areas assigned to the system machine;
- To indicate that floating-point registers are not to be saved when a particular task is interrupted.

CATALOGUED PROCEDURES

A catalogued procedure is a string of SCL commands related to a particular task, which is catalogued under a procedure name.

SCL commands which are expected to be repeated in the same, or nearly the same, format may be saved as a catalogued procedure. Once set up they may be invoked by a single catalogued procedure call, where the parameter values may be specified or changed at the time the procedure is called.

SCL catalogued procedures are held in a file named S:PROC on the system DAD, which is accessed through filecode /F6.

The construction and use of catalogued procedures are described in Chapter 2 of this manual.

ABT **Abort a Program** ABT

Effect on Background

Format ABT

Remarks

When used in the system machine the ABT command aborts the program, running in the Background machine. As the routine, interpreting the command is the same for System and Foreground, a program name may be given. However, a specified program is searched in the system machine and a system program cannot be aborted.

Errors

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

```
    PARAM ERROR
```

A parameter was given, that could not be a program name.

```
    SYSTEM PROG CANNOT BE ABORTED
```

A program name was given to the command, the program was searched in the system machine and found.

```
    PROG INACTIVE
```

The Batch machine was not yet started or did read an :EOB command.
A program name was given in the command, which could not be found in the system machine.

```
    BCP PROCESSOR CANNOT BE ABORTED
```

The program running in the Batch machine was the BCP processor.

```
    PROG ALREADY ABORTED
```

The background program was already in the abort state.

Effect on	System
Format 1	ASG fc1,fc2 Assign a filecode to another filecode
Format 2	ASG fc1,dn[da[ln]] Assign a filecode to a physical device
Format 3	ASG fc1,DDdc,ft,fn Assign a filecode to a disc catalogued file
Format 4	ASG fc1,DDdc,ft[,[ng][,NC]] Assign a filecode to a disc temporary file
Format 5	ASG fc1,dk,dad Assign a filecode to a DAD

fc1	is the filecode to be assigned and for which an entry is to be created in the system machine filecode table.
fc2	is a filecode which already exists in the system machine filecode table and to which the filecode `fc1' is to be assigned.
dn	is a 2-character device-type code; see Chapter 3.
da	is the device address; if omitted, the first device type dn that is found by MAS will be assigned.
ln	is a 2-character linenumber that can be specified for devices connected to the AMA8. This linenumber may also be added to the AMA8 device address in the da parameter.
DDdc	is the DAD filecode, DDF0 thru DDFF. (This DAD filecode must havebeen assigned in the system machine)
ft	is the file-type code, UF, SC, LM or OB. (also EF is accepted)
fn	is the file-name. (The file with file-name `fn', file-type `ft' and version 0 must exist in the directory of the first user of DAD `dc'.)
ng	is the number of granules to be reserved; the default value is 1, the maximum value is depending on the next parameter.
NC	is for non-consecutive granules; the default is consecutive.
dk	is the filecode of the disc containing the DAD.
dad	is the DAD name: 1-6 ASCII characters, left justified.

Errors

One of the following error messages will be output if an ASG command to the system machine is rejected:

WRONG FILECODE
An illegal fc1 parameter was given (all formats)

NOT ENOUGH PLACE IN DYNAMIC AREA
No place in the system dynamic area could be found to allocate buffers for assignment tables (all formats)

TOO MANY PARAM
More parameters were specified than expected (formats 1, 2 and 3)

I/O ERROR
Reading a buffer from disc (e.g. a GRANTB or a VTOC) an I/O error occurred (formats 1, 3, 4 and 5)

TOO MANY FC ASSIGN
The number of filecodes assigned in the system machine became greater than the maximum number of filecodes allowed, as recorded in the CVT (communication vector table). For adaption of this value see Appendix A, System Generation. (all formats)

DELETE FC ERR ST=abcd

The filecode to be assigned was already assigned and had to be deleted. This deletion went wrong. For an explanation of the status codes see Appendix C LKM 24 (all formats).

2ND FC NOT ASGN

In the Format 1 assign, the filecode fc2 was not assigned.

WRONG PARAMETER

In the Format 2 assign, the dn[da[ln]] parameter did not contain 2, 4 or 6 characters.

DEVICE UNKNOWN

The dn[da[ln]] specified a device that was not generated into the system (format 2) or the /Cx filecode was not assigned.(format 5)

WRONG FILE TYPE

The ft parameter was not SC, LM, OB, UF or EF. (format 3 and 4)

ERROR IN LAST PARAMETER

In the Format 3 assign, a 5th parameter was specified, but it was not NC.

DAD NOT FOUND

The specified DAD filecode did not exist (format 3 and 4)

DAD OVERFLOW

More granules requested than there were free ones in the DAD (format 3)

TOO MANY SECTORS

In the Format 3 assign, for a consecutive file more than 32767 sectors were requested or for a non-consecutive file, more granules were requested than could be recorded in the GRANTB.

WRONG FILENAME

An illegal filename was given (format 4) or file-type EF was given for a temporary file (when file-type EF is given, MAS automatically assumes a format 4 assign and expects the next parameter to be a filename)

DAD FILECODE DOES NOT EXIST

No F0 - FF was given in the dc parameter (format 4)

NO USERID

An attempt was made to assign a file in a DAD that contained no Userids in its Catalog (format 4)

FILE UNKNOWN

The file to be assigned did not exist (format 4)

ASSIGN ERROR STATUS=/xxx

An assign to a TDFM file failed (foramt 4) for an explanation of the status, see Appendix C LKM 23.

DATA DISC CANNOT BE ASSIGNED

A Format 5 assign was given for a DAD on an type F3 flexible disc.

WRONG DAD NAME

The DAD name in the Format 5 assign contained more than 6 characters.

DAD NOT FOUND

A DAD with the specified name did not exist on the disc, indicated by the /Cx filecode (format 5)

Note:

If an error occurred during the assign, it is possible that the old assignment does not exist anymore, because the error has been found after deletion of the old assignment.

Effect on System, Foreground and Background

Format 1 BYE [m1[,m2]...]

Format 2 BYE [$$,][m1[,m2]...]

$$ is an indication to the system that the open spool files have to closed and unspooled. This feature is only available from MAS release 8.50.

m is the name of a machine. If this optional parameter is entered, the specified machine(s) will be started as if an 'SM' (when BYE machid or BYE SYSTEM) or 'SB' (when BYE BATCH) command had been given.

Remarks

The use of the BYE command will cause the current SCL task to exit. All dynamic storage used by it will be freed and no more commands can be entered for this task. After a BYE command has been processed, the devices assigned to filecodes /E0, /01 and /02 are available to other tasks within the current machine.

Till MAS release 8.50, the lay-out of the BYE command is Format-1. The open spool files are not closed and unspooled. To unspool these files, the CLS command should be used.

Set the Clock

Effect on	Background, System and Foreground.
Format	CLK hh,mm[,ss]
hh	is the hour; two digits, in the range 00 to 23.
mm	is the minute; two digits, in the range 00 to 59.
ss	is the second; two digits, in the range 00 to 59, with a default value of 00.

Remarks

The CLK command is given to set the clock in the system machine, and normally should only be given just after IPL (Initial Program Load) and before starting any user machine. If a power failure occurs the CLK command can be used to reset the clock when the power is restored. However, the CLK command can be given at any time.

If the CLK command changes the clock value from, say, 23,30,00 to 01,45,00 or vice versa, the date is not changed.

A self-explanatory error message will be output if a CLK command is rejected.

Effect on Background, System and Foreground.

Format 1 CLS [m1[,m2]....]

Format 2 CLS [/fc1|ALL]

m1,m2	are machine names. It is an optional parameter, if entered the specified machine will be started, as with an SM or SB operator command.
fc1	is a spooled filecode. The spooled file assigned to this filecode will be closed and unspooled.
ALL	indicates that all spooled filecodes have to be closed and unspooled.

Remarks

Format 1 of the CLS command applies for MAS releases until MAS 8.50. The same functions as for the BYE command are performed, but also all spool files are closed and unspooled.
Format 2 of the CLS command applies from MAS 8.50, it closes and unspools one or all spool files.

Effect on Foreground.

Format CMA p[,lib]

p is the number of 2K pages (1 to 16) in the communications area. This defines Segment 0, which is addressable by all memory and disc-resident programs in this foreground machine using virtual addresses not exceeding 32KW. If 16 pages are specified, the foreground machine comprises solely Segment 0 and no disc-resident programs or other segments will be allowed for this foreground machine.

lib specifies the number of characters in the Public Library Area that is, the size of all programs loaded by LOD commands into Segment 0. The remainder of Segment 0 is available for dynamic buffers, to be obtained when any user program in this foreground machine issues an LKM 4 (Get Buffer). The default value is zero.

Remarks

The CMA command must immediately follow the DCF command referring to the same machine.

Errors

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

```
WRONG NUMBER OF PAGES
DCF OR DCB COMMAND IS EXPECTED
TOO MANY PARAMETERS
NUMBER OF PAGES >16
WRONG PUBLIC LIBRARY SIZE
PUBLIC LIBRARY SIZE TOO BIG
NOT ENOUGH FREE PAGES
ONLY ONE CMA ALLOWED
```

Note: Because Segment 0 is always part of the current active program area, this area + Segment 0 must not be greater than 32KW.

Effect on	Background, System and Foreground.
Format	DAT dd,mm,yy

dd	is the day; two digits representing a valid day.
mm	is the month; two digits in the range 01 to 12.
yy	is the year; two digits representing the last two digits of a year in the twentieth century. Thus 28,02,00 is accepted as the 28th February, 1900.

Remarks

The DAT command is given to set the date in the System (Initial Program Load). If a fatal power failure occurs the DAT command can be used to reset the date when the power is restored. However, the DAT command can be given at any time.

A self-explanatory error message will be output if a DAT command is rejected.

Effect on Background.

Format DCB [size][,level]

size	specifies the number of pages to be reserved exclusively for a non-swappable background machine. If omitted, a swappable background machine is assumed.
level	specifies the software level of all programs which will be run in the background machine. If omitted, the highest free level but two less than the level of the Idle Task is assumed. If specified, it must never be a lower level than the default value, and cannot be connected to any other program.

Errors

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

```
    COMMAND NOT ALLOWED (DCB command not given in the System Machine)
    MACHINE ID ALREADY EXISTS (DCB command already given without KIM BATCH)
    WRONG BATCH SIZE (first parameter is not numeric or >16)
    WRONG PARAMETER (2nd parameter not numeric)
    TOO MANY PARAMETERS (more than two parameters given)
    WRONG LEVEL NUMBER (level of Idle Task-1 was specified)
    NOT ENOUGH PLACE TO CREATE MACHINE (no place in system dynamic area)
    LEVEL NOT FREE (the specified level was already occupied)
    LEVEL NUMBER TOO HIGH (specified level > than level of Idle Task)
    NO PLACE TO CREATE BATCH (nr of requested pages not available)
    D:CI NOT ASSIGNED (for a swappable Batch, the swap DAD D:CI was not found)
```

Effect on   Foreground.

Format   DCF m[,n]

m   is the name of this foreground machine (a maximum of six ASCII characters). The names SYSTEM and BATCH are not allowed. Each foreground machine in any one session must have a unique name.

n   is the number of memory resident segments, other than Segment 0 (the communications area). The default value is zero. (There must be `n' SEG commands, one for each segment, defining the size of these segments.) The maximum number of segments is 9. This value is recorded in the CVT (communication vector table) and can be changed during system generation. See Appendix C.

Remarks

The DCF command must be the first SCL command in the set defining a foreground machine.

If all the memory-resident foreground programs, including their dynamic buffers obtained by LKM 4, occupy less than 16 pages, there is no need to specify any memory-resident segments. Segment 0 must then be specified on all FCL LOD commands given after the machine has been started.

However, disc-resident programs for a foreground machine must be less than 16 - v pages (where v is the number of pages in Segment 0). Segment 0 should therefore only contain memory-resident programs which are required by the disc-resident programs and/or memory-resident programs of the same machine (but not those loaded into Segment 0).

Some machine names starting with 'X:' like X:IO, X:MASG, X:IDLE cannot be used as these names are used internally in the monitor.

For a Foreground machine, a Program Control Table (PCT) is created, which has to be connected to a level. The highest free level is searched and connected to the Foreground machine, however, the levels for the Idle Task I/O (Idle Task level-1) and the Batch (Idle Task level-2) are not taken.

Errors

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

```
    COMMAND NOT ALLOWED (DCF given not in System machine)
    WRONG PARAMETER (illegal machine name)
    MACHINE ALREADY EXISTS
    WRONG NUMBER OF SEGMENT (second parameter not numeric)
    TOO MANY PARAMETERS
    DCF COMMAND UNALLOWED DURING DCF
    MACHINE NAME > 6 CHARACTERS
    DCF BATCH NOT ALLOWED
    DCF SYSTEM NOT ALLOWED
    TOO MANY SEGMENTS REQUESTED (more than value in CVT)
    NO SPACE TO CREATE MACHINE (no space in system dynamic area)
    NO FREE LEVEL, CREATION IMPOS.
```

Effect on Background and Foreground.

Format DEN

Remarks

The DEN command indicates to MAS that all the commands to declare a machine have been entered. A further machine may now be declared. If no more machines are required to be declared, the BYE command may be given and the machines started, or other system commands (like MAP, DUM) can be given.

Errors None.

Purpose	To define an assignment of a Datacom line.
Effect on	Foreground and Background.
Format 1	DLC lc1,dn[da[l]]
Format 2	DLC lc1,lc2

lc1	is the linecode to be assigned.
dn	is a device name as described in Appendix A.
da	is the device address. If omitted, the first device with the name dn is taken.
l	is a linenumber. Only applicable for LSM16 and AMA8.
lc2	is an already assigned linecode, to which this linecode has to be assigned by equivalence.

Remarks
The assignment is made in the consecutive linecode table or in an alternate block, depending on the value of lc1 and the DLN command.
When the user has the intention to give a DLN command, this must be done before any DLC command is given.

Errors
One of the following error messages is output if the DLC command is rejected:

```
     LINE CODE ERR
Incorrect value of lc1 given.
     LINE CODE NOT NUMERIC
     LINE CODE MISSING
No parameter was given in the DLC command.
     INV. LINE CODE
lc1 is negative, zero or greater than 255.
     DEV NAME ERROR
     DEV NAME MISSING
Only one parameter given.
     INV DEV NAME
NO device is not allowed.
     2ND LINE CODE ERROR
lc2 is negative, zero or greater than 255.
     TOO MANY PARAM
     DYN AREA OVERFLOW
     DEV. UNKNOWN
The specified device is unknown or not generated in the system.
     2ND LINECODE NOT ASSIGNED
     DEV ADDR ERROR
     ASSIGN ERR (ST=xxxx)
The assignment gave an error (see Appendix C, LKM 48).
```

Purpose	To define the number of consecutive linecodes that can be used in the machine to be declared.
Effect on	Foreground and Background.
Format	DLN n
n	is the number of consecutive linecodes. It is an integer value from 0-255. Before the DLN command, no DLC (define linecode) command may be given.

Remarks

If no DLN command is given, the default value from the CVT is taken.
The DLN command causes the system to reserve a block in the System Dynamic Area of n+1 words. The first word contains the number of linecodes (n), the other words contain the address of an LCB (if assigned). Each line that is assigned to a linecode not greater than n is inserted in this block on the displacement that corresponds with the linecode assigned. When a line is assigned to a linecode greater than n then a alternate linecode assign block is created, which is chained to the other blocks containing linecode assignments (i.e. alternate blocks and the block containing the consecutive linecodes). The advantage of having a block with consecutive linecodes is that it costs less System Dynamic area.

Errors

One of the following error essages will be output when the command is rejected:

```
    # OF DTC L.C. ERROR
n is incorrect
    # OF DTC L.C. NOT NUMERIC
    TOO MANY DTC L.C.
n is negative, zero or greater than 255.
    WHAT IS THE 2ND PARAM?
More than one parameter given.
    DYN AREA OVERFLOW
    SEQUENCE ERROR
Some linecode was already assigned, receiving the DLN command.
```

Purpose To define or redefine the number of lines per page for a Teletype or line printer.

Effect on All machines.

Format DLP dn[da],lp

dn is the device name (TY or LP).
da is the device address (2 hexadecimal digits without the preceding slash (/) character).
lp is the number of lines/page. The permissible range is 1-99 or NO. If NO is given, no top of form skipping is performed.

Remarks

If the device address parameter (da) is omitted, all devices of type `dn′ will have the requested number of lines per page.

Errors

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

```
DEV. NAME ERROR
DEV. NAME MISSING
DEV. ADDR NOT HEXA
INCORRECT DEV. NAME & ADDR.
UNKNOWN DEVICE
# OF LINES PER PAGE MISSING
INVALID # OF LINES/PAGE
# OF LINES/PAGE ERROR
TOO MANY PARAM
PAGE CONCEPT NOT APPLICABLE
```

Effect on   Background, System and Foreground.

Format   DOF dnda

dn	is a two character non-disc device-type code. (Device-type codes are listed in Chapter 3.) A disc cannot be specified.
da	are two hexadecimal digits, representing the device address.

Remarks

The DOF command is used by the systems manager to inform MAS that a device may not be used. It is entered because a device requires repair, or if systems generation specified a device which has not yet been installed. Its function is identical to the OF operator command.

Errors

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

```
    DEV. NAME ERROR
    DEV. NAME IS NUMERIC
    DEV. NAME IS MISSING
    INCORRECT DEV. NAME & ADDR
    DEV. ADDR NOT HEXA
    UNKNOWN DEVICE
    DISK DEVICE
    WHAT IS THE 2ND PARAM?
```

Effect on Background, System and Foreground.

Format DON dnda

dn	is a two character non-disc device-type code. (Device-type codes are listed in Chapter 3.)
da	are two hexadecimal digits, representing the device address.

Remarks

The DON command is used by the systems manager to inform MAS that a device, previously specified by a DOF or OF operator command, may now be assigned under MAS. Its function is identical to the ON operator command.

Errors

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

```
    DEV. NAME ERROR
    DEV. NAME IS NUMERIC
    DEV. NAME IS MISSING
    INCORRECT DEV. NAME & ADDR
    DEV. ADDR NOT HEXA
    UNKNOWN DEVICE
    DISK DEVICE
    WHAT IS THE 2ND PARAM?
```

Purpose	To set or redefine the timeout value for a non-disc device.
Effect on	All machines.
Format	DTO dn[da],tm
dn	is the device name (2 ASCII alphabetic characters).
da	is the device address (2 hexadecimal digits without '/').
tm	is the timeout value (expressed in minutes in the range 0-255). Zero means no timeout.

Remarks

If the da parameter is omitted, the timeout value will be set for all devices of type dn.

Errors

If this command is rejected, one of the following error messages will be output:

```
    TOO MANY PARAM
    DEV. NAME ERROR
    UNKNOWN DEVICE
    DISK DEVICE
    INVALID TIMEOUT (greater than 255)
    DEV. ADDR. NOT HEXA.
    INCORRECT DEV. NAME & ADDR.
    TIMEOUT ERROR
    TIMEOUT MISSING
    DEV NAME MISSING
```

Effect on System and Foreground.

Format DUF fc,from[,to]

fc	is either a DAD filecode /F0 to /FF or a filecode assigned to a discfile. The file-code must have been assigned at system generation or by an ASG command.
from	is the first sector to be dumped.
to	is the last sector to be dumped, if omitted, only the from sectoris dumped.

For a DAD, from and to are DAD or file relative sector numbers.

For a disc file, relative sector 0 is the third sector. It is the sector following the fileheader and the granule table sector.

Remarks

The DUF command will dump the specified sectors, in hexadecimal format, to the device assigned to system machine filecode /02.

Errors

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

```
FC PRT NOT ASGN
PARAM ERROR
PARAM MISSING
DAD FC NOT ASGN
OUT OF LIMITS FILE
DYNAMIC AREA OVERFLOW
READ FILE I/O ERROR
2ND ADDR LESS THAN 1ST ADDR
FC FORBIDDEN
FC ASSIGN TO PHYS DEVICE
```

Effect on System.

Format DUM from,to

from	is an upto six digit hexadecimal number representing the 18-bit (P857/P858) or 20-bit address of the first word to be dumped.
to	is an upto six digit hexadecimal number representing the 18-bit or 20-bit address of the last word to be dumped.

Remarks

The memory locations specified within the system machine are dumped to the device assigned to filecode /02. If this device is not ready MAS sends a message to the operator.

Errors

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

```
    FC PRT NOT ASGN
    DYNAMIC AREA OVERFLOW
    PARAM ERROR
    PARAM MISSING
```

Effect on Background and Foreground.

Format FCD {dk | fc,dn[da[l]]| dc,dkl,dd}

fc	is a filecode.
dn	is a two-character non-disc device-type code. (Device-type codes are listed in Chapter 3.)
da	is a device address, a 6-bit hexadecimal number. This address is only required if there is more than one device of type `dn' and the filecode is to be assigned to one other than the first one in the systems generation device table.
dc	is a DAD code in the range /F0 thru /FF.
dk	is a disc code in the range /C0 thru /CF.
dd	is a DAD name contained in the VTOC (Volume Table of Contents) on disc `dkl'.
l	is a line number that can be specified with a device address connected to an AMA8 device. The linenumber can also be added to the AMA8 device address.
dkl	is a disc code in the range /C0 thru /CF, which has to be declared in a previous FCD command for the same machine.

The FCD command defines a filecode for a file when the machine is created.

Remarks

The three types of format for a FCD command, as shown above, define filecodes for physical discs, physical devices and DADs, respectively.

After a machine has been defined, additional filecodes may be defined by the ASG command or by LKM.

Errors

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

```
WRONG FILE CODE
WRONG PARAMETER
NO MORE SPACE FOR FC CREATION
DISC CODE MUST BE Cx
WRONG DAD NAME
FILE CODE UNKNOWN
/Fx MUST BE USED WITH /Cx
NUMBER OF FC TOO BIG
/Cx NOT ASSIGN
```

Purpose Floating-point registers are not to be saved.

Effect on Background, System and Foreground.

Format FOF

Remarks

The FOF command cancels the effect of a previous FON command. The contents of floating-point registers will no longer be saved after this command has been processed. There is no theoretical limit to the number of times that the FON and FOF commands may be given in a MAS session. Also, there is no check, whether the floating point processor was on.

Errors

If the floating-point hardware feature is not present in the P800 configuration and a FOF command is given, it will be rejected with the following error message:

```
    NO FLOATING POINT OPTION
```

Purpose   Floating-point registers are to be saved.

Effect on   Background, System and Foreground.

Format   FON

Remarks

The FON command causes MAS to save the contents of floating-point registers if the task using them is interrupted. This command should be given before any task begins to use floating-point arithmetic, otherwise the register contents will be lost if the task is interrupted. The FON command is cancelled by the FOF command.

Errors

If the floating-point hardware feature is not present in the P800 configuration and a FON command is given, it will be rejected with the following error message:

```
    NO FLOATING POINT OPTION
```

Effect on — Foreground and Background

Format — KIM {machine-name | BATCH}

machine-name	is a valid machine name (maximum six ASCII characters).
BATCH	refers to the background machine.

Remarks

The KIM command 'kills' the foreground machine called 'machine-name', or the background machine if 'BATCH' is specified. This command cannot be used to delete a machine whilst it is running.

If the command is rejected and one of the error messages listed below is printed, the machine still exists.

Errors

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

```
    PARAM ERROR OR MISSING
    UNKNOWN MACHINE
    MORE THAN ONE PARAM
    SYSTEM MACHINE
    PRG  <Prog. Name>   ACTIVE
    PRG  <Prog. Name>   PAGES TAB DESTROYED
    PRG  <Prog. Name>   INV. CHAIN. ADD. IN SG *
    PRG  <Prog. Name>   SYST. DAD /F1 INCORR. **
    PRG  <Prog. Name>   BEING SWAPPED OUT.
    PRG  <Prog. Name>   BEING ABORTED.
    FCL RUNNING FOR THAT MACHINE
```

* Programs in a segment are chained; the chain word has been overwritten.

** DAD filecode /F1, normally assigned to D:CI, is not assigned.

Effect on Background.

Format KLM n

n is the secondary load module name, as specified on the LSM command which was used to load it. Giving a KLM commnad in the system machine applies on a Background secondary load module.

Remarks

The command is rejected if the secondary load module is connected to a primary load module (in the BATCH) which is still active.

Error messages

```
PARAMETER ERROR
SEC. LOAD MOD. ALREADY DELETED
SEC. LOAD MOD. STILL CONNECTED.
```

Effect on Foreground.

Format LAB s

s is the maximum number of scheduled labels which may, at any one moment, be invoked by any one program run for a foreground machine user. The maximum value is 255.

Remarks

The LAB command varies the value which is specified during systems generation. It enables the maximum number of scheduled labels to be reset without re-generating the system.

Errors

The following error message is output if the LAB command is rejected:

```
    WRONG PARAMETER
```

Purpose

To load a secondary load module into memory, which can be called by primary load modules in the Batch machine.

Format

```
LSM n,fc[,R|,W]
```

n is a 4 ASCII character name of a secondary load module, output by the Linkage Editor and catalogued in the first Userid of the DAD fc

fc specifies the DAD filecode from which the secondary load module is to be loaded. Giving the LSM command in the SYSTEM machine, the DAD filecode is searched in the filecode table of the BATCH machine.

R is entered if the secondary load module is to be considered as read-only.

W is entered if the secondary load module may be modified.

The default is R .

Error Messages

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

```
UNKNOWN SECONDARY LOAD MODULE
SEC LOAD MOD IS SEGMENTED
SEC LOAD MODULE ALREADY LOADED
DAD FILECODE ERROR
DYN AREA OVERFLOW
PARAMETER ERROR
DAD SECTOR TOO LONG
I/O ERROR
TOO FEW FREE PAGES
SEC. LOAD MOD. TOO LONG
NO DECLARED BATCH MACHINE
SLM NAME EXCEEDS 4 CHARACTERS
```

Effect on System, Foreground and Background.

Format MAP [filecode]

filecode is the print file filecode; the default is /01.

Remarks

The MAP command prints the status of all programs in all machines.

The format of the print file is as follows:

Example of a MAP print-out

```
SM T
FCL:LOD 0,MAIN,/F6
PROG :  MAIN   USER LOADING ADDR. :   /F3E4
FCL:SWP FUNPO,/F6
FCL:REP 5,1,PREP,/F7
PROG :  PREP   USER LOADING ADDR. :   /DFE2
FCL:RON PRON,/F7
FCL:MAP
T      ** 10.00.00 ** 16.11.78 *
MAIN    SOO  LEV=000 INA LOADED AT F3E4
FUNPO   SWP  LEV=000 INA NOT LOADED
PREP    REE  LEV=000 INA LOADED AT DFE2
PRON    RON  LEV=000 INA NOT LOADED
FCL:BYE SYSTEM
SM SYSTEM
FCL:MAP
SYSTEM ** 10.00.09 ** 16.11.78
X:IO   RES  LEV=000 INA SYSTEM PROGRAM
X:MASG RES  LEV=003 ACT SYSTEM PROGRAM RUNNING
X:USVC RES  LEV=001 INA SYSTEM PROGRAM
X:SWIO RES  LEV=002 INA SYSTEM PROGRAM
X:ALGR RES  LEV=009 INA SYSTEM PROGRAM
X:RTC  RES  LEV=012 INA SYSTEM PROGRAM
X:OCOM RES  LEV=008 INA SYSTEM PROGRAM
SYSTEM RES  LEV=010 ACT SYSTEM PROGRAM MAIN WAIT AT 953C
X:IDLE RES  LEV=119 ACT SYSTEM PROGRAM RUNNING
X:DUMP RES  LEV=013 INA SYSTEM PROGRAM
T      RES  LEV=116 INA SYSTEM PROGRAM
FCL:BYE
```

Errors

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

```
INVALID PRINT FILE CODE
PRINT F.C. ASSGN TO NO DEVICE
```

Maximum Number of Blocking Buffers

Purpose	Define maximum number of blocking buffers.
Effect on	Background and Foreground.
Format	MBF x
x	is the maximum number of disc file management blocking buffers.

Remarks

The MBF command varies the number of buffers specified during systems generation. It enables the number of buffers to be reset without re-generating the system.

Errors

The following error message is output if an MBF command is rejected:

```
    MBF VALUE IS NOT NUMERIC
```

Effect on   Background and Foreground.

Format   MFC f

f	is the maximum number of filecodes that can be assigned in the machine being generated. It is a numeric value not greater than 255.

Remarks

The MFC command varies the size of the table specified during system generation. It enables the size of the table to be reset without re-generating the system.

Errors

The following error message is output if an MFC command is rejected:

```
    MFC VALUE IS NOT NUMERIC
    NUMBER OF FC TOO BIG
```

Purpose	to introduce a new device without the necessity of generating a new system.
Effect on	Background, System and Foreground.
Format	NDV dnda[,i][,p]
dn	is a two-character device-type code; see Chapter 3. This device type must have been specified during system generation.
da	is two hexadecimal digits representing the device address of the new device.
i	is the interrupt level (1 to 61) which specifies the address of the hardware interrupt routine which will process this interrupt signal and the hardware level of this routine. If the device is attached to a controller which has already been declared, this parameter is not allowed.
p	specifies the number of lines per page, and should only be given if the device is a line printer.

Remarks

The NDV command allows the systems manager to add a new device to the system without undertaking a system generation.

Errors

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

```
DEV. ALREADY DECLARED
UNKNOWN DEV. NAME
DEV. INT. MISSING
DEV. INT. ERROR
DEV. INT. NOT NUMERIC
3RD PARAM ERROR
WHAT IS THE 3RD PARAM?
TOO MANY LINES/PAGE
0 LINES/PAGE
WHAT IS THE 4TH PARAM?
DYN AREA OVFL
DEV. CONT. ALREADY USED
DISK DEVICE
TOO MANY DEVICES
DEVICE NAME ERROR
DEVICE NAME IS NUMERIC
DEVICE NAME MISSING
INCORRECT DEV NAME&ADDR
DEV. ADDR NOT HEXA
INV DEV INT LEVEL
```

Purpose To print a map of all machines.

Format PCM [fc]

fc is the print filecode; the default value is /01.

Remarks

Per machine the occupied pages with their related MMU registers are listed.
Also information is given for what the page is used (segments, swappable programs).
The last line printed by the command gives in hexadecimal the number of free pages in the whole machine.

Error Messages

```
INVALID PRINT FILECODE
PRINT F.C. ASSGN. TO NO DEVICE.
```

Effect on System and Foreground.

Format PFC [filecode]

filecode is the print ~~file~~ filecode. The default value is /01.

Remarks

The PFC command prints per machine all assigned filecodes. The filecodes are printed on the device assigned to the specified filecode.

The format of the print-out is as follows:

```
***SYSTEM FILE CODES
01 TY 10
02 LP 07
E0 TY 10
EF TY 10
C0 DK 02  X1215 REMOVABLE PACK 1215
C2 DK 12  X1215 REMOVABLE PACK 0012F0 DD     SUPERV  C0  DK  02
F1 DD     D:CI    C0  DK  02
FCL:
```

Errors

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

```
    INVALID PRINT FILE CODE
    PRINT F.C.ASSGN TO NO DEVICE
```

Effect on System and Foreground.

Format PLC [filecode]

filecode is the print filecode. The default is /01.

Remarks

The PLC command prints all linecodes assigned in all machines. The linecodes are printed on the device assigned to the specified filecode.

Errors

```
INVALID PRINT FILECODE
PRINT FILECODE ASSIGNED TO NO DEVICE
```

Effect on System and Foreground.

Format PLV [filecode]

filecode is the print file filecode. The default value is /01.

Remarks

The PLV command prints all the software levels in use in the whole machine. The levels in use are printed on the device assigned to the specified filecode.

Errors

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

```
    INVALID PRINT FILE CODE
    PRINT F.C. ASSGN TO NO DEVICE
```

Effect on Background, System and Foreground.

Format POF p

p is a page number in the range 0 to 63 (P857/P858) or 255 (P859/P854/P876).

Remarks

The POF command is used by the system manager to inform MAS that a memory page is unavailable for allocation, for instance because a permanent error occurred in the page. Only non-occupied pages can be set off.

Errors

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

```
    PAGE # ERROR
    PAGE # IS NOT NUMERIC
    PAGE # MISSING
    INVALID PAGE #
    PAGE ALREADY USED OR NOT EXISTING
    WHAT IS THE 2ND PARAM?
    DYN AREA OVERFLOW
```

Effect on Background, System and Foreground.

Format PON p

p is a page number in the range 0 to 63 or 255, dpending on the CPU type.

Remarks

The PON command is used by the systems manager to inform MAS that a memory page, previously specified by a POF command in this session, may now be allocated.

Errors

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

```
    NO PAGE OFF
    PAGE # ERROR
    PAGE # IS NOT NUMERIC
    PAGE # MISSING
    INVALID PAGE #
    WHAT IS THE 2ND PARAM?
    PAGE WAS NOT OFF
```

Print Registers of a Program

Effect on	System, Foreground and Background.
Format	PRG program-name,filecode,print-code

program-name	is a valid program name (maximum six ASCII characters).
filecode	is the print file filecode.
print-code	indicates which registers are to be printed: A = all registers (including scheduled label registers); M = main registers only (excluding scheduled label registers); S = scheduled label registers only.

Remarks

The PRG command prints the contents of program registers according to a print-code.

Errors

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

```
    INVALID PROGRAM NAME
    PROG. NAME MISSING
    PROG. NAME TOO LONG
    PROG. NAME UNKNOWN
    INVALID PRINT F.C.
    PRINT F.C. MISSING
    PRINT F.C. NOT ASSIGN
    INVALID PRINT CODE
    PRINT CODE MISSING
    PRINT CODE TOO LONG
    PRINT CODE UNKNOWN
    SYS DYN AREA OVERFLOW
```

Effect on System and Foreground.

Format PRS program-name[,filecode]

program name	is a current program name;
filecode	is the print file filecode. The default is filecode /01.

Remarks

The PRS command prints the status of the specified program on the device assigned to the print file according to the lay-out of one line of the MAP command.

Errors

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

```
    PARAM ERROR
    PROG NAME MISSING
    PROG NAME UNKNOWN
    PRINT F.C. PARAM ERROR (output filecode error)
    PRINT F.C. BAD ASSGN   (output filecode not assigned)
    INVALID PRINT FILE CODE
    PRINT F.C. ASSGN TO NO DEVICE
```

Remove Abort State

Effect on Foreground and Background.

Format RAB program-name

program-name	is any current program name.

Remarks

The RAB command clears the abort state of the specified program.

If a foreground program is aborted it is suspended, in order to allow the user to debug the program by dumping memory/registers. A aborted program can be restarted by the FCL commands RUN or ACTivate, without removing the abort state, but it can only be (re-)activated by the timer or clock, or by another program, after the abort state has been cleared by means of the RAB command.

Errors

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

PARAM ERROR
PROG NAME MISSING
PROG NAME UNKNOWN
PROG INACTIVE
PROG NOT ABORTED
RAB NOT ALLOWED, EVC # 0 (the aborted program has an outstanding event. The abort state can only be cleared when the event is ready).

Effect on System and Foreground.

Format RDV device-address

device-address is a valid device address.

Remarks

The RDV command is used to release an I/O operation. It has the same effect as the RD operator command.

The command may be used after an I/O error has been signalled by the PU operator message (Physical Unit Intervention Required), which terminated with 'RY'.

Errors

One of the following errors will be output if an RDV command is rejected:

```
    PARAM ERROR
    DEVICE ADDRESS MISSING
    DEVICE ADDRESS UNKNOWN
    DEVICE NOT IN RETRY
    DISK DEVICE CANNOT BE RELEASED
```

Effect on — Background.

Format — RST [A7-value]

A7-value — is a value which will be loaded into register A7 before the specified program is restarted. If omitted, zero is loaded by default.

Remarks

When used in a foreground machine the RST command restarts the specified program.

When used in the background machine, the RST command restarts the BCL Processor.

Errors

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

```
    PARAM ERROR
    BAD A7 PARAM VALUE
    PROG NOT IN PAUSE
```

Effect on System and Foreground.

Format RYD device-address

device-address is a valid device address.

Remarks

The RYD command is used to retry an I/O operation. The command may be used after an I/O error has been signalled by the PU operator message (Physical Unit Intervention Required), which terminates with 'RY'. The effect is the same as the RY operator command.

Errors

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

```
    PARAM ERROR
    DEVICE ADDRESS MISSING
    DEVICE ADDRESS UNKNOWN
    DEVICE NOT IN RETRY
```

Effect on System.

Format SCR filecode

filecode	is the filecode to be removed from the filecode table for the system machine.

Remarks

The SCR command causes the filecode table entry to be scratched form memory.

Errors

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

```
    FILE CODE MISSING
    FILE CODE ERROR
    FILE CODE IS NOT NUMERIC
    FILE CODE TOO BIG
    I/O ERROR
    SCR ERROR
    DELETE ERROR
    SPOOLED DVCE
    D:CI IN USE
```

Effect on Foreground.

Format SEG k,x

- k identifies the segment being defined. It is a positive integer not equal to a value given in another SEG command for this machine and not more than the value 'n' in the DCF command for this machine.
- x is the number of 2K pages (minimum 1) in this segment. The maximum is 16 - p, where 'p' is the number of pages specified in the CMA command for this machine.

Remarks

One SEG command must be given for each segment implied by the DCF command. SEG commands must follow the CMA command for the same foreground machine. They can not be given if the machine has no memory-resident segments apart from Segment 0.

Errors

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

```
    SECOND COMMAND MUST BE CMA
    WRONG PARAMETER
    WRONG SEGMENT NUMBER
    SEGMENT NUMBER 0 NOT ALLOWED
    WRONG NUMBER OF PAGES
    # OF PAGES 0 UNALLOWED
    SEGMENT NUMBER ALREADY DEFINED
    NOT ENOUGH FREE PAGES
    # OF 'PAGES FOR ROOT + SEGMENT > 16
```

Effect on System.

Format TIM

Remarks

The time and date will be output to the device with the system filecode /01.

Error

If the timer does not contain a value that can be converted to a valid time, then the error message:

```
    THE DATE IS DESTROYED
```

is output to the device with the system filecode /01.

Effect on System.

Format WRD dc,s,d,v0[,v1]...

dc	is the DAD filecode.
s	is the DAD relative sector number.
d	is the displacement in characters from the beginning of sector 's'.
v0 to vn	is a list of one or more values to be placed in sector 's'. 'v0' is placed 'd' words from the beginning of the sector, 'v1' is placed d+1 words from the beginning, and so on until 'vn' which is placed 'd+n' words from the beginning. In other words the values in the value-list are placed consecutively and contiguously from the address represented by the displacement 'd'.

Remarks

The WRD command is used by the systems manager to alter one or more words on a DAD which is assigned to the system machine. The WRD command cannot access sectors with a length of more than 512 words.

Errors

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

```
    FILE CODE MISSING
    FILE CODE ERROR
    FILE CODE IS NOT NUMERIC
    INVALID FILE CODE
    SECTOR NUMBER ERROR
    SECTOR NUMBER NOT NUMERIC
    SECTOR NUMBER MISSING
    INVALID SECTOR NUMBER
    FILE CODE NOT ASSIGNED
    FILE CODE IS NOT A DAD
    SECTOR NBR OUT OF DAD
    SECTOR SIZE TOO BIG
    DISPL. ERROR
    DISPL. NOT NUMERIC
    DISPL. MISSING
    DISPL. TOO BIG
    READ ERR. STAT=yyyy
    VAL xxxx ERROR
    VAL xxxx NOT NUMERIC
    VAL xxxx DISPL. OUT OF SECTOR
    VAL xxxx TOO BIG
    VALUE MISSING
    WRITE ERROR. STAT= yyyy
```

(xxxx is replaced by the number of the parameter in error.)

Effect on Background, System and Foreground.

Format WRM a,v0[,v1 ... vn]

a	is a five digit hexadecimal number representing the 18- or 20-bitabsolute machine address of the first word to be written. The maximum value is /3FFFF or /7FFFF depending on the CPU type.
v0 to vn	is a list of one or more values to be placed in memory. \`v0′ is placed in memory address \`a′, \`v1′ is placed in memory address \`a+2′, and so on until \`vn′ which is placed in memory address \`a+2n′. In other words, the values in the value-list are placed consecutively and contiguously from memory address \`a′. Each value must not contain more than 4 hexadecimal digits.

Remarks

The WRM command is used by the systems manager to alter one or more words of memory from filecode /E0 of the system machine.

The command cannot be given on the device assigned to the system filecode /E1. The command is similar to the WM operator command.

Errors

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

```
    WRONG PARAMETER
    WRONG LOCATION
    WRONG VALUE
    NO VALUE
```

Note: The format of this command differs from that of the FCL WRM command.

PART 3 THE BACKGROUND MACHINE

GENERAL

MAS can support simultaneously a number of foreground machines and, optionally, one background machine. Once created, the background machine exists until the next IPL, unless an `SCL KIM' command (Kill Machine) is given for it.

BACKGROUND MACHINE PROGRAMS

General Characteristics

Only one background program can be active at a time. Background programs (apart from the BCP) are activated, either by a BCL RUN command or by a BCL Processor-
call command. They are deactivated by an LKM 3 issued by the program (other than in a scheduled label routine).

They may be placed in a Pause state by a SCL PSE command or by a PS operator command (except for the BCP, which may only be placed in a pause state by the BCL PSE command).

Except for the BCP, they may be placed in an Abort state either by MAS or by an AB operator command or by an SCL ABT command or by LKM 46. Unlike a foreground program, this does not cause them to remain in a non-executable state in the machine.

They may enter a Wait state in exactly the same manner as foreground programs.

No Activation queue is maintained for a background program. The active background program is executed whenever it is the executable task with the lowest software level (highest priority) known to MAS.

The background program executes without any communication with the simultaneously active foreground programs. It is not connected to a timer or a clock, and cannot be connected to a level by a foreground program or by a previously active background program.

The background machine may be declared non-swappable or swappable by the system manager, with the SCL DCB command. This will determine whether every background program executed in the background machine is memory-resident or disc-resident. Particularly it allows batch processing and is therefore sometimes referred to as the batch machine.

Memory-Resident Background Programs

If the System manager defines the background machine to be non-swappable (with the SCL DCB command), then all batch programs will be memory-resident. The number of pages specified in the DCB command will be permanently reserved for the background machine, until the next IPL. Background programs are loaded upon activation into these reserved pages, and remain there until the program is deactivated. Operating the Background Machine in this way may reduce the execution time of the Background Machine (because the background programs are never swapped out), but may increase the execution time of the foreground machines (since the foreground machines are not able to use the pages reserved for the background machine).

Disc-Resident Background Programs

Unless the System manager specifically requests a non-swappable background machine, MAS will define the background machine as swappable and will, during the session, allow the background machine to reserve up to 16 pages (32K words) from the same pool of pages that is used for the swappable (declared by FCL RON or SWP commands, or middle-ground) foreground programs. The background machine will be swapped out and in by MAS during its execution. Operating the background machine in this manner may cause the execution time of background programs to increase, but may improve the overall throughput of all the machines, since there will be more memory available for foreground programs. The SIZE parameter on the BCL RUN or Processor-call commands specifies the number of pages which the program assumes it may address, additional to the ones occupied by the load module. When MAS loads or swaps in the program, it will reserve the number of pages required to contain the load module, plus the number of pages of the SIZE parameter.

Programming Techniques

When coding a background machine program, it is not necessary to know whether the system manager will execute the program in a Non-swappable or a Swappable Background Machine. There is no need for a background program to allocate its variables externally, nor for the load module to consist only of constants and instructions. A background program is loaded from the libraries on disc at execution time every time it is activated, so there is no need to reset program variables and accumulators back to their initial values at the end of the program. Background programs may allocate and free dynamic storage using the LKM 4 and 5 Monitor calls. MAS will allocate the dynamic storage from the background machine dynamic area. Although the nature of this area for a swappable background machine differs from that for a memory-resident background machine, the difference is transparent for the programmer providing the SIZE parameter has, if necessary, been used on the BCL RUN command. The LKM 4 and 5 Monitor calls are identical for both types of background machine.

BACKGROUND MACHINE MEMORY ORGANISATION

Background Machine Types

The system manager defines the type of background machine with the SCL DCB command. Two types of background machine are supported by MAS:
- Non-swappable
- Swappable.

Non-Swappable Background Machine

The background machine consists of the number of pages requested in the SCL DCB command. These pages are reserved exclusively for the background machine. All activities in the background machine will use these pages.

The pages may thus contain:
- The BCP (when reading a BCL command)
- A user program activated by a BCL RUN command
- A processor activated by a BCL Processor-call
- Dynamic buffers.

Dynamic buffers are allocated whenever the active program in the background machine issues an LKM 4 Monitor Request. They are allocated by MAS from the pages which were allocated by the DCB command, but which are not being used by the active background program. In other words, the Dynamic Area for a non-swappable background machine consists of the pages reserved for the background machine but not being used by the active background program. This means that the program could in fact simply use virtual addresses to address the dynamic area, without giving any LKM 4. In a swappable background machine this can only be done if the SIZE parameter is given on the BCL RUN command.

Example

Suppose the systems manager defines a 10 page memory-resident background machine thus:

```
    DCB 10
```

If an 11K words program is activated (by a RUN command for example) by a background user, then Pages 0-4 and the first 1K of page 5 contain the program. The second 1K of page 5 and pages 6-9 are the dynamic area for LKM 4 _or_ may be virtually addressed directly by the program.

Swappable Background Machine

If the number of memory-resident pages is not specified on the SCL DCB command, the pages for the background machine are allocated dynamically by MAS from the Dynamic Loading Area shared by the foreground machines. At any moment, the entire background machine is regarded by MAS as a swappable disc-resident program, and its memory is allocated exactly as for a disc-resident foreground program.

Example

Suppose the systems manager defines a swappable background machine thus:

```
    DCB
```

If an 11K words program is activated by a background user by a RUN command with the parameter SIZE=2, then pages 0-4 and the first 1K of page 5 contain the program. The second 1K of page 5 and pages 6-7 are the dynamic area for LKM 4 _or_ may be addressed directly by the program.

DEFINING A BACKGROUND MACHINE WITH SCL

Introduction

The background machine is defined by the Systems Manager using the following SCL commands:

DCB	Define the background machine
FCD	Define a filecode for a machine
MFC	Define the maximum number of spare filecode table entries for a machine
MBF	Define the maximum number of blocking buffers for a machine
LAB	Define the maximum number of scheduled labels for a machine
DEN	End machine definition
LSM	Define a secondary load module for the background machine
KLM	Kill a secondary load module from the background machine.

The last two commands must be given outside the DCB-DEN sequence.

Two other SCL commands (DLN and DLC) are used to define the datacommunications lines used by the background machine.

The system machine stops reading the SCL commands when the BYE command has been read from the SCL file on /E0.

The operator may start the background machine with the Operator Command SB, after the DEN command has been processed or by specifying "BATCH" in the BYE command.

Example of Background Machine Definition

```
DCB 8
FCD 1,TY10
FCD 2,LP07
FCD /E0,TY10
FCD /E1,CR06
FCD /C0
FCD /C1
FCD /C2
FCD /C3
FCD /F0,/C0,SUPERV
FCD /F1,/C1,DAD1
DEN
BYE
```

The SB Operator Command can then be given to start the batch machine, which will cause the first BCL Command to be read from the device assigned by SCL FCD command to the background filecode /E0.

If BATCH is entered as a parameter on the `BYE' command, the background machine is started automatically.
Filecodes /F0, 1, 2 and /E0 mandatory in the background machine.

When, for any physical device, only one has been generated or when the first declared one is meant to be assigned, the device address can be omitted.

Filecodes /C2 and /C3 as they are declared in the example, can only be used, when in the BCP DADs are assigned to these filecodes (via the ASD command or LKM 71).

BACKGROUND MACHINE FILECODES

MAS maintains a filecode table for each machine, which (for user machines - foreground or background) it initialises from SCL FCD commands when the machine is defined. The filecode tables can subsequently be extended or modified by programs which are run in the relevant machine. This is done by LKM Monitor Calls to MAS issued by User programs, or by the command language (FCL or BCL) processors. For the foreground machines this allows unlimited flexibility of task mixes. For the background machine it also has advantages. It is not necessary for every program to have a command defining the files it uses. Files or devices which are used by several programs need only to be assigned once when the background machine is defined, and thereafter every job and program can refer to the file or device merely by specifying the filecode which identifies the relevant entry in the MAS filecode table for the background machine.

Filecodes which are defined by SCL FCD commands remain in the MAS filecode table from the time the SB operator command is entered, to the time the next IPL is made, unless an SCL KIM command is given for the background machine. The BCL :EOB command does not remove them, so there is no need to re-specify them before a subsequent SB operator command is given. New filecodes can be added by Jobs, either by BCL ASG, ASD or REQ commands or by LKM 33, LKM 54, LKM 71 or LKM 23 requests issued by background programs. The same methods can be used to modify the filecodes assigned when the background machine was defined.

Extensions and modifications to the background machine filecode table are cancelled by the BCP at the end of the Job, and the filecode table is restored to the status it had when the background machine was defined.

The BCP and all the Standard Processors have been developed in a way which allows all the devices and files which they use to be defined once and once only by SCL FCD commands. Any temporary filecode assignments which these programs may make are performed by LKM 23 and use the filecodes /D0 to /DF. This makes it very easy to define the background machine and to run the BCP and the Standard Processors.

The filecodes which the Systems Manager must assign with SCL FCD commands entered at background machine definition time are /01, /02, /E0, /F0 and all disc codes required by user Jobs.

The following are the filecodes reserved for the BCP and Standard Processors:

/01	error messages to the background user
/02	printed output
/E0	BCL commands
/E1	ASCII input files
/E2	Binary input files
/D*	Standard Processor temporary filecodes
/ED	Catalogued Procedures work file
/EC	Filecode assigned by BCP to the file B:PROC in the Job USID and DAD
/EE	Catalogued Procedures work file
/F0	This filecode defines the System DAD (whose name, unless modified at SYSGEN, is SUPERV). The first directory in this DAD can be accessed as the USERID SYSTEM, even if another name is has been stored in the catalogue of the DAD. This directory must locate the file B:PROC of catalogued procedures for all the background machine users. Note that the system machine filecode /F0 also refers to the SYSTEM user in the System DAD, in order to locate the Standard Processors.
/C*	Disc drives.
/F*	User DAD filecodes.

Any other user filecodes assigned should obey the following rules (regardless of how the assignment is made - by SCL FCD, by BCL ASG, BCL ASD or by BCL REQ or by LKM 33, LKM 54, LKM 71 or LKM 23):

- Take care, using any of the filecodes listed above.
- Do not use the filecodes: /D0 - /DF, reserved for the Standard Processors and the BCP to assign by LKM.

Filecodes /S, /O and /L

The FCOD parameter occurs on many BCL commands. The background user may, if he wishes, specify the filecodes /S, /O or /L. This facility is provided for compatibility with DOS. The BCP will convert these filecodes as shown below:

FCOD=	CONVERTED
/S	/D4
/O	/D5
/L	/D6.

Background Machine Filecode /E0

This filecode is used for reading BCL commands and Standard Processor commands, in other words, the set of commands which comprise the batch of background machine jobs.

The filecode /E0 must be assigned to a device by a SCL FCD command. If the device is a disc file, tape, card reader or paper tape reader it should not be used by any other filecode in any machine. The batch of BCL and Processor commands will be read as a sequential file.

If the device is a console or display, then whenever the BCP wishes to read the next BCL command, it will output the message:

```
    BCP:
```

and the user must key in the next BCL command. Similarly, the Librarian Processor will output the message:

```
    LIB:
```

when it requires to read the next Librarian Command. The user should then key in the next Librarian Command (or LEN to stop the Librarian).

The Processors ASM, and FRT do not signal when they require to read a control command. They should be keyed in by the user when appropriate. For example:

```
    BCP:ASM
    ASM:OPT PROG=/E0
              IDENT   BB
        HJ    DATA    56
              END
```

If /E0 is a console or display, the device may be shared by other filecodes and files. In the example above, the source program for the ASM Processor has been keyed in on /E0.

Background Machine ERR Device

The ERR device is used to report all rejected BCL commands, and to request that the user resubmits the corrected command. If a command is rejected, 3 lines are output on the ERR device. These are the command, the error message and the program identifier. For example, if a BCL :LIB Processor Call has an invalid DUMP parameter, the BCP will output on the ERR device:

```
    LIB DUMP=BLL
     INVALID PARAM=DUMP
    BCP:
```

and the user should re-enter the LIB command on the 3rd line with the corrected DUMP parameter.

Under some circumstances, the first line will not be output. This occurs when it is already the most recent line on the device. For example, if the BCP device (filecode /E0) is assigned to a console or display and the ERR device is assigned to the same device as /E0, then the invalid BCL command will have been keyed in by the user and thus already be displayed. There is no need for line 1 to be output in this case. The same applies if filecodes /02 and the ERR device are assigned to the same device, since all commands are copied to /02.

The ERR device may be shared by several filecodes in several machines, if the other filecodes allow this.

The default error correction device is /01. which specification is mandatory during machine definition.

Background Machine Filecode /02

This device is used by the BCP to output a copy of all the commands that have been read from /E0. If a command is rejected, the error message sent to the ERR device and the respecified command subsequently entered from the ERR device will also be output on /02. The date and time is also output whenever a job starts or ends.

If /02 and ERR are assigned to the same device, the error message and the re-specified command are not output to /02, since they are already present.

If /E0 is assigned to a console or display and /E0 and /02 are assigned to the same device, the input to /E0 is not output to /02.

The device assigned to background machine filecode /02 may be assigned to several filecodes in several machines if it is allowed to share these other filecode devices.

Background machine filecode /02 is also used for all "printed" output from the Standard Processors.

Background Machine Filecode /01

When no ERR command has been given, error messages are output and can be corrected via filecode /01, which has to be declared during machine definition.

ACCOUNTING ROUTINES

Functions

Accounting routines may be written by the user to perform the following functions for the background machine:

- To verify that the account number specified for each background machine job is correct and to stop the job from being executed if it is not.
- To verify that the password specified for each background machine job is correct and to stop the job from being executed if it is not.
- To output information when each background machine job ends.

Naming Restriction

Two accounting routines may be written, one called USRJOB and the other (which is not required if passwords and/or account numbers are to be verified, but no information is to be output) called USREOJ.

BCP Interface

USRJOB is called by the BCP when a :JOB card is read.

USREOJ is called by the BCP:

- When a BCL :EOJ command is read.
- When a BCL :JOB command is read and it was not the first BCL command read in this batch, and neither was it preceded by a BCL :EOJ command. This means the previous job has ended without an :EOJ command. In this case the BCP calls USREOJ first, and upon return from that routine, calls USRJOB.
- When a BCL :EOB command is read which was not preceded by a BCL :EOJ command, the BCP calls USREOJ.

USRJOB

The routine may examine the account number and password on the BCL :JOB command, by declaring two 8 character external areas ACNT and PASW respectively. USRJOB should not alter the contents of any register except A7, which is set to 0 if BCP is to accept the job (i.e. read the next BCL command and process it), or a non-zero value if BCP is to reject the job (i.e. ignore all subsequent BCL commands until the next :EOJ, :EOB, :JOB command). Having set A7, the user accounting routine USRJOB must return to BCP with an RTN A14 instruction.

USREOJ

This routine ensures, as USRJOB, that all the registers it modifies are restored to their value upon entry. It returns to BCP with an RTN A14 instruction.

Standard Accounting Information

The BCP outputs accounting information for each background machine Job to the background filecode /02.

- the date and time is output to /02 after every :JOB, :EOJ, :EOB, RUN and Processor Call command read by the BCP from /E0.
```
        DATE: MM DD YY // TIME: **H.**M.**S
```
- the elapsed time, and the number of records printed and punched, is output to /02 at the end of performing each RUN command.
```
        ELAPSED TIME=****,PRINTED RECORDS=****,
        PUNCHED RECORDS=****
```
- the exit code (value in A7 at LKM 3) is printed after each RUN and Processor Call command.
```
        EXIT /**
```

DEBUGGING NOTES

When a dump of a program occurs, it will be noted that the LM is preceded by some MAS control tables. Some of the information in these control tables may be useful for debugging purposes.

The contents of these control tables is shown below. Note that word 0 refers to the first word shown in the Dump listing.

WORD	CONTENTS
0	program start address (+1 if the program is segmented into overlays).
1	the number of LM disc sectors containing the root or only segment.
2	program length, including the blank common area if it is not absolute.
3	address of the debug symbol table generated according to the DBUG keyword parameter of the LKE OPT command.

If the program is not an overlay program, word 4 onwards contains the LM.Otherwise, the contents from word 4 are as described below:

WORD	CONTENTS
4	the amount of memory required to execute the program. This is the length of the longest path plus the amount of memory required for the Segment loaded and its control tables.
5	the number of overlay segments in this LM. The root segment is not an overlay segment.

Each overlay segment has a 4-word entry. The entries occur in the same order as the overlay segments appear in the overlay tree (in other words, in ascending segment level within ascending lowest path for this segment). The first 4-word entry begins in word 6 of the dump.

Each 4-word entry has the format:

WORD	CONTENTS
0	<u>bit 0</u> is 1 if this segment is loaded in memory. <u>bits 1-15</u> contain the number of the 4-word entry for the ascendent segment to this segment (0 if the root segment is the ascendent segment).
1	the address where the segment has been or is to be loaded.
2	the relative sector number in the disc LM file from which the segment has been or is to be loaded.
3	the length of the segment.

The last 4-word entry is followed by the segment loader, which in turn is followed by the root segment of the LM.

By examining bit 0 of all 4-word entries, one can determine which path of an overlay program is loaded and was executing when the dump occurred.

BACKGROUND MACHINE OPERATION

Background Machine Activities

The background machine is used to execute the following programs:

- BCP Background Control Processor
- Standard Processors
- Non-standard Processors
- User Batch Programs
- Accounting routine.

The activities to be performed in the background machine are specified with Background Command Language (BCL) Commands which are processed by the BCP.

THE BACKGROUND CONTROL PROCESSOR (BCP)

Method of Operation

The BCP is loaded into the Background Machine when the SB Operator Command is given. The BCP will read each BCL Command from the device /E0 and perform the required operation. If the BCL Command specifies a Processor to be executed, or a user program to be run, this program will be loaded into the background machine, overwriting the BCP. When the program has ended, the BCP will be reloaded to process the next BCL command. When the BCP has completed the processing of a BCL command, a return code is set. This return code may be set either by the BCP or by the program which was executed by the previous BCL command. The return code has a value from /0 to /7F.

The return code should be set by user batch programs (in A7 when they exit with LKM 3), using the same conventions as the BCP and the Processors, namely:

/0	No error
/01-2F	Minor errors
/30-3F	Errors which forbid program execution
/40-4F	Errors which forbid Linkage Edit e.g. error in compilation; no object deck has been produced.
/50-5F	Unused
/60-6F	Errors which forbid FORTRAN or Assembly Processor execution, e.g. error in source update.
/70-7F	Errors which forbid execution of the JOB, e.g. disc overflow, no dynamic buffer available.

Operating Modes

The BCP can be run in two modes: Closed-shop and Interactive.

The Closed-shop mode is used if the user responsible for the Job will not be present in the computer room when the Job is executing, and does not want the operator or anyone else to try to correct any errors that occur.

The Interactive mode is used if the user will be present at a terminal when the Job is run, and can thus interactively correct any errors which may occur.

Closed Shop

In this mode, BCL commands are grouped into steps. A maximum return code is specified for each. If any BCL command causes a return code to be issued which exceeds this maximum value, the remaining BCL commands in the step are ignored. BCL :STP commands are used to specify the closed-shop mode, to define the steps and to set the maximum permitted return code for each command in the step.

Interactive

In this mode, if a BCL command is found invalid, the BCP will allow the background machine to correct the command from a specified device. After the command has been corrected, BCP will process it and then resume reading the BCL from the device /E0.

The BCL ERR command is used to specify the interactive mode and the correction device.

If both the ERR and one or more :STP commands are used in one Job, then correcting commands from the ERR device is only possible for commands which ended with a return code less than the current Step value.

BCL - BACKGROUND COMMAND LANGUAGE

Tasks and Programs

The basic unit of work for the background machine is a task. A task is performed by a program. For example, the LIBRARIAN may be used to copy a sequential file of 20 records. In this case the task is to perform 20 times the sequence of instructions which copy a record from one file to another. If this sequence is, say, 10 instructions, then the task consists of 200 instructions. The program which performs this task consists of over 1000 instructions, only 10 of which are relevant to this task.

A background task may not activate other tasks, but it may activate subtasks by using scheduled labels.

The difference between a task and a sub-task is that the former is not synchronised with the activating task, whereas a subtask is.

Foreground tasks can activate other foreground tasks by LKM 10 or LKM 12, and then continue to execute and even exit without any regard as to whether these other foreground tasks have been performed.

Background tasks cannot do this.

Foreground tasks can create subtasks either by LKM 12 with an associated Wait or by scheduled labels. Only scheduled labels may be used by background tasks to create subtasks.

In the background machine, there is at any one moment only one active task. The active task will be performed either by the BCP, or by a Processor (if the last BCL command was a Processor Call command), or by a user program (if the last BCL command was a BCL RUN command).

Only one program may be active in the background machine at a time, and it may only be activated once. Background programs have no activation queue.

Batches, Jobs and Steps

A batch is a set of BCL commands, the last of which is :EOB. For each batch the operator must give the SB command to load the BCP to process it.

The :EOB command deactivates the BCP and thus frees the devices and memory it was using.

Within a Batch, there can be one or more Jobs. A Job is a series of BCL commands from one user. Each user wishing to make use of the background machine during the current session should submit a separate Job. A user may submit more than one Job.
Each Job may consist of one or more Steps. A Step is a sequence of dependent BCL commands. Thus, if a BCL command is not performed successfully due to some error situation, all subsequent BCL commands in the Step should be ignored. For example, a Step might consist of an Assembly followed by Linkage Edit and a Run of a user program. If the Assembly ends abnormally, Linkage Edit and Run should be bypassed.

The hierarchy of operations is as follows:

SEVERAL BATCHES PER SESSION
SEVERAL USERS PER BATCH
1 USER PER JOB
SEVERAL JOBS PER USER
SEVERAL STEPS PER JOB
SEVERAL PROGRAMS PER STEP

BCL Functions

BCL performs the following functions:
- defines Batches, Users, Jobs, Steps and Programs.
- assigns filecodes used by the background program to devices, files or DADs.
- allows messages to be sent to the operator when a JOB reaches a certain stage.
- controls the loading of programs in the background machine. (For overlay batch programs, the BCP only loads the root segment. The overlay segments are loaded automatically during the execution of the task. This is done by LKM 27 inserted into the segment loader of the root segment by the Overlay Linkage Editor).

BCL Scheduling Commands

Recommendation

Four BCL commands are used for job scheduling in the background machine. These are the :JOB, :STP, :EOJ and :EOB commands.

MAS will abort any program which tries to read a record commencing with one of the strings:

```
     :JOB  :STP  :EOJ  :EOB
```

unless the program is the BCP. Placing the command names for the four BCL scheduling commands in the first record position will therefore ensure that only the BCP may read them. (The abort code is /10 if a program tries to read a BCL scheduling command.)

The BCP will, however, accept these BCL commands with the command name anywhere in the record. See the :STP command description for an exception.

If a user background program input filecode is assigned to the same device as the BCL filecode /E0, and if when this user background program is run it fails to detect the end of its input file (because of a missing :EOF record, for example), then it may erroneously read BCL commands for the next JOB if the :JOB command name for the next JOB is not placed in record position 1.

Job Scheduling Commands

These commands control the initiation and termination of job streams and determine the mode of operation (closed-shop or interactive).

:EOB **End of BCL** :EOB

Format :EOB

If the previous command was not :EOJ the BCP will call the program USREOJ (which is either the dummy system supplied program, or a User accounting routine). The BCP will then issue an Exit (LKM 3) to MAS. The background machine, although still defined, will then be closed until the operator enters an SB operator command. No BCL commands which occur after the :EOB command will be read by the background machine until the SB operator command is given by the operator.
The BCL reacts with the message:

```
     END OF BATCH
```

Format :EOJ

The BCP calls the program USREOJ, which is either the dummy program supplied or a User accounting routine.

If the command is placed in positions 1-4 of the BCL record (as is strongly recommended), it will ensure that under no circmstances can the next Job's BCL commands be erroneously processed as input data for this Job.

The BCP will read all subsequent records from background machine filecode /E0 and ignore them until it reads a :JOB or an :EOB command.

Format 1 ERR [FCOD=r]
Format 2 ERR [r]

r is the recovery filecode, an optional parameter specifying the filecode from which the BCP will read corrections when the BCL command being analysed is invalid. The filecode specified must have been assigned to a console or display device type. If the parameter is omitted, filecode /01 of the background machine is assumed.

The ERR command is used to put the background machine into interactive mode. If a BCL command is found to have invalid syntax, the BCP will inform the background machine user by sending an error message to the device assigned to the recovery filecode. From the same device the user may submit the corrected BCL command. After this has been processed, the BCP will resume reading BCL from the device /E0. This facility is only available for correcting BCL syntax errors. It canot be used to correct errors which occur during processing a valid BCL command. For example, if the PRNT parameter of the BCL RUN command is specified as PRNT=A4, then it will be rejected since the parameter value is not numeric, and can be respecified by the background machine user from the recovery filecode device. If, however, the parameter had been specified as PRNT=50 but the program to be RUN requires to output 70 lines of print, then the program will be run until line 51 is to be output, but the user cannot then alter the PRNT value from the recovery filecode device.

The following error message may be output by the BCP if the ERR command is incorrect:

INV. FCOD
The filecode is not assigned, or is not assigned to a console or display device type.

If no ERR command has been supplied in a Job when a BCL syntax error occurs, all error messages are output and can be corrected via filecode /01.

Format 1	:JOB [USID=u[,DAD=d]][,ACNT=a][,PASW=p][,VOLN=v,DNAM=dn]
FORMAT 2	:JOB [u],[d],[a],[p],[v],[dn]

u	is a userid, a string of up to 8 ASCII characters, which specifies the name of the user directory. This name must already exist. Default value is the first userid on the DAD with filecode /F0. The same userid is taken, when USID=SYSTEM is specified, but then some priviliged system manager commands can be issued (like LIB commands DCD, DLD, DCU). The specified userid is default during the whole Job for every command. It can be overridden by a USID parameter specified in a BCL command in the Job. If the parameters DAD, VOLN or DNAM are given, the USID specification is obligatory. If DAD specification parameters are omitted, all assigned DADs are searched to find the specified userid, in the sequence of the declaration of the DADs in the machine declaration.
d	is the DAD filecode (/F0-/FF). It specifies if the DNAM and VOLN parameters are omitted, the DAD where the specified userid is located. When VOLN and DNAM are specified, it gives the filecode, which should be assigned to the DAD where the specified userid is to be found.
a	is an accounting number. A string of up to 8 characters, which is not used by the BCP or any other component of MAS, but may be used by a user accounting routine (if any).
p	is a password. This string of up to 8 characters can only be used in a user accounting routine.
v	is a volume number. It specifies, together with the DNAM parameter, a DAD to be temporarily assigned for this Job. The volume number consists of 4 hexadecimal characters.
dn	is a DAD name. It specifies the name of the DAD where the specified userid is located. This DAD is assigned to the filecode given in the DAD parameter or, if the DAD parameter is omitted to the first free DAD filecode, from /F1 to /FF.

The :JOB command must be the first BCL command for each background machine Job. It must thus also be the first BCL command read from the filecode /E0 when the background machine is started with the SB operator command.
When a :JOB command is given in a session without a previous :EOJ command, the BCP first generates a :EOJ command and then processes the :JOB command.
The :JOB command causes the BCP to reinitialise the background machine control tables (to the values set up by MAS when the background machine was defined by the SCL DCB command, etc.), and to locate the user directory from the parameters USID and DAD on the :JOB command.

If the ACNT and PASW parameters are specified, the BCP will pass these to the routines USREOJ (unless it is the first :JOB command in the batch, or if the previous command was :EOJ) and USRJOB. If the dummy system-supplied USRJOB program has been replaced by a User accounting routine, then this routine can, upon return to the BCP, signal whether the Job is to be processed or bypassed. In the latter case, the BCP will ignore all remaining BCL commands until the next :JOB or :EOB command. The reinitialising of the background machine filecode tables whenever a :JOB command is read is necessary, since the previous Job may have modified them by:

- The BCL ERR and :STP commands set the background machine operation mode.
- The BCL ASG and ASD commands can override the filecodes defined at DCB time.

The following error messages may be output by the BCP if the :JOB command is incorrect:

```
    PARAM xxxxx MISSING
    GIVE A DAD FILECODE
```
No free DAD filecode could be found to assign a DAD dynamically. The user should specify a filecode in the DAD parameter, overridding the original filecode given during the SCL DCB session.
```
    WRONG VOLUME NUMBER
```
The VOLN value consists of more than 4 characters, or a disc with the specified volume number has not been mounted.
```
    INVALID PARAM=xxxx
    BAD ASSIGN STATUS=xxxx
```
The LKM 71 (assign DAD) returned an error status. See Appendix C for the meaning of the status.
```
    I/O ERROR ON DISC
```
An I/O error occurred during the DAD assignment (LKM 71)
```
    SYSTEM DYNAMIC AREA OVERFLOW
```
No space could be obtained, to locate the tables for the DAD to be assigned.
```
    DAD NAME NOT FOUND
    /FO CANNOT BE REASSIGNED
```
DNAM and VOLN were given with a DAD parameter value /FO
```
    TOO MANY FILECODES
    ERROR READ DAD, STATUS=xxxx
```
An I/O error occurred, reading the Catalog of the DAD to be assigned..
```
    PARAM USID MISSING
    USID NAME UNKNOWN
```

Error messages output by the command analyser can be found at the end of this chapter.
Note: The semicolon (:) must be the first character on the JOB command and must not occur again within the body of the command.

Format 1   :STP [CODE=n][,ABCD=c][,NCOD=p]
Format 2   :STP [n],[c],[p]

n is a numeric value from 0 to /7F, defining the highest BCL error code allowed in this Step. If omitted, 0 is assumed. The n must be higher than the BCL error code from the preceeding step.

c is a numeric value from 0 to /7F, defining the value that BCP is to assign to the error code if any program executed in the background machine between the occurrence of this :STP command and the next one (or the next :EOB, :EOJ or :JOB command) is aborted. If omitted, aborts cause the error code to be set to /7F.

p is a numeric value from 0 t0 /7F, defining the new highest BCL error allowed in this step. With the NCOD parameter, the current BCL error code can be decreased. When the current error code is lower than the NCOD value, the NCOD is ignored.

The :STP command places the background machine into closed-shop mode, in which a Job consists of one or more Steps, each starting with a :STP command. BCL commands occurring before the first :STP command in a Job are regarded by the BCP as being in a Step having a maximum error code value of 0.

During processing, if a BCL command results in the error code being set to a value exceeding the maximum error code value for the Step, all subsequent BCL commands for the Step will be ignored.

A Step is terminated either by another :STP command, or by any of the following BCL commands: :JOB, :EOJ or :EOB.

The following error message may be output by the BCP if the :STP command is invalid:

```
    INVALID PARAM= xxxx
```

Note: The :STP command must start in column 1, otherwise it may be read as data by the processor or by a program.

BCL Filecode Allocation Commands

Default allocation

Permanent filecodes for the background machine are allocated by SCL FCD commands, entered by the system manager when defining the background machine and prior to the operator starting it with the SB operator command. These assignments can, however, be modified by background machine Jobs. Such modifications only have effect for the Job in which they are made, since the BCP resets the background machine device assignments back to their machine definition values whenever a new Job is started.

Whenever a :JOB command is read by the BCP, the filecode table is initialised to contain only the filecodes declared at machine declaration time.

A Job modifies its job filecode from the default background machine device assignments by:

BCL ASD commands
BCL ASG commands
BCL SCR commands
BCL REQ commands
BCL REL commands
LKM 23 monitor calls
LKM 24 monitor calls
LKM 33 monitor calls
LKM 54 monitor calls
LKM 71 monitor calls.

Only modifications by BCL commands are considered in this section. LKM (Link to Monitor) requests are dealt with in Appendix C.

Note:
A permanent filecode cannot be re-assigned to be equivalent to a temporary filecode.

Format 1 ASD FCOD=f,VOLN=v,DNAM=d
Format 2 ASD FCOD=f,DISK=c,DNAM=d
Format 3 ASD f,{c|v},d

f is a DAD filecode from /F1 to /FF. This filecode is assigned to the DAD specified by the VOLN or DISK and DNAM parameters. f may be an existing DAD filecode, but not /F0.
v is the volume number of the disk where the DAD to be assigned resides. It must be up to 4 hexadecimal charaters.
c is the filecode of the disk (from /C0 to /CF) where the DAD to be assigned resides.
d is a DAD-name, up to 6 characters. It is the name of te DAD to be assigned.

With the ASD command, the user may assign a DAD, not yet declared during machine declaration. The filecode to be assigned may be free or occupied, but an already assigned filecode cannot be reassigned if DFM or TDFM files are assigned to it. Also the Job DAD filecode cannot be reassigned.
Although the disk can be specified via the volume number, its disc filecode (/C0-/CF) must be contained in the filecode table of the Background machine.
The temporary DAD filecodes are scratched or reset to their original assignments on detection of a :JOB, :EOJ or :EOB command.

The following error messages may be output due to an errorneous command input, or to a status returned from the executed LKM 71 (assign DAD):

    I/O ERROR ON DISC
Error detected during read of the DAD's catalogue
    SYSTEM DYNAMIC AREA OVERFLOW
No space to allocate the tables for the DAD to be assigned.
    DAD NAME NOT FOUND
The DAD does not exist on the disc indicated by the volume number or the disc filecode
    WRONG VOLUME NUMBER
VOLN parameter value contains more than 4 characters or is not found.
    DISC F.C. OR DAD F.C. MUST BE /Cx OR /Fx
    DISC F.C. NOT ASSIGNED
The filecode specified for the disc has not been specified during machine definition.
    /F0 OR DAD OF THE CURRENT JOB CANNOT BE REASSIGNED
    TOO MANY FILECODES
    FCOD VALUE NOT ALLOWED
The FCOD parameter value is not a DAD filecode.
    GIVE DISC VALUE FROM /C0 TO /CF
    VOLN OR DISC PARAMETER MISSING
    VOLN OR DISC PARAMETER REDUNDANT
    BAD ASSIGN STATUS=xxxx
The LKM 71 returned an error status. For the meaning of that status see Appendix C.
    DAD F.C. CANNOT BE REASSIGNED, A DFM OR TDFM FILE IS ASSIGNED TO IT
    INVALID PARAM=xxxx

There are four types of ASG command which may be used in the BCL.

Assign a filecode to a physical non-disc device.

```
Format 1a   ASG  FCOD=f,DVCE=n[a[l]]
Format 1b   ASG  f,n[a[l]]
```

- f is a filecode to be assigned
- n is the device type (a 2 character code); see Appendix A.
- a is the device address (two hexadecimal digits without a preceding /). If omitted, the first device in the SYSGEN list for the device is assumed. This parameter should be omitted if n = NO.
- l For devices connected to an AMA8 controller, a linenumber from 0 to 7 can be specified. There are two ways to indicate an AMA8 linenumber: The linenumber can be added to the device address, so a+l ,or it can be specified separatily.
  Example: an assign of a display connected to an AMA8, device address /18 linenumber 1 can be done as: ASG 2,DY19 or
  ASG 2,DY181.

Assign a filecode to a disc temporary file

```
Format 2a   ASG FCOD=f[,DAD=d][,TYPE=t][,NBGR=n][,CONS={YES | NO}]
Format 2b   ASGT f,[d],[t],[n],[YES|NO]
```

- f is the filecode to be assigned.
- d is a DAD filecode (default is the Job DAD).
- t is the filetype code (default = UF).
- n is the number of granules (default = 1).
- CONS specifies whether the granules are consecutive (YES) or not (NO). The default is NO.

Note: If the file is to be created by Direct Write LKM 1, the number of granules (consecutive or non-consecutive) must be allocated at Assign time.

For example, if a file has 8 sectors per granule, and only 1 granule is allocated at Assign time, then if a Direct LKM 1 is given to write sector 23, it will be rejected with status /10.

Assign a filecode to a catalogued file

```
Format 3a
            ASG FCOD=f,FNAM=n[,DAD=d][,TYPE=t][,USID=u][,WPRO={YES | NO}][,VERS=v]
Format 3b
            ASGF f,n,[t],[u],[d],[YES|NO],[v]
```

- f is the filecode to be assigned
- d is the DAD filecode whose catalogue contains the filename (default is the Job DAD).
- n is the filename.
- t is the filetype code (default = UF). One of the codes SC, OB, LM, UF and EF.
- u is the userid of the Directory (default = :JOB Userid).
- v is the version number (a digit from 0 to 7, but not more than the value on the SMV command to LIB specified for this Userid). The default value is 0.

WPRO is used to set (YES) or leave alone (NO) the write protect flag for this file in the MAS tables. The default is NO. The flag is not updated in the directory on disc - this can only be done by the Processor LIB with SPF or RPF control commands - but only in the MAS memory tables. Specifying WPRO=YES is only required if the write protect flag was not set ON when the file was created; nevertheless this background machine job requires the file to be protected from being written to by any foreground machine task until the background machine Job is finished.

Assign a filecode to another filecode

Format 4a ASG FCOD=f,ECOD=e
Format 4b ASGE f,e

f	is the filecode to be assigned.
e	is the filecode which has been assigned.

Note: filecode \`e′ must not itself have been assigned to another filecode. This is to prevent problems if filecode \`e′ is scratched.

Filecode ′f′ may be a permanent filecode. If filecode \`e′ is subsequently reassigned, filecode \`f′ is also reassigned.

The following error messages may be output by the BCP if the ASG command is rejected:

```
    PARAM xxxx MISSING
    INVALID PARAM=xxxx
    BAD ASSIGN STATUS=xxxx
```
The LKM 23 (assign) which is used for the execution of this command, returned a status. For an explanation of the status see Appendix C.
```
    DEVICE NAME UNKNOWN
    I/O ERROR ON DISC
```
I/O error, reading the Catalog of the DAD, the directory of the userid or the GRANTB of the file to be assigned.
```
    SYSTEM DYNAMIC AREA OVERFLOW
```
No space to allocate the tables to assign the filecode
```
    NO SPACE FOR FILE DESCRIPTION TABLE
    DEVICE OR DAD UNKNOWN
    NO SPACE ON DISC
```
There are not enough free granules in the DAD to assign the temporary file
```
    UNKNOWN FILE NAME
```
The catalogued file to be assigned does not exist in the directory of the specified userid.
```
    ECOD NOT ASSIGNED
    INVALID ASSIGN TYPE
```
The specified ASG parameters caused the BCP to construct an illegal control block for the LKM 23.
```
    USID IS UNKNOWN
```
The specified userid did not exist in the DAD Catalog
```
    ILLEGAL FILE TYPE
```
The file type was not UF, SC, LM, OB or EF.
```
    INVALID FILE CODE
```
A DAD filecode was not /F0-/FF or a filecode was not /00-/FF.
```
    DAD FCOD NOT ASSIGNED
    PERMANENT FILECODE CANNOT BE REASSIGNED BY EQUIVALENCE
    BAD ASSIGN BLOCK ADDRESS
```
This message can never be caused by the user. It is a Hardware or BCP error.
```
    TOO MANY FILECODES
    BAD DEVICE ADDRESS
    UNSHARED FILE
```

The catalogued file that was to be assigned, was not shareable and the user, trying to assign it was not the owner, nor SYSTEM.

TOO MANY GRANULES FOR THE SECTOR SIZE

The NBGR parameter value was more than (DAD sectorlength-10)/2.

BAD LINENUMBER FOR AMA8

In the Format 1 assign, the l parameter was not 0-7.

Additional error messages for the assign of TDFM files:

USER IDENT UNKNOWN FOR ONE SUBFILE
NAME UNKNOWN FOR ONE SUBFILE
DYNAMIC AREA OVERFLOW IN SYSTEM MACHINE
ONE PACK NOT ON LINE
THE DAD OF ONE SUBFILE IS UNKNOWN
THE DAD OF ONE SUBFILE IS NOT ASSIGNED
THE FILECODE OF THE DAD OF THE DESCRIPTOR IS UNKNOWN
FILE ALREADY ASG. IN OTHER MACH. WITH NON IDENTICAL DAD FILECODE
NO FCT ENTRY FOR DISC OF DESCRIPTOR
NO FCT ENTRY FOR DAD OF DESCRIPTOR
FILE CODE WAS USED BY RUNNING TRANSAC. FOR ANOTHER EDFM FILE.

Assigning TDFM files, in the ASG command, only the identification of the descriptor file has to be given. In the descriptor file MAS finds references to KEY and DATA files.Most of the above mentioned error messages are caused by the fact that these references call discs (via the volume number), DADs or userids, which are not known or assigned in the Background machine.

NOTE:

If the syntax of the assign command is incorrect, the specified f can be deleted.

Format 1 SCR FCOD=f
Format 2 SCR f

f is the filecode to be removed from the MAS tables.

The SCR command is used to remove a filecode from the MAS filecode table for the current background machine Job. It is only necessary to do this if the current Job is likely to cause an overflow of the filecode table, because it has a large number of filecode assigns, and will exceed the SCL MFC limit, or the system generation default.
It is also used if a new object file is to be created (processors only extend it) or if filecodes have been assigned to a DAD that has to be reassigned. Certain filecodes used by the BCP to process the BCL commands for the Job (for example filecodes /E0, /02) cannot be specified in the SCR command.

If a magnetic tape or cassette was assigned by a BCL REQ command, it should be scratched by a BCL REL command in order to:
- Scratch the filecode
- Tell the operator to dismount the tape
- Release the device attached by the REQ command.

LKM 54 may be used instead of the REQ and REL commands, if preferred.

The following error messages may be output by the BCP if the SCR command is rejected:

```
FILE CODE NOT ALLOWED
FILE CODE NOT FOUND
COM. EXECUTED, BUT F.C. WAS USED BY STILL RUNNING TRANSAC FOR A TDFM FILE
```

BCL Tape and Cassette Commands

The background user may use these devices by performing the following sequence of commands:

- Give a BCL REQ command (to attach the device, assign a filecode to it, and instruct the operator to mount it).
- Give commands to position the tape or cassette, and to label it as required. Commands to either the BCP or LIB may be used; they have the same format.
- Run the program which uses the tape (via LKM 1 or LIB).
- Give commands to the BCP or to LIB, to write EOV or EOF and to reposition the tape as necessary, these can also be done via LKM 1.
- Give a BCL REL command to detach the device, scratch the filecode, and instruct the operator to dismount the tape.

<u>Format 1</u> FBS FCOD=fc[,NUMB=n]
<u>Format 2</u> FBS fc[,n]

<u>fc</u> is a filecode assigned to an MT or TK device
<u>n</u> is the number of tape marks to be back-spaced.

The tape will be positioned before the <u>n</u>th tape mark skipped. If the NUMB parameter is omitted, NUMB = 1 is assumed.

One of the following error messages is output if an FBS command is rejected:-

```
NO TAPE MARK ON M.T. (begin of tape is found)
I/O ERR ON M.T. (FCOD=xy), STATUS=abcd (see Appendix C, LKM 1)
```

Format 1 FFS FCOD=fc[,NUMB=(ALL | n}]
Format 2 FFS fc[,ALL|n]

fc	is a filecode assigned to an MT or TK device.
n	is the number of subsequent tape marks to be skipped over. The tape will be positioned after this tape mark.
ALL	The tape will be forward-spaced until two consecutive tape marks arefound and positioned after the first of these two.

If the NUMB parameter is omitted, NUMB=1 is assumed.

One of the following error messages is output if an FFS command is rejected:

```
PARAM ERR
NO TAPE MARK ON THE TAPE
KEY PARAM TOO LONG
I/O ERR ON M.T. (FCOD=xy), STATUS=abcd
```

Format 1 PLB FCOD=fc Format 2 PLB fc

fc is a filecode assigned to an MT or TK device.

The volume label of the MT or TK will be printed on the operator console and the tape will be positioned at the first record of the file following the label. If the tape has no volume label, the message:

```
    NO VOLUME LABEL (first record does not start with 'VOL1')
```

is output, and the tape is backspaced one block.

The following error message is output if a PLB command is rejected:

```
    I/O ERROR ON M.T. (FCOD=xy), STATUS=abcd
```

Format 1 RBS FCOD=fc[,NUMB=n]
Format 2 RBS fc[,n]

fc is a filecode assigned to an MT or TK device.
n is the number of records to be skipped.

The tape will be positioned before the nth previous record. If the NUMB parameter is omitted, NUMB = 1 is assumed.

One of the following error messages is output if an RBS command is rejected:

```
BEGINNING OF TAPE IS ENCOUNTERED
EOF ENCOUNTERED
I/O ERR ON M.T. (FCOD=xy), STATUS=abcd
EOS ENCOUNTERED
```

Rewind File

Format 1 REF FCOD=fc
Format 2 REF fc

fc is the filecode.

The file is rewound and positioned at the first record.

The following error message may be output:

```
**** I/O ERROR. (FCOD=xy), STATUS abcd
```

Release a Device

Format 1	REL FCOD=fc,MESS='m'
Format 2	REL fc,'m'

fc	is a background machine filecode mentioned on a previous REQ command.
m	is a message to be sent to the operator on filecode /EF of the system machine, maximal 72 charcaters.

The system will detach the device and send the operator the message:

```
    DISMOUNT dn da m ,PLEASE
```

where

```
    dn = device name code
    da = device address
    m  = message on REL command.
```

One of the following error messages will be sent to the background machine user on filecode /01 of the background machine, if a REL command is rejected:

```
    DISK DEVICE OR DEVICE NOT ATTACHED TO THE PROGRAM
    FILE CODE NOT ASSIGNED
    NOT ASSIGNED TO A PHYS. DEV. OR ASSIGN TO NO DEVICE
```

Format 1 REQ DVCE=d,FCOD=fc,MESS='m'
Format 2 REQ d,fc,'m'

d	is a two-character device-name code, either MT or TK. An operational device of this type will be attached (reserved exclusively) to the Job.
fc	is the filecode to be assigned to the device; two hexadecimal characters, preceded by the / character.
m	is a message to be sent to the operator on filecode /EF of the system machine, maximal 72 characters.

The REQ command is used by the background user to perform the same functions as LKM 14 (to request the exclusive use of a device), or LKM 54 (REQ or REL).

The background user may not know the device addresses, and even if he does he will not necessarily know which devices are being used by the foreground machines at any moment. The ASG command cannot in these cases be used by the background user, and the REQ command is available for such circumstances.

The system will:
- Allocate an operational device (one not mentioned on an 'OF' operator command or an SCL DOF command from the system manager) of the requested type to the batch machine.
- Attach it to the Job.
- Assign the specified filecode to it.
- Send the message:
  MOUNT ON dn da 'm' THEN RS, PLEASE
  to the operator on filecode /EF of the system machine, where:
  dn = device type on REQ command
  m = message on REQ command
  da = the device address allocated.
- Enter the pause state until the operator enters the RS (Restart) command for the background machine.

One of the following error messages will be output to the background user on filecode /01 of the background machine, if a REQ command is rejected:

```
NOT ASSIGNED
DYNAMIC AREA OVERFLOW
INVALID DEVICE NAME (ONLY MT OR TK)
NO FREE AND OPERABLE DEVICE
BAD ASSIGN STATUS = abcd
```

Format 1 REW FCOD=fc
Format 2 REW fc

fc is a filecode assigned to an MT or TK device.

The tape is rewound to the load point.

One of the following error messages is output if a REW command is rejected:

```
    FILE CODE MISSING
    FILE CODE ERROR
    FILE CODE UNKNOWN
    I/O ERR ON M.T. (FCOD=xy), STATUS=abcd
    ERR ENCOUNTERED IN REWIND (the LKM 1 order rewind returned a positive
status, not equal to load point detected)
```

Format 1 RFS FCOD=fc[,NUMB=n]
Format 2 RFS fc[,n]

fc is a filecode assigned to an MT or TK device.
n is the number of records to be skipped. If the NUMB parameter is omitted, NUMB = 1 is assumed.

The tape will be positioned after the nth subsequent record.

One of the following error messages is output if an RFS command is rejected:

```
END OF TAPE IS ENCOUNTERED
EOF ENCOUNTERED
I/O ERR ON M.T. (FCOD=xy), STATUS=abcd
EOS ENCOUNTERED
EOV MARK ENCOUNTERED
NO DATA ON TAPE
```

Format 1 ULD FCOD=fc
Format 2 ULD fc

fc is a filecode assigned to an MT or TK device.

The tape device is switched to the manual state by sending a 'switch off' command.

One of the following error messages is output if a ULD command is rejected:

```
THE FUNCTION HAS NT WORK CORRECTLY (FCOD=xy) STATUS=abcd
```

Format 1 WEF FCOD=fc[,NUMB=n]
Format 2 WEF fc[,n]

fc is a filecode assigned to an MT or TK device.
n is the number of EOF marks to be written. If the parameter NUMB is omitted, NUMB = 1 is assumed.

The following error message may be output:

```
    I/O ERR ON M.T. (FCOD=xy), STATUS=abcd
```

<u>Format 1</u> WES FCOD=fc[,NUMB=n]
<u>Format 2</u> WES fc[,n]

<u>fc</u> is a filecode assigned to an MT or TK device.
<u>n</u> is the number of EOS marks to be written. If the parameter NUMB is omitted, NUMB = 1 is assumed.

The following error message may be output:

```
I/O ERR ON M.T (FCOD=xy), STATUS=abcd
```

Format 1 WEV FCOD=fc
Format 2 WEV fc

fc is a filecode assigned to an MT or TK device.

An EOV is written on the tape.

The following error message is output if a WEV command is rejected:

```
     I/O ERR ON M.T. (FCOD=xy), STATUS=abcd
```

An MT or TK label is 80 ASCII characters, formatted as follows:

CHARS	CONTENTS
0-3	VOL1
4-9	volume serial number
10	security code
11-40	reserved
41-79	owner code.

The tape label is followed by a Tape Mark.

```
Format 1   WLB  FCOD=fc[,SNUM=`sn'][,SCOD=`sc'][,OWNE=`ow']
Format 2   WLB  fc[,['sn'],['sc'],['ow']]
```

fc	is a filecode assigned to an MT or TK device.
sn	is the volume serial number (1-6 ASCII characters), default blanks.
sc	is the security code (1 ASCII character), default 0.
ow	is the owner code (1 or more ASCII characters). If more than 39 characters are entered, only the first 39 will be accepted, default blanks.

One of the following error messages is output if a WLB command is rejected:

```
PARAM ---- TOO LONG
I/O ERR ON M.T. (FCOD=xy), STATUS=abcd
```

BCL Interactive Commands

Introduction

Four BCL commands are available which permit interactive intervention; these are the ROI, ERR, PSE and MES commands. The ERR command is described in the section "BCL Scheduling Commands"; it permits the background user to correct interactively a BCL command which has been rejected by the BCP.

MES Output Message MES

Format MES n

n is a message to be sent to the operator.

The MES command causes the BCP to output the specified message on the system machine device /EF. Having done this, the BCP reads the next BCL command from the device defined by background machine filecode /E0 and continues. The MES command could be used by the background user to inform the operator that he has finished using special paper, for example.

Format PSE n

n is a message to be sent to the operator.

The pause command causes the BCP to output the specified message on the system machine device and then place the background machine into the wait state. The RS operator command must be entered to restart the background machine.

The PSE command could be given, for example, by a background user to request the operator to mount special paper and ready the printer. It complements the facilities available with LKM 6 and LKM 54.

Format 1 ROI FCOD=fc,MESS='m'
Format 2 ROI fc,'m'

fc is the filecode.
m is a message to be sent to the operator, maximal 72 characters.

The operator is sent the following message:

```
m ON dn,da THEN RS, PLEASE
```

where:
'm' is the message as specified on the ROI command
'dn' is the device name
'da' is the device address.

The background machine is placed in the pause state until the RS operator command is entered.

BCL Task Initiation Commands

How the BCP Initiates Tasks

There are 3 BCL commands available for initiating tasks in the background machine. These are:

- Standard Processor Call
- Non-Standard Processor Call
- RUN command.

The processing performed by the BCP on these BCL commands consists of four operations:

1. The BCL command is analysed for correct syntax and rejected if invalid.
2. The location of the program to be executed is calculated. In other words, the DADcode, Userid and the LM file (name, type, version) are identified.
3. The Job Parameter Table is initialised. This table contains the information required by MAS to load the program (calculated in 2 above) and other information about the program (e.g how many lines of print it may output and, for a swappable background machine, for how many seconds it must be resident).
4. The BCP then issues an LKM 3 (Exit) Monitor request with A8 pointing to the JPT (Job Parameter Table). This causes MAS to replace the BCP by the required program which will then be executed.

Task Initiation Commands

Standard Processor Call (See Volume III of this set).

Format 1

```
pname [SIZE={MAX | n}][,DUMP={ALL | PROG | NO}]
```

Format 2

```
pname [MAX|n][,PROG|ALL|NO]
```

pname	is one of the Standard Processor Names. These are the names of the Standard Processors supplied with MAS, for example: FRT Fortran Compiler ASM Assembler RTL RTL/2 compiler LKE Linkage Editor UPD Update LIB Librarian EDF Extended Disc File Processor MAC Macroprocessor.
n	is only required when the background machine was defined as swappable with SCL DCB command, and when the processor assumes it may use virtual addresses larger than that for the last word in the last page containing the load module (or issues any LKM 4). If both conditions are true, n specifies the number of pages to be allocated additional to those required to contain the load module.The default is 0. If the background machine is memory-resident, the SIZE parameter is ignored, and theProcessor is allocated the pages reserved for the background machine by SCL DCB command. If so, it should ensure that it does not issue LKM4 as well asassume it can address pages beyond the end of the load module directly.
MAX	MAS will reserve 32K words for the Processor.
DUMP	is used to request a postmortem dump in the event of a Processor Abort, or if the Processor issues LKM 3 (Exit) with bit 8 of the exit code set (i.e. Exit with postmortem dump). The parameter is ignored if the postmortem dump option was not selected at SYSGEN.
ALL	requests a dump of both the system machine and the background machine.

PROG requests a dump of the background machine only.
If NO is specified, or if the DUMP parameter is omitted, no postmortem dump will be produced.

For a Standard Processor Call the BCP evaluates the program location as:
DAD = /FO
Userid = the first user in the /FO Catalogue.

The Processors must therefore be contained in the Directory for the first User in the Catalogue for the DAD /FO.

Fortran Diagnostic Error Messages

To obtain full documentation about Fortran runtime errors, the parameter D(debug)={YES | FULL} can be specified in the OPT card. With this parameter, Fortran maintains a chain of blocks to print if need be an error message with full documentation (error number, statement number, called at).

To print a full error message correctly, all Fortran programs, main program and subroutines, should be compiled with D=YES or D=FULL.

Problems occur using Fortran routines called by Assembler routines. The diagnostic error message is incorrect when the main program is an Assembler program, because Fortran does not know where the start of the chain is. No problems occur when using Assembler subroutines called from Fortran.

To overcome problems with Assembler main programs, two solutions exist:

1. Let the Assembler program be called by a dummy Fortran main program, consisting of a CALL of the Assembler program, and compiled with D=YES or D=FULL.
2. Insert in the Assembler main program the following instructions:
```
     EXTRN     W:DGER,W:ERLK
```
   and
```
     LDKL      A1,W:DGER
     ST        A1,W:ERLK+2
```

Using one of these alternatives, the full diagnostic error messages are printed correctly for non re-entrant code. When re-entrant code must be generated, only the first method can be used.

Non-Standard Processor Call

A Non-Standard Processor Call is used, either when the Processor name is not one of the standard processor names or the Processor is not contained in the Directory for the first user in the Catalogue for the DAD /F0.

Format 1 pname [SIZE={MAX|n}][,DUMP={ALL|PROG|NO}][,USID=u][,DAD=d]
Format 2 pname [[MAX|n],[ALL|PROG|NO],[u],[d]

pname	is any name of 1-6 ASCII characters except a BCL command mnemonic. In general BCL commands either have a 3 character name or commence with : (colon). The user can thus easily ensure that no non-standard processor has the same name as a BCL command (which would cause the BCP giving an error if it happened).
DUMP	Identical to a Standard Processor Call.
SIZE	Identical to the Standard Processor Call.
u	the userid (1-8 characters).
d	the DAD filecode (/F0 - /FF).

For a Non-Standard Processor Call the BCP evaluates the program location according to whether the parameters USID and DAD are, or are not, specified in the call:

IS DAD SPECIFIED?	Y	Y	Y	N	N	N	N
IS USID SPECIFIED?	Y	N	N	Y	N	N	N
WAS USID ON BCL :JOB?		Y	N		Y	Y	N
WAS DAD ON BCL :JOB?					Y	N	
SET DAD TO SPECIFIED VALUE	*	*	*				
SET DAD TO :JOB VALUE			*				
SET DAD TO VALUE COMPUTED FOR JOB						*	
SEARCH ALL DADS FOR USID				*			
SET DAD = /F0							*
SET USID TO SPECIFIED VALUE	*			*			
SET USID TO :JOB VALUE		*			*	*	
SET USID TO 1ST USER IN DAD			*				*

The following error messages may be output by the BCP if a Standard or Non-Standard Processor Call is invalid:

PARAM. NOT VALID
PROCESSOR NOT CATALOGUED
BAD ASSIGN
I/O ERROR ON DISK
SEARCH DIRECTORY NOT POSSIBLE
PARAM.... MISSING
INVALID PARAM= ....
BAD COMMAND NAME (the command name contained more than 6 characters)

Format 1 RUN optional-keyword-parameters
Format 2 RUN optional positional parameters in the order as occurring below

The optional keyword parameters are:-

KEYWORD	DEFAULT	SPECIFICATION RULES
PROG=	/D6	Up to 6 ASCII chars specifying the name of the program to be executed. It must be the name of an LM file. If omitted, filecode /D6 is assumed to define the program, and is assigned by the LKE to its output load module.
USID=	:JOB USID	Up to 8 ASCII characters, specifying the Directory in which the program is to be located.
DAD=	:JOB DAD	A DAD filecode (/F0 - /FF) defined by an SCL FCD command which specifies the catalogue in which the Directory is located.
VERS=	0	A digit from 0 to 7, specifying the version of the program to be executed. The value must not exceed the value specified on the SMV command to LIB for this Userid.
SIZE=	0	Either the word MAX or a number from 0 to 15, specifyingthe number of memory pages, additional to those contain-ing the load module, which the program assumes it may address using virtual addresses or by giving one or moreLKM 4 requests. This value is only required if the SCL DCB command is defined a swappable background machine.
DUMP=	NO	Only required if one desires a post-mortem dump in the event of the program aborting. Specify ALL to dump the system and the background machine. Specify PROG to dump the background machine only. The parameter is ignored if the post-mortem dump option was not selected at SYSGEN.
PNCH=	1000	Either the maximum number of records which may be punched by this program, or NO if there is no limit.
PRNT=	1000	Either the maximum number of lines which may be printed by this program, or NO if there is no limit.
TIME=	300	Either the maximum number of seconds for which the program may run in the background machine, or NO if there is no limit. The operator should not set the clock off if a value is specified, or it will be ignored.
FR1=,FR3=	0	The values to be placed in the floating point registers FR1 - FR3 when the program is started.

If, during the execution of this program, an attempt is made to exceed one of the three limits PNCH, PRNT or TIME, the program will be aborted.

The default value for PROG is the program output by the last execution of the Linkage Editor (LKE Processor) in the same Job, to the Load Module file (/D6).

The following error message may be output by the BCP if the RUN command is rejected:

```
    INVALID PARAM.=....
    BAD ASSIGN
    USID NAME UNKNOWN
    PROG. NAME UNKNOWN
    SEARCH CATAL. NOT POSSIBLE
    SEARCH DIRECTORY NOT POSSIBLE
    PROG. NAME MISSING
    I/O ERROR ON DISK
    FILE /D6 NOT ASSIGNED
```

INC — Include a Module — INC

The INC command can be used to prepare an Object file for the LKE Processor. There are two types of INC command, depending on whether the modules to be added to the Object file are to be read by the BCP from a non-disc device or from a disc device. The object modules will be output to the disc temporary object library (filecode /D5). If this object library is not assigned, or has been closed (an EOF mark has been written), the BCP will automatically create a new one in the DAD whose catalogue contains a pointer to the Directory for the Userid specified on the BCL :JOB command for this background machine job. The relevant DAD will have been found by the BCP when the BCL :JOB command was processed - either because the DAD was also specified on the BCL :JOB command, or because the BCP searched all the DADs (defined by SCL FCD commands when the background machine was defined) until it found the one whose catalogue contained the relevant Userid. The DADs are searched in the same order they were defined by SCL FCD commands.

INC from a disc device

In this case the INC command has a set of parameters which locate the library to be read by the BCP. It is possible to select an individual module from the library. Filecode /DO will be used by the BCP to assign the file.

```
Format 1   INC  LIBR=f[,MNAM=n|ALL][,USID=u][,DAD=d]
Format 2   INC  f[,[n|ALL],[u],[d]]
```

- f is the library name (filename);
- n is the object module name, or ALL if all the object modules in the library are to be read by the BCP. Default is ALL.
- u identifies the Directory to be searched for the filename. If USID is not specified, the Userid from the BCL :JOB command will be assumed.
- d identifies the DAD catalogue to be searched for the user Directory. If it is not specified, the DAD specified on the :JOB will be assumed. (If no DAD was specified on the :JOB, the BCP will have identified it by searching all the DADs defined by SCL FCD commands, until it found the Userid.)

INC from a non-disc device

```
Format 1   INC  [FCOD=fc]
Format 2   INC  [fc]
```

- fc is the filecode which defines the device type (default=/E2) and address from which the BCP is to read the object modules. It can refer to a disc file. The BCP will read everything up to the EOF mark.

The following messages may be output by the BCP if the INC command is rejected:

```
    PARAM xxxx MISSING
    BAD ASSIGN
    INV. FCOD
    ....= UNNECESSARY PARAMETER
    NO SPACE IN FILECODE TABLE
    NO SPACE FOR FILE DESCRIPTION TABLE
    NO SPACE ON DISK
    UNKNOWN FILE NAME
    DAD UNKNOWN
    USID NAME UNKNOWN
    UNKNOWN LIBR NAME
    ERR. ASSIGN FILE /DO (ST=..)
    NOT AN OBJECT FILE
    ERROR READ/WRITE (STT=...)FILE=...
    UNKNOWN MOD. NAME
    ERROR IN DIRECTORY OBJ.
    DAD FCOD NOT ASSIGNED
    TOO MANY FILECODES
```

Format NOD nodename[,{rovs | *}[,a]]

nodename	is a 6 ASCII character name, conforming to the same rules as any external program name or label. This is copied to the temporary object file (/D5) as a card image, to be processed eventually by the Linkage Editor. It defines a branch in an overlay tree.
rovs	is a 4 character segment name, referring to a secondary load module name. This must not duplicate any other entry point, common block or segment name.
*	is a signal to the Linkage Editor to implicitly name the segments in this compilation in ascending order, in accordance with the parameters on the LKE option command.
a	is an absolute address in hexadecimal and, if present, it forces the loading of the module at this page boundary address.

The BCP merely copies this command into the file defined by filecode /D5 (i.e. the object temporary file). It will be processed later by the Standard Processor LKE.

The following error messages may be output by the BCP if the NOD command is invalid:

```
    BAD ASSIGN
    ERROR. ASSIGN FILE /DO (ST=...)
    PARAM MISSING
    PARAM MUST BEGIN WITH ALPHA CHAR
```

FORMAT 1 HLP [CMND=c][,FCOD=fc]
Format 2 HLP [[c],[fc]]

c is a BCP command name. All keyword parameters for this command are output to filecode fc . If omitted, the syntax of all BCP commands is printed. The keyword parameters are output in the order of the positional parameter input.

fc is the filecode where the commands are printed. Default is filecode 1.

Error messages:

```
INVALID COMMAND
PRINT FILECODE NOT VALID
```

Format 1 SKP [PAGE=n]
Format 2 SKP [n]

n is the number of pages to be skipped on the lineprinter (0-100). The defalt is 1 page.

The SKP command causes the BCP to output a number of skips to Top-of-Form on filecode 2. A negative number of pages or a number of pages greater than 100 is changed into 100.

BCL Datacommunication commands

Some BCL commands exist to assign, scratch and manage datacom devices. These commands are:

DAS	assign a linecode
DDC	disconnect a linecode
DDL	delete a linecode
DHD	halt and disconnect
DHL	stop an exchange

DAS Assign a linecode DAS

Format 1 DAS LCOD=1,DVCE=d

Format 2 DAS 1,d

1 is a linecode from 1 to 255. It remains valid during the Job, just like the ASG command.

d If a binary is supplied, d is assumed to be a linecode and 1 is assigned to that linecode by equivalence. If not a binary number is supplied, d is assumed to be assumed to be a device-name-address with eventually a linenumber. Its lay-out is then: dn[da[ln]]

- dn is a name of a datacom device. See Appendix A
- da is its device address. If omitted, the first occurrence of dn is chosen.
- 1 is the linenumber, only for AMA8 and LSM16. If omitted, zero is assumed.

d may be NO, to assign the linecode to a dummy device.

Error messages

One of the following error messages may be output by the BCL:

DEVICE NAME UNKNOWN
BAD DEVICE ADDRESS
NO DYNAMIC AREA IN THE MACHINE
DEVICE UNKNOWN
SECOND LINECODE NOT ASSIGNED
INVALID LINECODE

Other BCL Datacommunication commands

Format 1	c lcod=l
Format 2	c l

c	is a command mnemonic: DDC, DDL, DHD or DHL.
l	is an assigned linecode.

Depending on the command mnemonic , the linecode is deleted (DDL) or disconnected (DDC), there is a halt and disconnect issued on the line (DHD) or the exchange is stopped (DHL))

Error messages

    DISCONNECT A LINE ERROR, LC=l, STATUS=abcd
For the DDC command, the disconnect (LKM 8 order /11) returned a status #0.
    INVALID LINECODE
For the DDL command, the specified linecode was not assigned or is not a value between 0 and 256.
    HALT AND DISCONNECT LINE ERROR LC=l, STATUS=abcd
For the DHD command, one of the issued LKMs-8 returned a status #0.
    STOP AN EXCHANGE ERROR LC=l, STATUS=abcd
For the DHL command, one of the issued LKMs-8 returned a status #0.

PART 4 FOREGROUND MACHINES

GENERAL

A foreground machine is required only if non-batch user programs are executed. A non-batch program may be loosely defined as a program whose main input arrives as isolated records whose order and volume is unpredictable (unlike batch programs, whose main input is a batched file of records), and where the program cannot delay dealing with the next record until it has finished with the current one. Records may be signals sent by a machine connected to the computer; each signal must be processed (or stored on an I/O device or in memory) immediately it arrives. In other words, the input controls the program with foreground machines, whereas the program controls the input with background machines.

MEMORY ALLOCATION

After IPL, the System Machine will occupy physical memory locations from 0 up to some multiple of 2K words, the exact size depending on how the system was generated. The maximum size of the system machine is 32K words (16 pages) for non Extended mode systems. For systems containing Extended Mode, the maximum length of the system machine is 54K words (27 pages).

The remaining memory locations in the bare machine may be used to create user machines. One background machine and/or one or more foreground machines may be created. These user machines exist only until the next IPL, or until an SCL KIM command is given.

When all user machines have been defined, any memory locations which are still unused are allocated for use as a common dynamic loading area for all machines. The background machine is only required if Standard Processors or user batch programs are to be executed. It may be memory- or disc-resident (as specified by the user on the SCL DCB command). If memory-resident, a number of pages (specified by the user in the SCL DCB command) are permanently reserved for the background machine and cannot be used by foreground machines. If disc-resident, no pages are permanently reserved for the background machine. Instead, the background machine is regarded as a disc-resident task which competes with disc-resident tasks of the foreground machines for memory allocation in the common dynamic loading area shared by all user machines. Memory will be allocated to it according to its software level (specified on the SCL DCB command, or defaulted to 2 less than the Idle Task) in relation to the competing foreground tasks.

MULTI-USER FACILITIES

A foreground machine has only one set of filecodes. It may thus be connected to up to 16 DADs. All the disc files and programs used by a foreground machine must be contained in the Directory for the first user for each of these DADs.

Secondly, if several foreground tasks were run in one foreground machine, all sharing devices, it would be difficult to determine which input was from which task and which output was for which task. Using separate foreground machines enables each application to ensure (via LKM 14 Attach Device and LKM 15 Detach Device) that devices such as card readers and line printers cannot be shared by unrelated tasks.

Thirdly, for unrelated foreground tasks, it is easier to operate if each set of related tasks is performed in its own separate foreground machine, each machine being controlled by a separate user and with its own programs and files.

Fourthly, errors in one foreground machine will not affect activities in the other foreground machines.

MEMORY ORGANISATION

Segments

Each foreground machine consists of a number of segments. The main purpose of this is to allow foreground machines to make optimum use of the MMU (Memory Management Unit) of the P800 range of computers. Foreground machines can thus exceed 16 pages.

Communications Area

Segment 0 is called the Communications Area. It consists of two parts called the Public Library Area and the Dynamic Area.

Public Library Area

The Public Library area contains zero or more user programs.

Dynamic Area

The Dynamic Area contains all the buffers obtained by LKM 4 (get buffer) for programs running in this foreground machine (in any segment).

Segment Organization

Other segments each contain zero or more user programs. Segment 0 is logically part of each of these segments, and programs in these segments may address programs and work areas in Segment 0 by using virtual addresses of 32K downwards (e.g. if Segment 0 was defined as 2 pages on the SCL CMA command, each segment can address it using virtual addresses from 28K to 32K). Segments (other than 0) can have a maximum size of:-

    16 - N PAGES

where N is the number of pages in Segment 0. The size of each segment is defined with SCL SEG commands. Note that the virtual addresses of Segment 0 are always from 32K downwards, even in a segment whose size is less than the maximum allowed. Foreground programs are loaded into Foreground Machine segments (including Segment 0) by FCL REP or FCL LOD commands. Programs within one segment may address each other directly. Programs in different segments can only communicate with each other through Segment 0.

Dynamic Loading Area

Not all foreground machine programs are loaded into segments. For programs which do not require to be in memory all the time, the user may declare them to be disc-resident (with FCL SWP or FCL RON commands), or they may be middle-ground programs. MAS will load these programs in and out of memory as necessary. The Common Dynamic Loading Area will be used for this purpose.

The Common Dynamic Loading Area is shared by all foreground machines and by a disc-resident background machine. It should not be confused with the Dynamic Area in Segment 0 of each foreground machine, or the System Dynamic Area in the system machine.

Example

Suppose the system manager defines a 4 page communications area, of which the public library area is 2048 characters:

```
CMA 4,2048
```

Also, the systems manager defines 3 segments of 9, 12 and 3 pages respectively:

```
SEG 1,9
SEG 2,12
SEG 3,3
```

Now suppose the foreground user declares the following programs, whose size in Kwords is shown in brackets:

```
LOD 1,PGM1,/F2 (5)
LOD 1,PGM2,/F4 (12)
LOD 2,PGM3,/F1 (4)
LOD 2,PGM4,/F5 (4)
LOD 2,PGM5,/F1 (2)
LOD 3,PGM6,/F2 (5)
SWP PGM7,/F5 (9)
SWP PGM8,/F2 (23)
```

Then MAS will construct five addressing areas (i.e. 5 MMU tables). Each area isconstructed page bye page, starting with page 15 and then 14 etc.The pages do not need to be consecutive in memory. The first three areas are allocated pages reserved excusively for this foreground machine. The fourth and fifth areas are allocated pages from the common dynamic loading area shared by all machines (except for pages 12-15 of these areas, which are in Segment 0 of this foreground machine).

The first area will comprise Segment 1 and Segment 0.

PAGE	WORDS	CONTENTS	
3	0-1K	Unused	1st half page 3
3	1K-2K	PGM2	2nd half page 3
4-8		PGM2	
9	0-1K	PGM2	1st half page 9
9	1K-2K	PGM1	2nd half page 9
10-11		PGM1	
12-14		Dynamic Area	
15	0-1K	Dynamic Area	1st half page 15
15	1K-2K	Public Library Area	2nd half page 15

The second area will comprise Segment 2 and Segment 0.

PAGE	WORDS	CONTENTS
0-6		Unused
7		PGM5
8-9		PGM4
10-11		PGM3
12-15		The same pages as pages 12-15 of the first area.

The third area will comprise Segment 3 and Segment 0.

PAGE	WORDS	CONTENTS	
9	0-1K	Unused	1st half page 9
9	1K-2K	PGM6	2nd half page 9
10-11		PGM6	
12-15		The same pages as pages 12-15 of the first area.	

The fourth area will comprise:

PAGE	WORDS	CONTENTS	
0-3		PGM7	
4	0-1K	PGM7	1st half page 4
4	1K-2K	Unused	2nd half page 4
12-15		The same pages as pages 12-15 of the first area.	

The fifth area will comprise:

PAGE	WORDS	CONTENTS	
0-10		PGM8	
11	0-1K	PGM8	1st half page 11
11	1K-2K	Unused	2nd half page 11
12-15		The same pages as pages 12-15 of the first area.	

Note: the fourth and fifth area is an addressing area for swappable programs. These addressing areas are only filled when the swappable program is loaded. If the program is not active or swapped out, only the pages 12-15 (CMA) are recorded in the addressing area.

FOREGROUND TASKS

Initialising the Task Environment

Once a foreground machine has been defined (as explained in a later section), all the programs which may be executed in it must be defined, unless they are Middle-ground programs.

Each program which is to be executed in a segment of a foreground machine (i.e. a program which is to be permanently resident in memory reserved exclusively for one foreground machine) is called a Memory-Resident foreground program. Memory-resident foreground programs may be re-entrant or non-re-entrant.

Foreground programs which are executed in the Common Dynamic Loading Area sharedby all user machines (foreground and disc-resident background) are called Disc-
Resident Foreground programs. They are loaded in and out of the Common Dynamic Loading Area by the system as required. There are three types of Disc-Resident foreground program: Middle-ground, Read-only and Swappable programs. Read-only programs may be reentrant too. Every foreground machine program must be defined before it can be executed, unless it is a Middle-ground program.

For a Memory-Resident foreground program, the definition involves loading it into a foreground machine segment, indicating whether or not it is re-entrant, and specifying the software level (priority) of the program.

For a Disc-resident foreground program, the definition involves specifying whether it is a Read-only or a Swappable foreground program, and specifying the software level of the program.

For a Middle-ground program, there is no need to declare it, nor to connect it to a software level.

Task Scheduling

Once the programs have been defined, they may be used to perform tasks. A foreground task is started either by Activating a program (by an FCL RUN or FCL ACT command given by the foreground machine user, or by an LKM 12 issued by another task in the same foreground machine), or it may be started automatically by the system at a set time, or regularly after a specified period of time has expired. These times are specified by the foreground machine user. For example, a program may be activated at 9.06 a.m., or the user may specify that it is to be activated every 2.6 seconds. This is done either by an LKM 10 issued by another task in the same foreground machine, or by an FCL CNT command from the foreground machine user.

A foreground task ends when the foreground program, which started when the task started, Exits (by LKM 3).

The first task must be started by an FCL RUN, FCL ACT or an FCL CNT command from the foreground machine user. The program activated by this command can then activate other foreground machine programs (by LKM 12) or make them active when timers or clocks reach certain specified values (by LKM 10). These programs can in turn activate, or connect to timers or clocks, other foreground machine programs, and thus create new foreground tasks. Meanwhile the user may have started other tasks by FCL ACT, RUN or CNT commands.

Activation Queues

A foreground machine task is one execution of a foreground machine program. It is possible for the same foreground program to be required for several tasks simultaneously. For example, program 1 may Activate program 7 which then starts. It is then interrupted by program 4 because the latter has a higher priority (a lower software level), and program 4 then activates program 7. MAS maintains an Activation Queue for each non re-entrant foreground program. Every time a foreground machine program is activated (or made active by timers or clocks), a new entry is created in its Activation Queue. Whenever this program becomes the highest priority executable task in the bare machine (i.e. in all machines - system, foreground and background), the first task in its Activation Queue is resumed. When this task Exits, its entry is removed from the program's Activation Queue. Multiple activations of the same foreground machine program are thus processed one at a time on a FIFO (first-in-first-out) basis.
Re-entrant foreground programs (which are always memory-resident when declared with FCL REP commands and always disc-resident when declared with the RON command) do not have activation queues. A re-entrant program can perform several tasks conjointly.

Activation with Scheduled Labels

If a foreground machine program uses an LKM 12 to Activate another program, the system branches to the scheduled label address (if present) when the Activated program Exits.

If the Activating program has lower priority than the Activated program, then the NSI (next sequential instruction) to the LKM 12 will be executed if:

The Activated program enters a Pause or Wait.
The Activating program becomes the highest priority executable task in the bare machine.

If the Activating program has higher priority (lower software level) than the Activated program, then the NSI to the LKM 12 in the Activating program will be executed before the Activated task starts. The Activated task will only be capable of starting when the Activating task enters a Pause, Abort or Wait.

If the Activating program activates itself, then a new entry is created in the Activation Queue for the program and the Activated task can only start when the Activating task Exits. This can never occur if the Activating task is waiting for the scheduled label to complete. A deadlock will thus occur.

Task Synchronisation

If one foreground task wishes to wait for another task in the same foreground machine to complete, then the following procedure may be used:

When program 1 activates program 2 by LKM 12, A8 is set to point to an event byte.

When program 1 wishes to wait for program 2 to Exit, it ensures that A8 is pointing at the event byte, and issues an LKM 2.

When the task starts (i.e. when program 2 is the highest priority executable program in the bare machine and the task activated by program 1 is at the top of the Activation Queue for program 2), program 2 receives the value of A8 set by program 1 when it issued the LKM 12.

Just before program 2 Exits, it should ensure A8 has the same value as on entry, and issue LKM 18 to set the event bit.

Note that if an activated task aborts, bit 15 of A8 will be set to 1 and the scheduled label of the activating task will be started.

FOREGROUND PROGRAMS

Memory-Resident

Memory-resident foreground programs are loaded by the Foreground user into foreground machine segments, with FCL REP or FCL LOD commands. They may be loaded into Segment 0 (the Communications Area) or into any other segment defined by an SCL SEG command, submitted by the System Manager when the Foreground machine was defined.

Memory-resident foreground machine programs may be contained in any DAD assigned to the foreground machine (by SCL FCD commands given by the System manager when the Foreground machine was defined), but they must be contained in the Directory for the first userid of any such DAD. The DAD containing each program is specified on the REP or LOD commands.

Several memory-resident foreground programs may be loaded into one segment. Programs in one segment can address each other directly. All programs can address Segment 0. Programs in different segments of the same foreground machine can thus communicate with each other through Segment 0. They can also communicate via Segment 0 with disc-resident programs of the same foreground machine and executing in the Common Dynamic Loading Area. Once loaded into a segment, memory-resident foreground programs remain in that segment until the next IPL, unless an FCL KIL command is given.

If a memory-resident foreground program is an Overlay program (i.e. it has been created by the Linkage Editor to contain a root segment and one or more overlay segments), the root segment will contain LKM 27 (inserted by LKE) which will, in the case of disc-resident overlays, load the overlay segments from the Load Module file into the same foreground machine segment as the root segment. The root segment will be permanently resident in memory. The overlay segments will be loaded one at a time, as required, into a fixed area in the foreground machine segment.

In the case of programs containing memory-resident secondary load modules, these are loaded into memory before run time by means of LOD commands, and the root segment will contain LKM 57 requests to connect these modules into the program path by modifying the MMU Table. The technique of using memory-resident secondary load modules is discussed in the Software Processors Manual in the section on the Linkage Editor.

Each memory-resident non-re-entrant foreground program (declared by FCL LOD) has a software level (priority) which is associated with every task performed by it.

The software level of memory-resident re-entrant foreground programs (declared by FCL REP) is the minimum software level which may be used. Each task will be attached to the lowest unconnected software level which is not less than this minimum. A program cannot be activated, nor connected to a timer or clock, until it has been connected to a software level. Memory-resident foreground programs are connected to a software level, either by an FCL CNL command from the foreground machine user, or by an LKM 20 issued by another program in the same foreground machine.

Disc-Resident

Disc-resident foreground programs are defined by the foreground machine user (unless they are Middle-ground programs) by FCL RON or FCL SWP commands. They may be contained in any DAD assigned to the foreground machine (by SCL FCD commands given by the System manager when the foreground machine was defined), but must be contained in the Directory for the first userid of any such DAD. Each RON and SWP command specifies which DAD contains the program. Several disc-resident programs can be specified for one foreground machine, but they can only address each other or address memory-resident foreground programs via Segment 0. Disc-resident foreground programs can activate other foreground programs (either disc- or memory-resident) for the same foreground machine. This they do using LKM 12, permitting them to pass parameters to the activated program via A3 or A8 and to receive parameters from it via A8 when it has completed (providing the activating program has a scheduled label or waits for the activated program).

All disc-resident foreground programs may address Segment 0 of the same foreground machine, by using virtual addresses of 32K downwards.

Disc-resident foreground machine programs are loaded and unloaded as necessary in the Common Dynamic Loading Area, also used for the Background machine if the SCL DCB command did not define a memory-resident background machine.

Overlay disc-resident foreground programs partially control their own loading into the Dynamic Loading Area. The root segment is loaded and unloaded as for any normal disc-resident foreground program. The overlay segments are loaded by the root segment as required (by LKM 27 inserted in the root segment by the LKE Standard Processor). Overlay disc-resident programs must be either Middle-ground or defined by SWP and not RON.

Each disc-resident foreground program must have a priority (software level), which is associated with every task performed by the program. A program cannot be activated, nor connected to a timer or clock, until it has been given a priority (either by an FCL CNL command from the foreground machine user, or by an LKM 20 issued by another program in the same foreground machine). Middle-ground programs are connected automatically by MAS, when they are activated, to the lowest unconnected software level.

Disc-resident foreground programs may be Middleground, Swappable or Read-only.

Read-Only Programs

These are declared by FCL RON commands given by the Foreground user. Its core-image is written to (disc temporary file in) the DAD D:CI. This DAD has no user, and is automatically connected by MAS to filecode /F1 of the systemmachine.

If a higher priority program requires to use memory in the Common Dynamic Loading Area occupied by a Read-only foreground program, the latter is over-written<u>without</u> being written out to the core-image file sectors created by the RON command. When the program is re-loaded, the registers are restored to their values when the program was over-written. Processing resumes at the instruction being executed when the program was over-written. Read-only foreground programs cannot, therefore, modify their core-image, and may contain only instructions and constants. All variable-content memory (memory whose contents vary for different tasks, or at different times during the same task - such as I/IO buffers, accumulators and switches) must be contained in memory in the Dynamic Area of Segment 0. LKMs 4 and 5 are used to obtain this memory and to free it. If a Read-only, Reentrant or Read-only-reeentrant program tries to modify its core-image, it will be aborted with error 3 (protection) because the pages wherein the program is loaded are set to write protect.

Swappable Programs

These are disc-resident foreground programs which are liable to modify their core-image during the execution of a task. All disc-resident foreground programs which are also Overlay programs must be either Middle-ground or Swappable. Swappable programs are defined by FCL SWP commands from the foreground machine user.

Swappable programs may contain I/O buffers, work areas, switches and other variable-content memory. They do not have to allocate these in Segment 0 with LKM 4, though they may do this if required - because another program in the same foreground machine needs to be able to address them directly. Swappable programs cause the total throughput of the bare machine to be less than would be the case if they were Read-only programs. This is because the swapping operation involves more disc accesses than is required by Read-only programs.

Middle-ground Programs

Middle-ground programs are identical to swappable programs, except that they are not declared by SWP commands, and are not connected to a software level by the user but by MAS itself. Middle-ground programs are located by searching (when they are activated) all DAD's known to this foreground machine in the order /F0 to /FF. Only the directories for the first userid of each DAD are examined.

When a middleground program is activated, the system in fact executed first a SWP and a CNL (connect level) command before the activating command is executed. On exit of a middleground program, the system executes a KIL program command, except when the program is aborted. On abortion, the middleground program is not killed and a new activation results in execution of the same core image.

Swapping

When a swappable program is declared (by an FCL SWP command), MAS writes the program from the relevant DAD to the Core Image file (contained in the DAD D:CI - system machine filecode /F1), and reserves an area for the program in the same DAD to contain the Swap Image.

When a task which is to be performed by a swappable program is started (i.e. when there are enough free pages in memory, or a program has been swapped out to free enough pages, and this is the first activation), the swappable program is loaded into the Common Dynamic Loading Area from its Core Image created by the SWP command. This load is done even if the program is already in the Common Dynamic Loading Area (because it has just finished the task that was previously at the top of its Activation Queue). The reason is that, unlike Read-only programs, Swappable programs may modify their instructions and switches while executing a task. To ensure that these are reset to their initial values at the start of the next task performed by the program, the program must be reloaded from its initial core image. However, a special loading mechanism is available to prevent this time-consuming reloading. Of course, using this special loading mechanism the swappable programs must be re-usable.

Whenever the dynamic loading area is occupied, either by a program declared with a SWP command or by a Middle-ground program which has exceeded its minimum core-residence time, the program cannot simply be over-written (as is the case for a Read-only program). The swappable program may have modified its core-image, while executing its current task. MAS writes any pages that have been modified since the program was last swapped into the Swap Image area reserved by the SWP command for this program.

When the swappable program again becomes eligible to run and it is not a new task to be performed (i.e. the interrupted task is to be resumed), it is reloaded by loading all the pages in its swap image, and loading any other pages from its core image.
The core image file in the DAD D:CI (which is defined by system machine filecode /F1) thus contains:

- The initial core image of all read-only and swappable programs for all foreground machines. These are copied from Load Module libraries when the foreground machine users give the FCL commands RON or SWP. The DADs containing the Load Module libraries (in the first userid) containing the programs are specified on the RON and SWP commands.
- A swap image of all swappable programs (declared by SWP, or middle-ground programs) for all foreground machines. Each swappable program has one swap image area, which contains either garbage (if the program has never been swapped out), or the memory-image of the modified pages when it was last swapped-out. The background machine will be one swappable program if the SCL DCB command did not specify a memory-resident background machine.

The physical swapping operation may be delayed because the swap event count is non-zero. Each swappable program has a swap event count. Whenever it issues an LKM which will modify its allocated memory (for example an LKM 1 to Read a record into a buffer embedded in the program), MAS increments the program's swap event count. If a program is to be swapped-out, the physical operation only starts when its swap event count reaches 0.

If the physical swapping of a program is delayed by one of the above causes, no other program in the same machine can be swapped out until the first swapping is finished.

MIDDLEGROUND PROCESSING

To perform Batch processing with more than one user at a time, Middleground processing has been developed.

Middleground processing enables the user to run Batch programs (e.g. Software processors) in a Foreground machine.

Use of Middleground

To run in Middleground, the most suitable way is to declare a Foreground machine with filecodes /E0 and /01 assigned to an interactive device like TY or DY.

The filecode /F0 should be assigned to a user's DAD, and the system DAD should be assigned to a higher /Fx filecode. The advantage of this is that all processors, and some LKMs, use as default value for assigning files etc. the first userid on DAD /F0, which is the user's userid in this case.

It will be clear that, for each foreground machine which is meant to be used for Middleground processing, the filecodes /E0, /01 and /F0, assigned to interactive devices or user DADs respectively, should be independent of other machines.

Middleground programs can be activated via the FCL commands RUN and ACT, and via LKM 12 (Activate). As Middleground simulates Batch processing, the RUN command will be the normal method of activation.

Middleground programs need not be declared (via SWP, LOD, REP or RON commands), nor need they be connected to a level.

On activation, MAS searches for the PCT of the specified program. If not found, the program is assumed to be Middleground. The program is searched for in the directory of the first userid of each DAD declared in the machine, starting with DAD /F0. When found, a PCT is created and the program is loaded and connected to the highest free software level. Then it is activated and handled as a swappable program.

When the program exits, it is disconnected from its level and removed from memory, and its PCT is deleted.

Printing

With printing, care should be taken. The activities of other users must be allowed for, otherwise outputs will be mixed.
Two possibilities exist of avoiding mixed print output:

1. Assign filecode /02 to the interactive DY/TY device to which the filecodes /E0 and /01 have been assigned, or to another device which can be used exclusively by one user. If a user wants to print something, he must warn the other users not to use the lineprinter.
2. A more convenient way is to spool the lineprinter. Outputs of one process are gathered and printed together afterwards, so that mixing of outputs is impossible.

Omission

When /F0 is assigned to a user's DAD in an LKE step, the SLIB parameter should be set to some value (NO or a library name), because on the user's DAD the SYSLIB object library (default for SLIB parameter) will usually not be present.

Example

Declare a Foreground machine:

```
FCL:DCF FORG
  D:CMA 1,200
  D:FCD /C0
  D:FCD /C1
  D:FCD /F0,/C1,USER
  D:FCD /F6,/C0,SUPERV
  D:FCD 1,DY18
  D:FCD 2,DY18
  D:FCD /E0,DY18
  D:DEN
FCL:BYE FORG          start the Foreground machine
FCL:RUN LIB
LIB:CSF INAM=1,ONAM=10,ODAD=/F0
CSF:       IDENT    TEST
CSF: --- input of a program
CSF: ---
CSF::EOF
LIB:KPF FCOD=10,FNAM=TEST,TYPE=SC
LIB:LEN
FCL:RUN ASM
ASM:OPT PROG=TEST
FCL:RUN LKE
LKE:OPT CATL=TEST,MAP=YES,SLIB=NO
FCL:RUN TEST
FCL:BYE               end of the session.
```

Note that, running a middleground program, from MAS rel 8, it is sufficient to give the name of the program, without RUN just as in the background machine.

DEFINING A FOREGROUND MACHINE

Sequence of Operations

Only the System Manager may define foreground machines. SCL commands entered from the device assigned to the system machine filecode /E0 are used to define foreground machines. Once a foreground machine is defined, it may be started by an SM operator command entered from the device assigned to the system machine filecode /EF. The foreground machine will then start to read FCL commands from the device assigned to its filecode /E0.

Defining a foreground machine involves specifying:

- The name of the machine (to be used subsequently in operator and SCL commands).
- The number of memory-resident segments and the size of each segment.
- The devices and DADs to be used by this foreground machine and the filecodes which identify them.
- If necessary, the System manager may also override certain SYSGEN default values which control the tables MAS constructs for a new foreground machine. These values are the number of scheduled labels usable in user programs in this foreground machine, the maximum number of filecodes which may be assigned by this foreground machine, and the maximum number of blocking buffers which may be used by this foreground machine.

When all SCL commands necessary to define a foreground machine have been entered, the SCL DEN command should be given. A full description of the SCL commands is given in Part 2: The System Machine, Chapter 7. When a foreground machine has been defined and started, the FCL commands specify:

- The memory-resident programs to be loaded (from which DAD and into which segment).
- The disc-resident programs which may be used (and in which DAD they reside). Middle-ground programs are located automatically by MAS.
- The priority (software level) to be associated with each program. All tasks performed by one program (unless it is re-entrant) have the same software level. Middle-ground programs are connected automatically by MAS.
- The programs to be activated immediately, and those to be activated according to timers or clocks.

Once programs have been activated, further FCL commands allow the foreground user to control his application.

Example of a foreground machine declaration:

```
    DCF  MNAME,2
    CMA  4,2048
    SEG  1,9
    SEG  2,12
    FCD  /C0
    FCD  /C1
    FCD  /C2
    FCD  /01,TY10
    FCD  /02,LP07
    FCD  /E0,TY10
    FCD  /F6,/C0,SUPERV
    FCD  /F0,/C1,DAD1
    FCD  /F1,/C2,DAD2
    DEN
```

FCL COMMANDS

When a foreground machine has been started, using the SM operator command, FCL commands will be read from the device that was assigned to filecode /E0 during the machine definition phase.

FCL commands are used to complete the initialization of the foreground machine, load application programs, assign priorities, connect programs to clocks and timers, and activate programs.

The last FCL command should be BYE, which causes the FCL control program to exit.

If an FCL command is rejected, the file codes /01 and /02 are used (in the same way as for the system machine) to log and submit corrections.

Error Messages

Two general error messages can occur while processing FCL commands; these are:

```
INPUT COMMAND I/O ERROR
COMMAND UNKNOWN
```

Further error messages, specific to the command causing them, are described under the relevant command.

Corrections

An input command error results in the following messages being output on the console (filecode /01):

- The erroneous command;
- An error message;
- FCL: (prompting for an FCL command).

The system then waits for the correct message to be input via the device attached to filecode /E0.

Filecodes

The following filecodes must be assigned by SCL FCD commands before starting the foreground machine, since the FCL control program uses them:

/01	FCL correction device; this must be a console or VDU, and is used to input corrections to rejected FCL commands. Subsequent corrections will be read from the device assigned to filecode /E0.
/02	Print device; used for outputting FCL error messages. This may be assigned to the same device as filecode /01 or /E0.
/E0	The device from which FCL commands will be read when the foreground machine is started. This may be assigned to the same device as /01 or /02.

/FO-/FF These are assigned to all the DADs used by this foreground machine. All programs and files for an application are contained in the first Userid Directory within these DADs. At least one DAD must therefore be specified, since otherwise no user programs could be executed in this foreground machine.

If catalogued procedures are used, filecode /FO must define the DAD containing the file F:PROC.

Foreground programs should not use filecodes /EC, /ED or /EE, since these are used for processing foreground catalogued procedures.

Parameters

Positional parameters are used, each of which is limited to 6 ASCII characters (truncated if more). Hexadecimal numbers must be preceded by a slash (/) character.

Declaring Foreground Programs

All user programs (apart from middleground programs) must be loaded or declared using one of the following commands:

LOD - Load a memory resident program
REP - Load a re-entrant program
RON - Declare a read-only program
SWP - Declare a swappable program.

Default values

References to any disc file in a foreground machine apply to the directory of the first Userid of the a specified DAD (if the DAD filecode is not specified, the default is DAD /FO). However, from MAS release 8.50, these default can be overruled by the FCL JOB command or by LKM 73 (Set default DAD and Userid).

A description of these follows.

FORMAT

```
LOD s,p,d[,l]
```

s is the segment number into which p is to be loaded, expressed as an integer in the range 0 to the maximum number of segments defined for this foreground machine when the 'SCL DCF' command was entered.

pis the name of the program which is to be located in DAD d and loaded into segment s.

d is the DAD containing program p as a load-module file, located via the Directory for the first Userid.

l is an optional field, specifying the maximum number of scheduled labels which may be started during any task of this program. If this field is not entered, the default value is that specified on the 'SCL LAB' command or, failing this, the SYSGEN default value. Since MAS must reserve space in the System Dynamic Area in order to preserve data as each scheduled label routine is started, it is in the user's interest to keep this reserved area to a minimum. If the number of possible simultaneous scheduled label routines is less than the default value, this field should be specified.

Error Messages

```
THIS PROG DOES NOT EXIST
PROG. ALREADY LOADED
SEGMENT NO. TOO HIGH
LACK OF ROOM IN THE SEGMENT
DAD FILE CODE ERROR
DYN AREA OVERFLOW
TOO MANY SCHEDULED LABELS
PARAMETER ERROR
```

Purpose

To load a secondary load module into memory; this is a module output by the Linkage Editor, which can be referenced by one or more other modules called primary modules.

FORMAT

```
LSM n,fc[,{R | W}]
```

n is the 4 ASCII character name of the secondary module, output by the Linkage Editor and catalogued in the first userid of the DAD fc.

fc specifies the DAD filecode from which the secondary load module will be loaded. Otherwise the DAD invoked is the corresponding one declared in the current user foreground machine under the same filecode.

R is entered if the secondary load module is to be considered read-only.

W is entered if the secondary load module is to be considered as modifiable at execution time. In this case it can be used by only one primary module.

The default value is 'R'.

Remarks

Before referencing the secondary load module, the primary load module must have been linked to it, i.e. the pages of the secondary load module must have been connected by means of LKM 57. This LKM 57 coding is generated by the LKE, linking the primary load module with its secodary load modules.

Error Messages

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

```
UNKNOWN SECONDARY LOAD MODULE
SEC. LOAD MOD. IS SEGMENTED
SEC. LOAD MOD. ALREADY LOADED
DAD FILECODE ERROR
DYN. AREA OVERFLOW
PARAMETER ERROR
DAD SECTOR TOO LONG
I/O ERROR
TOO FEW FREE PAGES
SEC. LOAD MOD. TOO LONG
SLM NAME EXCEEDS 4 CHARACTERS
```

Format

```
REP  n,s,p,d[,l]
```

n	is the maximum number of simultaneous activations allowed for program p.
s	is the segment number into which this re-entrant program is to be loaded. It is the number given in the 'SCL SEG' command which defined this segment of the machine. Zero means that the program is to be loaded into the Public Library Area.
p	is the program name, consisting of from 1-6 ASCII characters, defining a Load Module.
d	is the DAD filecode, as specified on the 'SCL FCD' command which defined the DAD of this foreground machine. The re-entrant program p is entered in the directory of the first Userid.
l	specifies the maximum number of scheduled labels which may be simultaneously active.

Remarks

A re-entrant foreground program does not have an activation queue; if the program is already performing a task when a second activation is requested, this second activation occurs immediately. Each new activation causes a new program control table (PCT) to be created.

Once the user has connected a re-entrant program to a level (by means of the FCL CNL command or LKM 20, for example), each new activation will be connected to the next highest unconnected level (i.e. the next lowest priority).

A re-entrant foreground program cannot be specified on a PS, RS or KI operator command. If it is the subject of an AB operator command, all currently active tasks of this program are aborted.

Error Messages

If the command is rejected one of the following error messages will be output:

```
THIS PROG. DOES NOT EXIST
PROG. ALREADY LOADED
SEGMENT NUMBER TOO HIGH
LACK OF ROOM IN THE SEGMENT
DAD FILE CODE ERROR
DYN. AREA OVERFLOW
TOO MANY SCHEDULED LABELS
PARAMETER ERROR
```

Purpose

To declare a disc-resident Read-only Program. This command allows the system to allocate disc space (in the core image disc area) to receive the initial core image of the read-only program, and to set up a Program Control Table for the program.

Format

```
RON  p,d[,l][,t][,R,a]
```

p	is the program name.
d	is the filecode defining the DAD whose first Directory contains the program.
l	is the maximum number of scheduled label routines which will be activated simultaneously.
t	is the minimum core-residence time in seconds (default value is in the CVT).
R	indicates that the program is to be considered as Read-only-Reentrant.
a	is the maximum number of simultaneous activations for the program.

Remarks

Read-only programs cannot be overlaid, must not alter their core image during execution and must contain only instructions and constants. All variables must be allocated in the dynamic area of Segment 0, using LKM 4, and these areas must be freed, before a task ends, using the LKM 5 request.

The maximum length of a read-only program (excluding its dynamic areas) is 16 pages minus the number of pages in Segment 0.

If the Read-only program has been declared as reentrant, the pages occupied by the program are only freed when all activations for the program have been finished.

Error Messages

If this command is rejected, one of the following error messages will be output on the device assigned to file code /01:

```
PARAM ERROR
PARAM MISSING
USER F.C. PROG NOT ASGN
PROGRAM ALREADY DEFINED
PROGRAM DOES NOT EXIST
DISK I/O ERROR ON READ L.M.
SYSTEM DYNAMIC AREA OVERFLOW
D:CI F.C. /F1 NOT ASGN
SYST D:CI FILE OVERFLOW
DISK I/O ERROR
PROGRAM TOO LONG
PROG SEG. NOT READ ONLY
     the read-only program is an overlay program.
```

Purpose

To declare a swappable program. This command causes the system to copy a program from a user DAD to the system core-image file, D:CI, and to reserve another area in this file to contain the swap image of this program. This second area is only used if the program is swapped out and has modified one or more of its core-image pages in memory at the time the swap-out occurs. The user must declare all the swappable programs used in the application before they are activated, otherwise they will be considered to be Middleground programs.

Format

```
SWP p,d[,l][,t]
```

p is the program name.
d is the filecode of the DAD, in whose first directory the program is entered.
l is the maximum number of scheduled label routines which may run simultaneously.
tis the minimum core-residence time in seconds (default value is in the CVT).

Remarks

Swappable programs may be overlay programs.

The maximum size of a swappable program (excluding any buffers assigned in the dynamic area of Segment 0) is 16 minus the number of pages in Segment 0.

Error Messages

```
PARAM ERROR
PARAM MISSING
USER F.C. PROG NOT ASGN
PROGRAM ALREADY DEFINED
PROGRAM DOES NOT EXIST
DISK I.O. ERROR ON READ L.M.
SYSTEM DYNAMIC AREA OVERFLOW
D:CI F.C /F1 NOT ASGN
DISK I/O ERROR
PROGRAM TOO LONG
```

Starting the User Program

Once all the user programs have been declared, the application to be run in the foreground machine can be started.

Until the 'BYE' command is given deleting the FCL control program, the user application, once started, executes in conjunction with the control program. Since this usually has a lower priority than any user task, then, with the exception of the 'FCL RUN' command (which causes suspension of the FCL task until the user task has exited), the user task will be executed and the FCL control program will only be able to read commands when the user programs are suspended, have exited, or have aborted.

The following commands are relevant to the starting up of user applications:

ACT	Activate a program
CNL	Connect a program to a level
CNT	Connect a program to a timer or clock
RUN	Run a program.

Purpose

To start a foreground task, i.e. one execution of a foreground program.

Format

```
ACT  p[,a][,b]
```

p is the program name.

a is a value to be placed in register A3 before the program is activated. It is an optional parameter; the default value is 1.

b is a value to be placed in A4 before the program is started. The default value is 1.

Note: Before activation, the program must have been connected to a level. In the case of middleground programs this connection is made automatically by MAS on receiving the 'ACT' command. For all programs, except reentrant ones, an activation queue per program is created, when the program to be Activated is already active.

Error Messages

PROGRAM UNKNOWN
PROGRAM NOT CONNECTED
LACK OF ROOM IN DYN. AREA
PARAMETER ERROR
TOO MANY REENTRANT ACTIVATIONS
NO SOFTWARE LEVEL FOR REENTRANT PROGRAM
PROGR ABORTED (EVC#0)
the program to be activated has been aborted and there is still an outstanding event. No further activations can be made before this event is ready.

Purpose

To connect a program to a software level prior to activation.

Format

```
CNL p,n
```

p	is the program name, and must have been specified on a previous REP, LOD, RON or SWP command for this foreground machine.
n	is a number defining the software level of program p.

Remarks

The lower the software level to which a program is connected, the higher is its priority. Whenever an interrupt occurs, the task which is currently executing is suspended and MAS starts or resumes processing the highest priority activated program which is in an executable state.

Each program in the bare machine must have a different software level.

The software level of the Idle Task (specified at SYSGEN) must be higher (lower priority) than that of all other tasks. Only one program may be connected to one level. The maximum level is 239.

Error Messages

```
PROGRAM ALREADY CONNECTED
SOFTWARE LEVEL ALREADY USED
PROGRAM NOT YET DECLARED
PARAMETER ERROR
LEVEL NUMBER TOO HIGH
```

Purpose

To specify a start-time for an initial execution of a program, and optionally the time interval for periodic re-activation of the same program.

Format 1 Connect to a Timer:

```
CNT p,d,1,r,n
```

p	is the program name. No middleground.
d	is the length of one time period, and is an integer in the range 1-5 having the following meaning: 1 = minutes 2 = seconds 3 = 100 msecs 4 = 20 msecs 5 = a non-standard timer (only if the bare machine has the clock option).
1	is an indication that the program must be connected to a timer. The parameter is a 1 (one).
r	is the reactivation parameter. Once the program has been activated (which will occur for the first time after 'n' periods of length 'd'),it will be re-activated every 'r' periods of length 'd'. The value of r can range from 0-2047, but if it is zero, the program will only be activated once.
n	is an integer in the range 0-32767, specifying the number of periods of length 'd' which are to elapse before the first activation.

Format 2 Connect to a Clock:

```
CNT p,d,2,r,t
```

p	is the program name. No middleground.
d	is the length of one time period (as for Format 1).
r	is the reactivation parameter, and is an integer in the range 0-127. Once the program has been activated (at time 't'), it will be reactivated every 'r' periods of length 'd'. If 'r' is zero, the program will only be activated once.
t	is the time at which the program is to be activated, entered as: HH,MM,SS

where HH is an integer in the range 0-23 (hours);
MM is an integer in the range 00-59 (minutes);
SS is an integer in the range 00-59 (seconds).

Remarks

The program 'p' must have been connected to a level before this command is given. A program may be connected to several timers and clock values simultaneously. The system of activation and reactivation is administered via the MAS table T:RTC, which contains seven timer addresses (one for each of the six time periods and one for the clock). Each address in this table points to the first block in a chain of blocks for this timer.

Each of these blocks contains details of the times at which a program is to be activated, and points to the PCT identifying the program. Several blocks in one or more of these chains may point to the same PCT.

These blocks are created by the CNT command or by LKM 10, LKM 22 or LKM 28, and are removed by the DST command or by LKM's 11, 21 and 29.

The non-standard timer is a special hardware option which counts periods of time whose length is specified by the user.

If a non-re-entrant program connected to a timer has not exited before the time for the next activation has expired, the next (and subsequent) activations are placed on an activation queue until the exit occurs. Only one activation can be processed at a time; the queue is processed on a first-in first-out basis. The length of time taken to execute one activation of a program varies from activation to activation, depending on the software level of the program and the nature of the other active tasks occupying the bare machine.

If a program connected to a timer is aborted, the current task never exits, the tasks on its activation queue are ignored and no new activation queue entries will be made. The program can only be reactivated by an 'FCL RUN' command, or the abort condition must be removed by means of the 'FCL RAB' command.

Error Messages

```
PROGRAM NAME IS NOT ALPHA
PROGRAM NOT FOUND
WRONG TIMER NUMBER
WRONG CNT COMMAND
WRONG FORMAT
WRONG PULSE RATE
PULSE RATE TOO LARGE
WRONG NUMBER OF CYCLES
YOU MUST BUY THE CLOCK OPTION TO USE TIMER 5
WRONG DELAY VALUE
TOO MANY PARAMETERS
WRONG TIME
1 DAY = 24 HOURS
1 HOUR = 60 MINUTES
1 MINUTE = 60 SECONDS
PROGRAM ALREADY CONNECTED TO THIS TIMER
NOT ALLOWED FOR MIDDLEGROUND PROGRAMS
```

Purpose

Performs the same function as the ACT command, the difference being that no further FCL commands for this foreground machine will be accepted until the activation has been completed and the program has exited or been aborted.

Format

```
RUN p[,a3[,a4]]
```

p	program name declared in a LOD, REP, SWP or RON command for this machine. The program must have been connected to a software level, or be a middleground program.
a3	value to be placed in A3 on program entry. Default value: 1.
a4	value to be placed in A4 on program entry. Default value: 2.

Error Messages

```
PROGRAM UNKNOWN
PROGRAM NOT CONNECTED
PARAMETER ERROR
PROGRAM ABORTED (EVC#0)
```

Controlling the User Application

The following FCL commands can be used to control the running of user programs once they have been declared and started by means of the commands previously described. Most of the functions performed by the following commands can also be performed by means of LKM requests.

Format

```
ABT p
```

p is the name of the program to be aborted.

Remarks

The 'FCL ABT' command provides the foreground user with similar facilities to the 'AB' operator command (see Part 1, Chapter 5).

Abort is ineffective as long as the program is in a wait state.

Error Messages

```
PARAM ERROR
PROG NAME MISSING
PROG INACTIVE
PROG ALREADY ABORTED
```

Purpose

To assign filecodes to:
- a) another filecode
- b) a physical non-disc device
- c) a disc temporary file
- d) a disc catalogued file
- e) a DAD.

Format 1 - to another filecode:

```
ASG fc1,fc2
```

Format 2 - to a physical non-disc device:

```
ASG fc1,dn[da[ln]]
```

Format 3 - to a disc temporary file:

```
ASG fc1,DDdc,ft[,granules[,NC]]
```

Format 4 - to a disc catalogued file:

```
ASG fc1,DDdc,ft,fn
```

Format 5 - to a DAD:

```
ASG fc1,dk,dan
```

fc1	is the filecode to be assigned and for which an entry is to be created in the filecode table for this foreground machine.
fc2	is a filecode which has already been assigned in this foreground machine, and to which filecode fc1 is to be assigned.

Note - the filecode fc2 must not have been assigned to another filecode.
- if the filecode fc2 is subsequently reassigned, filecode fc1 is also automatically reassigned.

dn is a non-disc device name, as given in Chapter 3.
da is the device address. If omitted, the first device encountered in the MAS device table having device name 'dn' will be assigned.
dc is the DAD filecode, in the range /F0 - /FF, specified in the SCL FCD command which defined the DAD in which the file to be created or read will or does exist in the Directory of the first userid for this foreground machine.
ft is the file type, and is one of the following:-
- UF - User File
- SC - Source File
- OB - Object Module File
- LM - Load Module File
- EF - Extended Disc File.

ln	is a linenumber that can be specified for devices connected to the AMA8. This linenumber may also be added to the AMA8 device address in the da parameter.
granules	is the number of granules to be reserved. The default is one.
Note:	If the file is to be created by a Direct Write (LKM 1) request, the number of granules must be allocated at Assign time. If a Direct Write is attempted to a sector of a file beyond the number reserved by the FCL ASG command, the write will be rejected with a status of /10.

NC is a code indicating that the granules need not be consecutive, and can only be entered if a number of granules are allocated.

fn	is the filename. This file, of type `ft' and version 0, must exist in the directory of the first Userid of the DAD defined by DAD code `dc'.
dk	is the disc filecode (/C0 - /CF).
dan	is the DAD name.

Remarks

A new entry will be created in the filecode table for this foreground machine, providing the number of entries does not exceed the value specified on the `SCL MFC' command when this foreground machine was defined (or the SYSGEN value, if no MFC command was given).

The ASG command is an alternative method of assigning foreground machine filecodes to those of using the `SCL FCD' command or the LKM 23 request.

In the assign for a TDFM file, the filename specified is the descriptor file. The other files, the subfiles, containing the keys and the data are assigned automatically by TDFM. These subfiles are searched via the volumenumber of the disc where the subfiles reside. This volumenumber is recorded in the descriptor file. Due to this, problems can occur when two discs with the same volumenumber are mounted because the subfiles are then searched on the wrong volume.

Note: The `SCL FCD' command must be used to assign disc filecodes /C0 to /CF.
DAD`s can also be assigned by LKM 71.

Error Messages

```
WRONG PARAMETER
FILE CODE UNKNOWN
2ND FILE CODE UNKNOWN
EQUIVALENCE ASSIGNMENT IMPOSSIBLE
TOO MANY PARAMETERS
I/O ERROR
NOT ENOUGH SPACE IN DYNAMIC AREA
DEVICE UNKNOWN
WRONG DAD NAME
ERROR ON DISK FILE CODE
WRONG NO. OF GRANULES
DAD FILE CODE DOESN'T EXIST
DISK UNIT FILE CODE CREATION ONLY IN SYSTEM MACHINE
ERROR IN LAST PARAMETER
DAD NOT FOUND
DISK UNKNOWN
DAD OVERFLOW
WRONG FILE TYPE
WRONG FILE CODE
FILE UNKNOWN
```

For filetype EF, the following error messages may also occur:

```
    DELETE OLD F.C. ERR, ST= xxxx
    ERR IN CREATING FDC, ST= xxxx
    ASSIGN ERR, ST= xxxx
```

For the TDFM errors, an explanation of the status can be found in appendix C, LKM 23.
For a more extensive error description see Chapter 7 SCL ASG command.

Purpose

To cause the FCL task to exit and release the peripheral devices used to receive and log FCL commands.

FORMATS

```
BYE [ml[,m2]...]     from MAS 8.50:  BYE [$$,][ml[,m2]..]
```

ml, m2, etc.	are machine names. If one or more of these parameters are entered, then those machines will be started as though an SB or SM command had been given for each one after the BYE command.
$$	is an indication that open spool files have to be closed and unspooled.

Remarks

Since this command terminates the FCL task, it must be the last given. The user application continues until the last active task has exited, and the foreground machine then enters an idle state until the FCL task for this machine is reactivated using the 'SM' operator command.

Purpose

To cause the FCL task to exit and open spool files to be closed and unspooled (till MAS release 8.50) or to close and unspool one or all open spool files (from MAS release 8.50).

Format

```
CLS [ml[,m2]..]    or from MAS 8.50   CLS  [/fc|ALL]
```

ml,m2 are machine names. If one or more of these parameteres are entered, then those machines will be started

/fc is a spooled filecode which is to be closed and unspooled.

ALL indicates that all spooled filecodes are to be closed and unspooled.

Format

```
DAS  lc1,dn[da[l]]
DAS  lc1,lc2
```

lc1	is the linecode to be assigned from 0-255
dn	is the device name (see Appendix A).
da	is the device address (mandatory for AMA8 devices).
l	is the linenumber (for LSM16 and AMA8).
lc2	is the linecode to which lc1 has to be assigned by equivalence.

Remarks

The linecode is assigned by inserting it in the consecutive linecode block or by creating an alternate linecode block in the system dynamic area. Whether or not an alternate linecode block is created in depending on the value specified in the DLN command during machine declaration or on the (default) value in the CVT.

Error messages

```
LINE CODE ERR
LINE CODE NOT NUMERIC
LINECODE MISSING
INV. LINECODE
DEV NAME ERROR
DEV NAME MISSING
FOR AMA8, DEVAD MUST BE GIVEN
2ND LINECODE ERROR
TOO MANY PARAM
DYN AREA OVERFLOW
DEV. UNKNOWN
2ND LINECODE NOT ASSIGNED
DEV. ADDR ERROR
ASSIGN ERR (ST=xxxx)
```

For an explanation of the status, see Appendix C LKM 48.

Disconnect a line

Format

```
DDC  lc
```

lc is the linecode to be disconnected.

Remark

This command is translated into a Datacom call LKM 8 with order /10.
For more information about this LKM, see the Datacommunication manuals.

Error messages

```
LINECODE MISSING
LINECODE ERROR
LINECODE IS NOT NUMERIC
LINECODE TOO BIG
WHAT IS THE SECOND PARAM?
DISCONNECT ERR ST=xxxx
HALT ERR ST=xxxx
```

The meaning of the status is described in the Datacommunication manuals.

```
NO DATACOM IN YOUR SYSTEM
```

No Datacommunication devices were generated.

Format

```
    DDL  lc
```

lc	is the linecode to be deleted.

Remarks

The linecode to be deleted is removed from the consecutive linecode table or its alternate linecode block is removed from the system dynamic area.

Error messages

```
    LINE CODE MISSING
    LINE CODE ERROR
    LINE CODE IS NOT NUMERIC
    LINE CODE TOO BIG
    WHAT IS THE 2ND PARAM
```

Format

```
    DHD  lc
```

Remark

The DHD command has the same effect as a DDC command followed by a DHL command.

Error messages

The error messages returned by this command are the equal to the error messages of the DDC command.

Format

```
DHL  lc
```

lc   is the linecode to be disconnected.

Remark

This command is translated by the system into a LKM 8, order /11.
For more information, see the Datacommunication manuals.

Error messages

The error messages output by the DHL command are the same as for the DDC command

Format

```
DLF dc,fn[,ft][,fv]
```

dc	is the DAD filecode identifying the DAD in which the file exists. The file must be catalogued in the Directory for the first userid of this DAD. This DAD code must have been defined by an 'SCL FCD' command when this foreground machine was defined, by a previously given ASG command (format 5) or by a LKM 71 (assign DAD).
fn	is the filename of the file to be deleted.
ft	is the file type (e.g. SC, OB, UF or LM). Default is UF.
fv	is the version (0-7) of the file to be deleted. The default value is the latest (newest) version, i.e. version 0. The version numbers of the remaining older versions of this file are all decremented by one.

Remarks

The user can also use LKM 41 to delete a file. The specified version number must not exceed the value given on the SMV command to the LIB processor, and must exist in the Directory.

Error Messages

```
DAD F.C. ERROR
DAD F.C. NOT NUMERIC
DAD F.C. MISSING
DAD F.C. NOT ASSIGNED
INVALID DAD F.C.
F.C. NOT ASSIGNED TO A DAD
FILENAME MISSING
FILENAME ERROR
FILENAME IS NUMERIC
FILETYPE ERROR
FILETYPE IS NUMERIC
FILE NOT CATALOGUED
VERSION NBR ERROR
INVALID VERSION NBR
TOO MANY PARAM
DYN AREA OVFL
VERSION NUMBER NOT NUMERIC
```

Purpose

To disconnect programs that are no longer required from their software levels, so that other programs may be connected and activated. This command can also be used to change the software level of a program by immediately re-connecting to a different level using the 'CNL' command.

Format

```
DSL p
```

p is the program name.

Remarks

The number of programs that can be connected to software levels at any one time depends on the number of software levels specified at SYSGEN. The maximum number of programs that can be connected is n - 1, where n is the level specified for the Idle Task. The maximum value of n is 239. A program that has been activated cannot be disconnected until it has exited from the last task in its activation queue.

Error Messages

```
PROGRAM NAME NOT ALPHA.
PROGRAM NOT FOUND
PROGRAM ACTIVE, DSL IMPOSSIBLE
PROGRAM NOT CONNECTED
```

Disconnect a Program from a Timer

Purpose

To prevent further activations of a program which is being regularly activated following a 'CNT' command or LKM 10 request.

Format

```
DST p[,z]
```

p is the program name
z identifies which timer the program 'p' is to be disconnected from.
Possible values are as follows:

1 = minutes.
2 = seconds.
3 = 100 msecs.
4 = 20 msecs.
5 = non-standard timer.

In the event of the optional parameter 'z' not being specified, the default value is all the timers to which the program is currently connected.

Remarks

No further activations of the program will occur due to the specified timer(s), but the program is still known to the system and is still connected to a software level.

Error Messages

```
PROGRAM NAME NOT ALPHA
PROGRAM NOT FOUND
WRONG TIMER NUMBER
TIMER NUMBER > 5
TIMER 5 CANNOT BE USED
TOO MANY PARAMETERS
PROG NOT CNTCD TO THIS TIMER
PROGRAM NOT CONNECTED TO A TIMER
```

Purpose

To list specified sectors of a file onto the printer.

Format

```
DUF fc,sf[,sl]
```

fc is either a DAD filecode (/F0 - /FF) or a filecode assigned to a disc file.
sf is the number of the first sector to be dumped.
sl is the number of the last sector to be dumped, if omitted only one sector sf is dumped.

Error Messages

```
FC PRT NOT ASGN
DYNAMIC AREA OVERFLOW
PARAM ERROR
PARAM MISSING
DAD F.C. NOT ASGN
READ FILE I.O. ERROR
OUT OF LIMITS FILE
2ND ADDRESS LESS THAN 1ST ADDRESS
```

Purpose

To list sequentially all or part of a segment of a foreground machine.

Format

```
DUM sn,fa[,la]
```

sn	is the number of the segment containing the area of memory to be dumped to the device assigned to filecode /02. Permitted values range from zero to the number given on the SCL DCF command when defining this foreground machine.
fa	is the virtual address within segment 'sn' of the first memory location to be dumped (as a 16-bit hexadecimal number).
la	is the virtual address within segment 'sn' of the last memory location to be dumped (as a 16-bit hexadecimal number). The default value, if this parameter is omitted, is the virtual address of the last memory location allocated to the segment by the 'SCL SEG' command which defined this segment.

Remarks

Only memory assigned to memory-resident foreground programs and dynamic buffers can be dumped with this command. The 'DM' operator command can be used to dump memory assigned to disc-resident programs of a foreground machine .

The dump addresses specified must be within the given segment, otherwise the command will be rejected.

Error Messages

```
FC PRT NOT ASGN
DYNAMIC AREA OVERFLOW
PARAM ERROR
PARAM MISSING
SEG. NUMBER UNKNOWN
DUMP ADDRESS FORBIDDEN
```

Format

    EOJ

Effect

The default Userid and DAD in the Foreground machine are reset to the first Userid of DAD /FO.

Remark

The EOJ command has been implemented via a special interface. The coding of this command is stored in the load module EOJ, residing on the system userid. The command is only available in releases from MAS 8.50.

Error message

ERR ON SET USID-DAD, ST=xxxx (in decimal value, the status returned by LKM 73).

Purpose

To move an object file or a memeber from an object file to the filecode /D5 (/O file) to be an input for the Linkage Editor or the Librarian. This command is implemented from MAS release 8.50.

Format

```
INC [fc]    or

INC libr,[mnam][,usid,dadfc]
```

fc	is a filecode assigned to a device from which the object code has to be read. The device is read up to the EOF mark. The default filecode is /E2.
libr	is the name of an object library.
mnam	is the name of an object module. If not specified, the whole object library is copied to /D5.
usid	is the userid, where the object libary can be found.
dadfc	is the filecode of the DAD, where the userid resides. The defaults for usid-dadfc is the first userid of DAD /F0, or the defaults, set by a LKM 73.

Remark

The effect of the INC command is exactly the same as for the BCL INC command. The coding of the INC command is not stored in the segment file (D:MASG or D:MSEG) but is read via a special interface from a separate load module INC on the system userid.

Error messages

```
1ST PARAM (F.C. OR LIBR NAME) ERR
2ND PARAM (MNAM) ERR
3RD PARAM (USID) ERR
4RD PARAM (DAD F.C.) ERR
ERR ON INPUT F.C. ASSIGNMENT, ST=xxxx (xxxx is returned by LKM 23)
ERR ON GET INFO. ABOUT F.C. yy
ERR ON DELETE F.C. /D0
NOT DFM FILE OF TYPE OB
I/O ERR ON GRANTB READ OF INPUT FILE
ERR ON OUTPUT f.c. /D5 ASSIGNMENT, ST=xxxx (returned by LKM 23)
OUTPUT SECT. SMALLER THAN INPUT SECT.
I/O ERR ON INPUT FILE
MODULE NOT FOUND
I/O ERR ON OUTPUT FILE, ST=xxxx (returned by LKM 1)
ERR ON GET INFO ABOUT INPUT F.C.
F.C. TYPE MUST BE DFM OR PHYSICAL
READ ERR FROM DEVICE
BAD GET COMMAND.
```

Purpose

To change the default DAD and Userid in a Foreground machine. This command is implemented from MAS releae 8.50.

Format

```
    JOB       [usid,dadfc]
```

usid	is a name of a userid.
dadfc	is a DAD filecode. The default for usid-dadfc are: first userid of DAD /F0.

Remark

The implementation of this command is done via the same interface as the INC command, via the load module JOB on the system userid.

Error messages

```
    BAD GET COMMAND
    2ND PARAM NO DADFC
    WRONG PARAM
    ERR ON SET USID-DAD, ST=xxxx
```

xxxx is a decimal value, indicating the status returned from the LKM 73 (set default usid and DAD).

Purpose

To disconnect and delete a foreground program.

Format

```
KIL p
```

p is the program name expressed as ASCII characters.

Remarks

The program must be inactive or aborted. It is disconnected from its software level, which thus becomes free for use by another program. The memory areas occupied by save areas and control tables are also freed.

If the program is memory-resident (declared by a LOD or REP command), the memory it was occupying in its segment becomes free for use by other memory-resident programs.

If the program is disc-resident (declared by a RON or SWP command), the space occupied by it in the D:CI swap-image file may be re-used for other disc-resident programs.

Error Messages

These are output to the device associated with filecode /01 of the foreground machine:

```
PARAM MISSING
INVALID PROG NAME
UNKNOWN PROGRAM
MORE THAN ONE PARAMETER
PROGRAM IN ACTIVE STATE
PAGE TABLE DESTROYED
CHAIN ADDRESS IN SEG DESTROYED
```

(Memory-resident programs belonging to the same segment are chained, and this chaining address has been overwritten by the user program.)

```
SYST FILECODE /F1 INVALID
```

(The filecode /F1 is not assigned to the system DAD D:CI.)

```
PRGRAM TO BE SWAPPED OUT
PROGRAM BEING SWAPPED OUT
PROGRAM BEING ABORTED
```

(The last three error messages indicate that the program to be killed is subject of some system task. The KIL command has to be re-issued when the system task has finished)

Purpose

To release all programs loaded into a memory-resident segment.

Format

```
KIS s
```

s is the segment number.

Effect

The pages allocated to the segment are released and thus no more programs can be loaded into it.

Note: All programs in the segment must be inactive or aborted.

Error Messages

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

```
PARAM ERROR OR MISSING
UNKNOWN SEGMENT
WHAT IS THE SECOND PARAM?
PROG n ACTIVE
PROG n PAGES TAB DESTROYED
PROG n INV. CHAIN ADD. IN SG.  *
PROG n SYST. DAD /F1 INCOR.  **
```

n	is a program name.
*	Memory-resident programs within a segment are chained; the chain address has been overwritten by a user program.
**	The file code /F1 is not assigned to the System DAD D:CI.

In the case of the last 4 messages, the command processing is terminated. The programs following the one giving rise to the error are not released and the pages are not freed.

Purpose

To disconnect and free the pages occupied by a secondary load module, so that they can be reallocated.

Format

```
KLM n
```

n	is the secondary load module name, as specified on the LSM command which was used to load it.

Remarks

The command is rejected if the secondary load module is connected to a primary load module which is still active.

Error Messages

```
PARAMETER ERROR
SEC. LOAD MOD. ALREADY DELETED
SEC. LOAD MOD. STILL CONNECTED.
     (the secondary load module is still in use by a program. This program
     must first issue an LKM 57 - disconnect a secondary load module)
```

Purpose

To catalogue a disc temporary file.

Format

KPF fc,fn[,ft]

fc	is the filecode defining the disc temporary file to be catalogued. This filecode must have been assigned previously, either by an `FCL ASG' command to this foreground machine, or by use of an LKM 33 or 21 issued by a program within this foreground machine.
fn	is the file name to be placed in the first Directory of the DAD to which this file has been written. It consists of from 1 to 6 ASCII characters.
ft	is the type of file, e.g.: UF = User File (default value); OB = Object File; SC = Source File; LM = Load Module. This need not be the same as the type specified when the filecode `fc' was assigned. The type will be catalogued in the same Directory as the filename, and version number will be set to zero.

Remarks

If previous versions of this file exist (with the same filename and type) within the same Directory, the oldest version will be deleted if its version number is equal to the value specified on the SMV command to the LIB Processor for this Userid. Other versions will have their version numbers incremented by one. Usually a file is catalogued by means of the LKM 40 request, but the KPF command can be used to catalogue a valid file created by a program which terminated abnormally before the LKM 40 could be issued.

Another use of the KPF command is where a foreground application does not have a clearly defined termination. Instead of the user having to activate a special program to catalogue his files, he may issue KPF commands instead.

Error Messages

```
FILE CODE MISSING
FILE CODE ERROR
FILE CODE NOT NUMERIC
INVALID FILECODE
FILE NAME MISSING
FILE NAME ERROR
FILE NAME IS NUMERIC!
FILE TYPE ERROR
FILE TYPE IS NUMERIC!
TOO MANY PARAM
FILE CODE NOT ASSIGNED
NOT A DISK FILE
ALREADY CATALOGUED
INVALID FILE TYPE
KPF ERROR
DYNAMIC AREA OVERFLOW
I/O ERROR
DIRECTORY OVFL
KEEP FILE ERROR
NO USERID (the file had to be kept on a DAD not containing any Userid)
```

Purpose

To provide the user with the same information as that provided by use of the PRS command for all programs in his foreground machine, i.e.:

Program Name.
Program Type:
- SWP = Swappable;
- REE = Re-entrant;
- RON = Read-only;
- Snn = Loaded into segment nn.

Software Level.
Active state.
Whether loaded, and if so the address.
Whether swapped out or not.

Format

```
MAP  fc
```

fc	is the file code assigned to the device on which the map is to be printed. The default value is /01.

For an example of a MAP printout, see Chapter 7, the MAP Command.

Error Messages

```
INVALID PRINT FILECODE
PRINT F.C. ASSGN TO NO DEVICE
```

Purpose

To declare a node in the temporary object file as input for the Linkage Editor.

Format

```
NOD       nodename[,rovs| [,a]]
```

nodename	is a 6 ASCII character name, conforming the same rules as any external program name or label. This is copied to the temporary object file /D5) as a card image, to be processed by the Linkage Editor. It defines a branch in an overlay tree.
rovs	is a 4 character segment name, referring to a secondary load module name. This must not duplicate any other entry point, common block or segment name.
*	is a signal to the Linkage Editor to implicitly name the segments in this compilation in ascending order, in accordance with the parameters in the LKE option command.
a	is an absolute address in hexadecimal of a page boundary. It forces the Linkage Editor to load the module at this address

Remark

FCL copies the command to the temporary object file /D5. It will be processed later by the Linkage Editor. The NOD command has been implemented via a special interface, so that its coding is not in the MAS segments file D:MASG or D:MSEG) but in a separate module NOD on the system userid.

Error messages

```
ERR ON /D5 ASSIGNMENT, ST=xxxx  LKM 23 status, printed in decimal)
ERR ON GET INFO ABOUT /D5
WRITE ERR ON /D5
BAD GET COMMAND
```

Purpose

To print a map of the program layout in a user machine.

Format

PCM [fc]

fc	is the print filecode; the default value is /01.

Error Messages

```
INVALID PRINT FILECODE
PRINT F.C. ASSGN TO NO DEVICE.
```

Purpose

To obtain a list of all filecodes assigned to this machine.

Format

PFC [fc]

fc	is the filecode assigned to the device on which the filecodes are to be printed. The default value is /01.

Error Messages

INVALID PRINT FILECODE
PRINT F.C. ASSGN TO NO DEVICE

Purpose

To obtain a list of all the linecodes, which have been assigned in the Foreground machine.

Format

```
PLC       [fc]
```

fc is a filecode assigned to a device on which the linecodes are printed. The default value is filecode /01.

Error messages

```
PRINT FILECODE NOT ASSIGNED
PRINT F.C. ASSIGNED TO NO DEVICE
```

Purpose

To obtain a list of all the software levels which are in use i.e. connected) in the bare machine.

Format

```
PLV [fc]
```

fc	is the filecode assigned to the device on which the software levels are to be printed. The default value is /01.

Remarks

For each level the system gives in reply:
The machine name.
The program name.
The program type.
Whether or not the program is active.

Error Messages

```
INVALID PRINT FILECODE
PRINT F.C. ASSGN TO NO DEVICE
```

Format

```
PRG  p,fc,{A | M | S}
```

p	is the program name 1-6 ASCII characters).
fc	is the filecode assigned to the printer on which the listing is to appear.
A	is a code indicating that the contents of all registers are to be printed both main sequence and scheduled label).
M	indicates that only main sequence registers are to be listed. Word 5 of the PCT is used to find the save area for these registers.
S	indicates that only scheduled label registers are to be printed. Word 26 of the PCT is used to locate the save area for these registers.

Remarks

Only one of the symbols A, M or S must be entered; this field is mandatory.

Error Messages

```
INVALID PROGRAM NAME
PROG NAME MISSING
PROG NAME TOO LONG
PROG NAME UNKNOWN
INVALID PRINT F.C.
PRINT F.C. MISSING
PRINT F.C. NOT ASSIGN
INVALID PRINT CODE
PRINT CODE MISSING
PRINT CODE TOO LONG
PRINT CODE UNKNOWN
```

Format

```
PRS p[,fc]
```

p	is the program name 1-6 ASCII characters).
fc	is the filecode of the print device on which the status is to be printed. The default value is /01.

Remarks

The following information about the program `p' is output to the specified printer:

```
        Program name.
        Program type:
                  SWP = swappable
                  REE = re-entrant
                  RON = read-only
                  Snn = loaded into segment `nn'
        Software level.
        Whether active or not.
        Whether or not it is loaded, and if so the address.
        Whether swapped out or not.
```

Error Messages

```
PARAM ERROR
PROG NAME MISSING
PROG NAME UNKNOWN
PRINT F.C. PARAM ERROR
PRINT F.C. BAD ASSGN
INVALID PRINT FILECODE
PRINT F.C. ASSGN TO NO DEVICE
```

Purpose

To suspend a foreground program. The program can be re-started using the 'FCL RST' command.

Format

```
PSE p
```

p is the program name 1-6 ASCII characters).

Remarks

This command gives the user similar facilities to those provided by the 'PS' operator command.

A Middleground program may be suspended between the 'RUN' command and its subsequent exit.

A re-entrant program cannot be suspended, since MAS cannot identify the appropriate PCT.

Error Messages

```
PARAM ERROR
PROG NAME MISSING
PROGRAM UNKNOWN
PROGRAM INACTIVE
PROGRAM ALREADY IN PAUSE
SYSTEM PROGRAM
```

Clear the Abort State of a Program

Format

```
RAB p
```

p   is the program name 1-6 ASCII characters).

Remarks

When a foreground program is aborted it remains in the foreground machine, but the abort flag in the PCT word 7, bit 1) is set and subsequently the program can only be activated by an `FCL CNT', an 'FCL RUN' or `FCL ACT' command.

The LKM's 10 or 12 and activations because of a previous CNT command are ineffective. The RAB command allows the user to reset the aborted foreground program.

Error Messages

```
PARAM ERROR
PROG NAME MISSING
PROG NAME UNKNOWN
PROG INACTIVE
PROG NOT ABORTED
```

Purpose

To allow the foreground user to cancel an I/O operation which did not terminate and caused a `PU' (Physical Unit) message to be output. This `PU' message must have ended with `RY' Retry), otherwise a release is not allowed.

Format

RDV da

da is the device address given in the preceding `PU' message.

Remarks

The `RDV' command provides the user with similar facilities to the `RD' operator command.

Error Messages

```
PARAM ERROR
DEVICE ADDRESS MISSING
DEVICE ADDRESS UNKNOWN
DEVICE NOT IN RETRY
```

Format

```
RST p[,n]
```

p	is the name of the program to be restarted.
n	is a value to be placed in A7 when the program is restarted. The default value is zero.

Remarks

This command gives the user similar facilities to those provided by the 'RS' operator command.

Error Messages

```
PARAM ERROR
PROG NAME MISSING
PROG NAME UNKNOWN
BAD A7 PARAM VALUE
PROG NOT IN PAUSE
```

Purpose

To make another attempt to execute an I/O operation, when a previous attempt was unsuccessful but a retry was invited by a system 'PU' message that ended in 'RY'.

Format

```
    RYD da
```

<u>da</u> is the device address given in the preceding 'PU' message.

Remarks

A 'PU' message ending in 'RY' must have been output before attempting a retry command.

This command provides the user with similar facilities to those provided by the 'RY' operator command.

Error Messages

```
    PARAM ERROR
    DEVICE ADDRESS MISSING
    DEVICE ADDRESS UNKNOWN
    DEVICE NOT IN RETRY
```

Purpose

To de-assign a particular filecode from the device or file to which it is currently assigned.

Format

```
SCR fc
```

fc	is the filecode which is to be removed from the filecode table for this foreground machine.

Remarks

If the filecode is assigned to a disc file, the memory used by its blocking buffer is released.

If the filecode is assigned to a disc temporary file and no LKM 40 (Keep File) or `SCL KPF' command has been given for the file, the granules allocated to it are released.

This command cannot be used to scratch a DAD filecode.

Error Messages

```
FILECODE MISSING
FILECODE ERROR
FILECODE IS NOT NUMERIC
FILECODE TOO BIG
I/O ERROR
SCR ERROR
DELETE ERROR
DAD F.C.
```

Format

```
TIM
```

Remarks

The time is output to the physical device assigned to filecode /01.

If the timer does not contain a binary value which can be converted into a time expressed as HH/MM/SS, where:

HH is a number of hours in the range 0-23,
MM is a number of minutes in the range 0-59, and
SS is a number of seconds in the range 0-59,

then the error message:

```
THE DATE IS DESTROYED
```

is output to the device assigned to filecode /01.

Purpose

To modify memory-resident segments including Segment 0) of a foreground machine.

Format

```
WRM s,a,v1[,v2[,v3]...]
```

s	is the segment number within this foreground machine containing the locations to be changed.
a	is the relative address of the first location to be modified, expressed as decimal or hexadecimal to a maximum of 16 bits.
v1, etc.	are the new values which will be inserted in locations a onwards,starting at location 'a'.

Remarks

The WM or CR operator commands may be used to modify memory occupied by disc-resident programs of a foreground machine.

Error Messages

```
WRONG PARAMETER
WRONG SEGMENT NUMBER
WRONG LOCATION
YOU ARE TRESPASSING
WRONG VALUE
NO VALUE
```

Processor call

Purpose

To RUN a middleground program, without the necessity to type the RUN command.

Format

ccc

ccc	is the name of a load module residing in the system userid.

Remark

The effect of the processor call is the same as the effect of the RUN command of a middleground program.

APPENDICES

INTRODUCTION

This System Generation process tailors MAS to the user's requirements for his particular P800 configuration, and is an important preliminary to the successful use of MAS. A large number of variables may be specified at system generation.
The variables concern:

- declaration of the devices connected to the P800 machine with their device addresses, their interrupt levels and some other characteristics.
- specification of the system machine.

GENERAL

The MAS System Generation Package is distributed on magnetic tape or on disc. When distributed on magnetic tape, the tape contains a copy of a disc with the System Generation package.

The generated system can be stored on any of the following disc types:

X1215
X1216
CDC-SMD 40mb
CDC-SMD 80mb
CDC-CMD.

The minimum configuration required for generating a MAS system (excluding the device from which the input generation package will be read) is:

- one of the disc storage units listed above (two is preferred)
- an operator`s console
- a P800 computer with at least 64K words of memory.

Summary of the Generation Process

The generation process can be broken down into 6 distinct phases, listed here and described fully under the appropriate headings:

- Creating the Starter Pack
- Creating the System Pack
- Creating the Configuration File
- Generating the MAS System Modules
- Assembling and Cataloguing the MAS Tables
- Link-editing the System.

CREATING THE STARTER PACK

If a MAS system is distributed on disc, the starter pack resides on that disc and can be used immediatily for System Generation.
If a MAS system is distributed on tape, a copy of a starter pack will reside on that tape and has to be stored on a disc.
This restoring is done by making an IPL from the tape. A stand alone program is then loaded to restore the information from the tape to a disc, which is identified by the following questions:

```
    DISK PHYSICAL ADDRESS (2 HEXA CHAR):
```

Reply the device address of the unit whereon the starter pack to be created has been mounted.

Now the Volume Label of the mounted disc is listed to prevent writing of the tape information to the wrong pack and the program asks:

```
    IS THIS THE DISK YOU WANTED? (YES OR NO)
```

Reply YE(S) if it is the wanted disc and NO if the wrong disc was mounted. If NO is given, the restore program stops.

Now the volume label and the creation date of the tape is listed, so that a check can be made, whether the right tape has been mounted, followed by the question:

```
    DO YOU WANT TO KEEP IT
```

Reply YE(S) OR NO. IF YES, the label and the packnumber of the disc stored on the tape are written to the disc and the restoring starts. If No is replied, the restore program asks for a new label (16 char) and packnumber (4 char) to be stored on the disc before starting the restoring.
When the restore is ready, the message:

```
    END OF RESTORING
```

is output and the starterpack is ready to be used.

CREATING THE SYSTEM PACK

Step 1 An IPL (Initial Program Load) is made with the starter pack as the load device. The method is described in Chapter 4, Operation.

Step 2 Give the replies requested by the system to the following messages:

1. CPU:
   Reply: 57, 58, 59 or 54, specifying the CPU type of the machine where the System Generation will take place.
2. DK1:
   Reply Format: /C0,disc,da,i
3. DK2:
   Reply Format: fc,disc,da,i
4. CR:
   Reply Format: either - da,i
   or - NO
5. LP:
   Reply Format: either - da,i
   or - NO

where	disc	=	1215, 1216, 40M or 80M.
	fc	=	disc filecode (/C1 - /CF).
	da	=	device address.
	i	=	interrupt level expressed in hexadecimal with preceding /. (Typical values: discs = /11, LP = /17; these are set by the engineer.)

Note: When creating a system using only one disc pack (only possible for CDC discs), question 2 above should be answered with SAME.

Step 3 Premark the System Disc Pack (if it is necessary), as follows:
Place the system disc pack on the disc drive (DK2).
Give the SCL BYE command to free the console.
Press the control panel interrupt button. The system will reply:

```
M:
```

Enter the command: PK Cx (Cx is the fc parameter from DK2 question)

The Premark procedure is then followed as under Operator Commands in Chapter 5, with the following advised parameter values:

DAD name: 'SUPERV'
Sectors per granule: 8
Userid: 'MASUP'.
Number of cylinders:
- $150 \leq n \leq 200$ for X1215
- $150 \leq n \leq 400$ for X1216
- $50 \leq n \leq 100$ for CDC.

Number of interlaces:
- 10 for CDC
- 5 for X1215/6 for efficiency
- 3 for X1215/6 for DOS compatibility.

For CDC, 2 extra questions are asked:
- SECTORS PER TRACK? - reply 39
- SECTOR LENGTH? - reply 410.

Note: If the user requires a premark date on the volume label, the date must be entered before the Premark is started. This can be done either by SCL/FCL or with an operator command. This date and time setting is, of course, only neceesary for those MAS release which do not ask the date and time before starting the FCL task of the system machine.

Step 4 Define a Background Machine using one of four catalogued procedures. The one chosen depends on the available peripherals; they are listed here together with the filecodes which will be assigned:

Procedure	Filecode /E0	Filecode /02
%%B2TY	TY	TY
%%B2LP	TY	LP
%%B2CR	CR	TY
%%B2CRLP	CR	LP.

Each of these catalogued procedures requires the following mandatory parameters in the invocation command:

```
%%B2xx dd,mm,yy,DSK2 = /Cx[,DADO = n]
```

where: dd, mm, yy is the date, not necessary for MAS releases asking for it.
n is the DAD name (the default name is SUPERV)
and /Cx is the filecode of the system disc, as specified in DK2. When generating with only one disc pack, /C0 should be specified.

Note: If the premark is attempted after the Batch machine has been declared and the system disc was on-line during this declaration, the user must re-IPL.

Note: The DSK2 parameter is set to /C0 omitted if only 1 disc pack and 1 DAD are being used.

Step 5 Start the background machine using the SCL command:

```
BYE BATCH
```

Step 6 Start a Job (with USID=SYSTEM) to create the DAD named D:CI on the system disc, using the catalogued procedure %%DCI (for a X1215/6 disc) or %%DCICDC (for a CDC disc). This last procedure also creates a DAD called D:MSEG for the MAS segments.

The invocation command for these procedures contains two parameters, DK (the filecode of the system disc) and CY (the number of cylinders required for the DAD). These are substituted for the DISC and NCYL parameters, respectively, in the LIB command DCD (declare DAD), which this catalogued procedure uses to create the DAD. If no parameters are entered when the procedure '%%DCI' is invoked, the following default values will apply:

Keyword Parameter	%%DCI Default	%%DCICDC Default
DK	/C2	/C6
CY	50	20

Step 7 Copy the starter pack system files onto the system pack. This can be done using the BCL catalogued procedure '%%CRESYS', using the following calls:

```
:JOB      USID = SYSTEM
%%CRESYS [IUSI=u1][,OUSI=u2]
where     u1 = Userid of the starter pack (default = MASUP)
          u2 = Userid of the system pack (default = MASUP).
```

These are the names given to the system USERID when the packs were premarked.

The following steps (8 - 11), must use the Userid MASGEN.

Step 8 Create the Configuration File; this can be done by usig the catalogued procedure %%CONGEN.

It is strongly recommended that the configuration file is created by using this catalogued procedure, and that LIB and UPD processors are used only to update an existing file. It is the reason for the fact that the description of the used System Generation macro's is deleted from this Manual.

%%CONGEN consists of a series of questions, to which answers must be given in order to create the configuration file. These answers can be given conversationally, or they can have been set up as a sequential file. In the latter case, syntax errors in the answers can be corrected from a console.

The invocation command has two formats:

1) Answers to be supplied from non-disc device:

```
%%CONGEN [CONF=fn][,IN=dn[da]]
```

fn consists of 1-6 ASCII characters, specifying the name to be given to the configuration file, which will be catalogued under USID=MASGEN. The default value for CONF is CONFIG. If this filename is not CONFIG, the user must remember to specify it on the CONF= keyword parameter of the %%GENMAS.

dn is the device name, for example, TY, CR, PR, TK, MT. Not a disc device. Default is TY.

da is the device address; two hexadecimal digits without a / character, representing a 6-bit address. The default is the address of the first device of type dn in the system tables:

TY	10
CR	06
PR	20
TK	05
MT	04

If conversational mode is to be used, the IN parameter should not be specified. Questions and answers then appear on TY10.

2) Answers to be supplied from a disc file:

```
%%CONGEN CONF=fn,FNAM=fn2[,USID=u[,DAD=d]]
```

fn is the filename of the configuration file, as for the first type of invocation command.
fn2 is the name of the disc file containing the answers.
u is the USERID (default value is MASGEN).
d is the DAD (default value is /F0).

Syntax Rules
For any question ending with `?' the allowed replies are:
Y[ES] carriage-return
N[O] carriage-return
or carriage-return (assumed to be NO).

For any question ending with `:' the reply is one or two parameters separated by a comma. The first character of each character string must be alphabetic. Number strings must be decimal (or hexadecimal, if preceded by a slash).

If a syntax error occurs a meaningful error message is printed. No action is taken, but the parameter in error may then be re-input irrespective of the medium used for the conversational input.

The Conversation

If the conversational mode is selected the message:
    SELECT MONITOR   (this question is not asked in every release)
is printed. The user replies:
    MAS
then the message:
    CONTROL PANEL INTERRUPT LEVEL   (not asked for every release)
is printed. The user replies the interruprt level, normally: 07.
Then the message:
    SPECIFY CPU TYPE
is printed. Reply:
    P857, P858, P859 or P854 identifying the CPU where the system to be generated should run.
The System Generation dialogue starts with:
    SPECIFICATION OF DEVICES
followed by a serie of quesions which the user is required to answer.
From MAS 8.50, first is asked:
    DO YOU WANT A LIST OF DEVICES
reply Y(ES) or N(O). If N(O), the DEV: question is output, if Y(ES), a list of devices that can be generated is printed. For this list, see below.

The Device Macro Phase

DEV:

This is the first question, the reply to which specifies one device. The question is then repeated and the second device specified, and so on. When no more devices are to be specified the reply EN[D] should be given.

Reply Format

    dnda,lv

dn is the name of the device to be generated. The names are:

CC	CDC-SMD disc 40mb or 80mb
CD	X1215/X1216 disc
FH	DDC fixed head disc
FL	0.25mb floppy data disc
MT	magnetic tape
TK	cassette
LP	lineprinter
CR	cardreader
TY	typewriter
PR	papertapereader
PP	papertapepunch
DY	display device
PL	plotter
DM	AMA8
MF	1mb floppy disc
DF	CDC-CMD disc
AS	ASCU4Z

When all devices have been specified, EN has to be typed.

da is the device address. For devices connected to the IOP the first device address must be specified.

lv is the device interrupt level, a value ranging from 4 to 61.

Error messages

```
ADDRESS OUT OF RANGE
ILLEGAL DEVICE NAME
ADDRESS ALREADY ASSIGNED
INT LEVEL OUT OF RANGE
INT LEVEL ALREADY ASSIGNED
FIRST DEVICE OF CONTROLLER PLEASE
4 CONSECUTIVE ADDRESSES MUST BE FREE   (for ASCU4Z)
ADDRESS MUST BE A MULTIPLE OF 4   (for ASCU4Z)
LEVEL OF INPUT INT OF AMA8 MUST BE EVEN   (for device DM)
LEVEL OF OUTPUT INT OF AMA8 ALREADY ASSIGNED   (for device DM)
DEFINE AT LEAST ONE DISC FOR SUPERVISOR   (EN given and no disc specified)
```

For some devices, device dependent questions are output:

CDC-SMD disc (device name is CC)
   DISC TYPE (40, 80, 150, 300)
Reply: 40 or 80 being a 40mb or 80mb disc. The replies 150 and 300 are reserved for future implementation.

X1215/X1216 disc (device name is CD)
   DISC TYPE (1215, 1216)
Reply: 1215 (X1215 2.5mb disc) or 1216 (X1216 5mb disc).

CDC-CMD disc (device name is DF)
   DRIVE x?
Reply: YES or NO specifying whether drive x (0-3) is present or not.
   TYPE OF FIXED PART? SIZE (16,48,80)
Reply: 16, 48 or 80 indicating the fixed part of the drive.

Cassette (TK) and 0.25mb floppy (FL)
   PROGRAMMED CHANNEL?
Reply: YES, device connected to programmed channel, or NO, device connected to IOP.

Lineprinter (device name is LP)
   PROGRAMMED CHANNEL?
   NUMBER OF LINES/PAGES
Reply: a number from 10 to 99

Typewriter (device name is TY)
   NUMBER OF LINES/PAGE
Reply: a number from 10 to 99 or 0. Zero indicates no Top-of-form skipping has to be done.

AMA8 (DM) and ASCU4Z (AS)
   NUMBER OF LINES
Reply: 1-4 for ASCU4Z and 1-8 for AMA8
   SAME DEFINITION FOR ALL LINES?
Reply: YES or NO. If YES, the line characteristics are only asked for the first line, if NO they are asked for every line.

LINE NUMBER?
Reply: a number from 0 to 3 for ASCU4Z and from 0 to 7 for AMA8. The linenumber specification need not to be in ascending order and identifies which lines are connected.
DEVICE TYPE (DY, TY, LP, TK)
Reply: TY, DY, LP or TK depending on the type of device connected to this line.
NB OF CHAR PER LINE
Reply: any positive value. This question is not output for a TK device.
NB OF LINES PER PAGE
Reply: any positive value. This question is not output for a TK device.
ECHO MODE?
Reply: YES or NO. This question is only output for AMA8, devices TY and DY.
RESTART TIME FOR PAGE HANDLING (0 TO 255 MN)
Reply: a value from 0 to 255, only output for a DY device. The page handling prevents lines racing over the screen without any possibility to read it for normal human beings. After n-2 lines in standard write mode (n being the number of lines per page) a message is written on the last line of screen to show the next possibilities:
- When the user inputs an 'S' the current standard writes are stopped and considered as ended. Any subsequent standard write is ignored, until a read or basic write has been executed.
- With any other character input, the standard writes are continued as if nothing has happened.
- Without inputting any character, the standard writes are continued after the number of minutes, specified in this Sysgen parameter.
TIME OUT (0 TO 255 MN)
Reply: a value from 0 to 255.
PARITY (N=NOPAR, O=ODD, E=EVEN)
Reply: N, O or E.

IOP devices

DEVICE xx ON THIS CONTROLLER?
Reply: YES or NO. The question is repeated if more than two devices can be connected to the controller, until the reply is NO or 4 devices have been declared.

Error messages On erroneous input on a device dependent question a self-explanatory error-message is printed.

When 'EN(D)' is entered to terminate device declaration, the system asks:

USER INT:

This requests the interrupt level to which any user written interrupt routines are to be linked.

Reply Format

lv,n   or   EN

lv	is the user interrupt level, in the range 4 to 61.
n	is a 4-character name which, when concatenated with 'I:', becomes the entry point label of the user interrupt routine. The user must be careful to choose a name not already in use. If the object code of his routine is kept in a MAS object library, it will be included automatically at link-edit time. This is an alternative method of specifying a user interrupt entry in the LOCAT module to that described in Appendix F. To include his DWT, the user may update the routine provided under the name USDWT, which is delivered on the starter pack.
EN	signifies the end of replies to the DEVICE macro phase.

The SYSMAC Macro Phase

When EN(D) is given as a DEV reply it indicates that all devices have been defined. Definition of the MAS options resident in the system machine is now begun with the message:

```
    DEFINITION OF SYSTEM MACHINE
```

The following is an example of the dialogue to define these options:

```
    STACK SIZE (/200 (NORMAL) TO /800): (specify /200. Another value needs
                                         regeneration of all MAS overlay
                                         segments)
    /200
    RTC FREQUENCY (50, 60)
    50
    NB OF CHAR FOR SYSTEM DYN AREA :   (from /800 to /7FFE, or 0; 0 means give
                                        maximum at IPL time).
    100
    DYN AREA SIZE OUT OF RANGE    error
    NB OF CHAR FOR SYSTEM DYN AREA :
    0                                    (the maximum, = up to 32kw)
    NB OF SOFTWARE LEVELS (MULT OF 30) : (from 30 to 240)
    12
    NB OF PROG OUT OF RANGE       error
    NB OF SOFTWARE LEVELS (MULT OF 30) :
    31
    MULTIPLE OF 30, PLEASE        error
    NB OF SOFTWARE LEVELS (MULT OF 30) :
    120
    DUMP STAND ALONE ?  (not asked for systems running in extended mode)
    N
    POST MORTEM DUMP IN BATCH ? (* see below, not asked for extended mode
    N                             systems)
    FLOATING POINT PROCESSOR ?
    Y
    TDFM ?
    Y
    MAX NB OF BUFFERS FOR TDFM :
    60
    OUT OF RANGE NUMBER           error
    MAX NB OF BUFFER FOR TDFM : (from 1 to 49, advisable is a multiple of 3)
    12
    NETWORK ACCESS?    (reply YES, only if AMS is incorporated in the system)
    N
    CORE LKM OPTION?   (reply YES, if some disc resident LKM's should be core
                        resident, because they are used very often)
    Y
    LKM 17 CORE RESIDENT?     (get date and time)
    Y
    LKM 52 CORE RESIDENT?     (send/receive letter)
    N
    LKM 55 CORE RESIDENT?     (semaphore)
    N
    LKM 56 CORE RESIDENT?     (request/release pages)
    N
    LKM 57 CORE RESIDENT?     (connect/disconnect secondary load module)
    N
```

*   If this command is answered by NO, LKMs 45 and 53 will not work.

If Dump Stand Alone is selected, the whole memory will be dumped in the event of a fatal error. This facility adds about 500 words to the size of the resident part if the generated MAS does not run with extended mode. If it is not selected, the system will halt on encountering a fatal error, with an error code in A1.

When the definition of the system machine is complete, the message:
    DEFINE SYSTEM FILE CODES
indicates that all the system filecodes must be specified by the replies to the following questions:

FCD:
The reply to this specifies one system filecode. The question is then repeated and the second filecode specified, and so on.When no more filecodes are to be specified the reply EN(D) should be given.

Reply Format
    filecode,dnda

filecode	is a valid filecode in the range /01 to /CF. Minimally all connected disc units must be defined by a filecode (/C0-/CF), the filecodes /01 and /EF must be assigned to an interactive device, filecode /E0 must be assigned to an input device and filecode /02 must be assigned to an interactive or an output device. If the user wants to utilize LKM 50 (submit a Job to the Batch machine, some filecode must be assigned to the CR, whether or not this device is physically present or not.
dnda	are as explained in the replies of the DEV question.

Error Messages
    FILE CODE OUT OF RANGE (from /01 to /CF)
    ILLEGAL FILE CODE (disc filecode for non-disc device)
    FOR DISC, FC RANGE FROM /C0 TO /CF
    DEFINE A DISC FC FOR SUPERVISOR
    UNKNOWN DEVICE
    FILECODE ALREADY ASSIGNED
    DISC ALREADY ASSIGNEDDEFINE FC /EF FOR OPERATOR CONSOLE
    DEFINE FC /E0 FOR INPUT COMMAND

Following an FCD reply specifying a device name LP, PP, PL or CR the question:
SPOOLED DEVICE?
is asked, but only when the concerned device name was specified for the first time.

Reply Format
    YES if spooling is to be used on this device.
    NO   if spooling is not planned for this device.

NUMBER OF STOP BIT (NORMAL VALUE=2):

Reply format
    1 or 2. This question is output when filecode /EF is specified.

For a filecode assigned to an AMA8 device the question:

LINE NUMBER
is output, defining to which AMA8 line the filecode must be assigned.

Reply format
    0 to 7. The line must be defined in the device specification phase.

For CDC-CMD devices (device name is DF) there is asked for the drive number and the part:

DRIVE NUMBER

Reply format
    a number from 0 to 3.

FIXED PART?

Reply format
    YES (fixed part) or NO (removable part)

NAME OF SUPERVISOR DAD:
    The reply to this specifies the name of the system DAD (e.g. SUPERV).

Reply Format
    Up to six ASCII characters.

FILECODE OF DISC:
Enter a filecode in the range /C0 to /CF, previously specified using the FCD commands. This specifies the disc filecode (normally /C0) on which the system DAD is located.

Error Messages
    DISC FILECODE OUT OF RANGE
    FILE CODE NOT ASSIGNED

SWAPPABLE PROGRAMS?
    The reply to this specifies that the user intends to use swappable (or middleground) programs.

Reply Format
    NO   if there are no swappable programs.
    YES if there are swappable programs. The following question is now asked:

FOR SWAP DAD, GIVE NOW

FILE CODE OF DISC:

Reply Format
The reply format is identical to that of the previous question: FILE CODE OF DISC:. It specifies the disc filecode on which the DAD D:CI will be set up.

The DTC Macro Phase

DO YOU WANT DATACOM? (Y/N)
If no Datacom device is used, the reply:
    N[O]
is given and the configuration definition is now complete. Otherwise, the reply:
    Y[ES]
leads to the following dialogue:

TWO SYSTEM GENERATIONS ARE AVAILABLE
WHICH SYSTEM GENERATION DO YOU NEED?
TMS OR DATEM2 (TMS=YES, DATEM2=NO)

Reply Format
    {Y | N}

Y	TMS is generated
N	DATEM2 is generated.

The following dialogue is different for each controller type and also different for DATEM2 and TMS. As the questions appearing in DATEM2 are all contained in the TMS dialogue, only a description of the TMS dialogue is given here.

To keep the description simple, first all questions with their possible answers and their error messages are given, followed by the sequence of questions occurring per device. Defaults on questions only exist when the answer is YES or NO. In that case the default is always NO. Whenever YES or NO has to be given, Y or N suffices.
For each controller, the dialogue starts with:
    CHOOSE YOUR CONTROLLER TYPE

Replies:	SA = SALCU	TMS only
	S2 = SLCU2	TMS & DATEM2
	S4 = SLCU4	TMS & DATEM2
	A2 = ALCU2	TMS & DATEM2
	A4 = ALCU4	TMS & DATEM2
	A8 = AMA8	TMS & DATEM2
	L6 = LSM16	TMS only
	H1 = HDLC	TMS only
	HV = HLVCU	TMS only
	SZ = SLCUZ	TMS only
	EN = all controllers have been specified.	

For DATEM2 the controller type has to be specified on the question DTC: the reply format is dnda,level
Error message: Illegal device name
    HOW MANY LINES DO YOU WANT?
Reply 1-8 for AMA8 and 1-16 for LSM16.
Error message: Illegal number of lines.
    ADDR:
Reply a value from 1 to 63. The value can be one or two decimal characters or one or two hexadecimal characters with a preceeding '/'.
Error messages: Address out of range
                Address already assigned
                Address of the device must be even
                Address of output line already assigned
    INT. LEVEL
Reply a value from 4 to 61.
Error messages: Int level out of range
                Int level already assigned
                Level of input int of AMA8 must be even
                Level of output int of AMA8 already assigned
    FULL DUPLEX?
Reply YES (full duplex) or NO (half duplex)
    PROGRAMMED CHANNEL?
Reply YES (programmed channel) or NO (IOP)
    DATEM2 STATUS MODE?
Reply YES or NO. YES means that the status values returned by TMS are compatible with the ones returned by DATEM2. No means they are not.

SWITCHED LINE?

Reply YES (switched line) or NO (leased line)

NB OF BITS/CHAR (5, 6, 7 OR 8):

Reply the number of bits per character.
Error message: Out of range number

CONNECTION MODE:
1=CT108/1, 2=CT108/2

Reply 1 or 2.
Error message: Illegal value.

PARITY?:

Reply 'O' (odd), 'E' (even) or 'N' (no parity).
Error message: Illegal value (N, O or E)

ASCII MODE?

Reply YES (ASCII) or NO (EBCDIC). The reply specifies which kind of characters is to be handled.

REDUNDANCY CHECK:

Reply 00 (or 0), 01 (or 1), 10 or 11, defining the CRC check.
Error message: Illegal parameter (00, 01, 10, 11)

HIGH FREQUENCY?

Reply YES or NO.

CARRIER ALWAYS ON?

Reply YES or NO.

PRECEEDING DEVICE IS FULL DUPLEX,
DEFINE NOW OUTPUT LINE

This is only a message, no reply is required. The message is followed by the 'ADDR:' and 'INT LEVEL:' questions.

SYNCHRONOUS MODE?

Reply YES (asynchronous mode) or NO (synchronous mode).

INHIBIT CHARACTER SYNCHRO?

Reply YES or NO.

AUTOMATIC SYN GENERATION?

Reply YES or NO.

HARDWARE CONTROL CHAR DETECTION?

Reply YES or NO.

FOR THE OTHER LINES DEFINE HALF OR FULL

This is only a message, no reply is required. The message is followed by the 'FULL DUPLEX' question.

INHIBIT CONTROL CHAR DETECTION IN OUT?

Reply YES or NO.

ITB PROCESS?

Reply YES or NO.

NO ACTIVITY TIMER VALUE
NO TIMER REPLY 0
FOR 650 MS REPLY 1
FOR 1.3 S REPLY 2
FOR 2.6 S REPLY 3
FOR 5.2 S REPLY 4
FOR 10.5 S REPLY 5
FOR 21 S REPLY 6
FOR 42 S REPLY 7
GIVE YOUR CHOICE:

Reply a value from 0 to 7.
Error message: Illegal value.

DOUBLE FREQUENCY?

Reply YES (double) or NO (SPLE)

SAME DEFINITION FOR ALL LINES?

Reply YES or NO. If YES is specified, it is assumed that all lines have the same characteristics. If NO, the characteristics for each line are asked separately.

NB OF STOP BIT (1, 2, 3 (FOR 1.5)

Reply 1, 2 or 3.
Error message: Out of range number.

DO YOU WANT HDLC PROCEDURE?

Reply YES or NO. IF YES the HDLC procedure is included in the TMS system, if NO, it is not.

HDLC procedure

The next questions are specifically for the HDLC procedure. The HDLC procedure can only be included for the HDLC and HLVCU controller in full duplex.
The dialogue starts with a print out of the default values:

DEFAULT VALUE:
K=NRT=7
LOCAL ADDRESS=01
MAX LENGTH=256
REMOTE ADDRESS=03

Then the questionning starts:

DEFAULT VALUE (YES OR NOT)

Reply YES or NO. If YES, the generation proceeds with the 'MAX NUMBER OF BUFFERS' question and the values printed above are assigned. If the reply is NO, the generation process asks for each parameter its value.

CURRENT VALUE OF CONSECUTIVE TIME OUT, NRT:

Reply a value from 0 to 7, specifying the value of the consecutive time out retransmission.
Error message: Out of range number.

MAX NUMBER OF OUTSTANDING FRAMES K

Reply one digit from 0 to 7.
Error message: Out of range number.

LOCAL STATION ADDRESS:

Reply two decimal or hexadecimal (with preceeding '/') digits, specifying the local staion address.
Error message: Out of range number.

REMOTE STATION ADDRESS:

Reply two decimal or hexadecimal digits.
Error message: Out of range number.

MAX LENGTH OF USER BUFFER:

Reply the maximum length of the user buffer from 0 to 4096 (decimal).
Error message: Out of range number.

MAX NUMBER OF BUFFERS GIVEN WITH OPEN REQUEST:

Reply a digit from 2 to 5.
Error message: Out of range number.
Now the default for each timer is printed. For the meaning of the timers, see the concerning questions. For all timers, the default value is /5A (*0.1 sec).
Then the generation proceeds with:

DEFAULT VALUE? (YES OR NOT)

Reply YES or NO. If YES, the HDLC procedure generation is ended, if NO, the value of each timer is asked:

TIME OUT VALUE IN NORMAL OPERATING STATE (*0.1S):
TIMER T1, (WAIT FOR F BIT AFTR TRANSMISSION OF P BIT):
TIMER T2, TIME OUT VALUE (WAIT FOR ACK) (*0.1S):
TIMER T3, TIME OUT VALUE (CHECK BUSY) (*0.1S):
TIMER T4, TIME OUT VALUE (VALUE OF RESET) (*0.1S):

All timer questions must be answered with a decimal value between 0 and 255 (0 and 255 excluded) or a hexadecimal value between /00 and /FF (with preceeding '/').
Error message: Out of range number.

Question sequence per device

SA - SALCU

-Addr:
For SALCU the address must be even. For the output line, an implicit address is assigned, one higher than the one for the input line.
-Int level:
-Full duplex?
-Programmed channel?
-Datem2 status mode?
-Switched line?
-Nb of bit/char?
-Parity?
-Redundancy check?
-Carrier always on?
Only asked for half duplex
-Synchronous mode?
-For output line define now:
    Int level:

S2 - SLCU2S

-Addr:
-Int level
-Full duplex?
-Programmed channel?
-Datem2 status mode?
-Switched line?
-Nb of bit/char?
-Parity
-High frequency?
-Carrier always on?
Only asked if half duplex
-Preceeding line is full duplex, define now output line:
Only asked if full duplex
-Addr:
Only asked if full duplex
-Int level
Only asked if full duplex
-Inhibit character synchro?
-Automatic syn generation?
-Hardware control char detection?
-Ascii code?
-Redundancy check?
Now, for half duplex, the generation asks for the specification of the second line, starting with 'ADDR:'. For full duplex, the generation for SLCU2S is ended.

S4 - SLCY4

-Addr:
-Int level:
-Full duplex?
-Programmed channel?
-Datem2 status mode?
-Switched line?
-Nb of bit/char:
-Parity:
-Ascii code?
-High frequency?
-Carrier always on?
Only output if half duplex
-Preceeding device is full duplex, define now output line
-Addr:
Only output if full duplex
-Int level
Only output if full duplex
Now the generation process starts with asking the specifications for the other three lines. The sequence is the same as for the first line, except for the question 'FULL DUPLEX', which is not asked for the 3rd and 4th line and for the 2nd line as follows:
-For the other lines define half or full
    Full duplex?

A4 - ALCU4

The question sequence for this device is excactly the same as for the SLCU4.

A8 - AMA8

-How many lines do you want?
-Addr:
-Int level:
The interrupt level must be even. It is the input interrupt level, the output interrupt level is implicitily assigned one level higher.
-Same definition for all lines?
-Switched line?
-Full duplex?
-Carrier always on?
Only asked if half duplex
The AMA8 controller is always connected to programmed channel.
-Nb of bit/char:
-Parity:
-Datem2 status mode?
Now, when for all lines the same definition was wanted, the AMA8 generation is ended. If not, the generation process asks for the next line, starting with 'SWITCHED LINE'. The specifications are asked for each line, as specified in the answer on the 'HOW MANY LINES DO YOU WANT' question.

H1 - HDLC

-Addr:
-Int level:
-Full duplex?
-Programmed channel?
-Switched line?
-Carrier always on?
Only output if half duplex
-Define now output line
-Addr:
-Int level:
-Do you want HDLC procedure?
This question is only output for full duplex mode. if the reply is YES, it is followed by the HDLC procedure dialogue as described above. If the reply is NO,the HDLC generation is ended.

L6 - LSM16

-How many lines do you want?
-Addr:
-Int level:
-Double frequency?
-Same definition for all lines?
-Nb of bit/char:
-Nb of stop bit:
The last two questions are repeated for each line if not all lines have the same definition.
Then the generation for the LSM16 is ended.

A2 - ALCU2

The question sequence for the ALCU2 is the same as for the SLCU4. Only the 'REDUNDANCY' question is asked additionally.

HV - HLVCU

The question sequence for the HLVCU is the same as for the HDLC.

End of Datacom generation

After the specification of all Datacom controllers, the system generation process outputs some general questions about it:

MAX NUMBER OF TIMER
Reply a value in the range 1-200, which specifies the highest number of pendingtime out requests in the system (only asked if DATEM2 is generated).

Error message
MAX NUMBER OF TIMER OUT OF RANGE

MAX VALUE OF TIME OUT DELAY (*0.1S)
Reply a hexadecimal number in the range /0A - /7FFF specifying the highest time out value in the system in tenths of a second (only asked if DATEM2 is generated).

Error message
DELAY OUT OF RANGE

TABLE NAME (2 CHAR)
Reply a two character name for a special character table. If all special character tables have been defined, reply END. Note that EN only is seen as a special character table name that does not end the generation.

Error message
TABLE ALREADY DEFINED

If a table name was entered, the system will require further information about the table contents and asks:

NB OF EDITION CHAR
Reply an integer specifying the number of edition characters that will be entered.

EDIT:
Reply p,c where:
p is a process value, one of the following values:
0 Delete the previous character and the special character
2 Delete all previous characters and the special character
4 Ignore the special character if it is a leading character
6 Ignore the special character in all cases.
c is the special character

Error messages
ILLEGAL PROCESS NUMBER
ILLEGAL CHARACTER VALUE

When all special characters have been input, the system will ask:

NB OF TERMINATION CHAR:
Reply aan integer, specifyng the number of termination characters to be entered in the table. For each character the system will prompt with:

TERM:
Reply the termination character.

The system will repeat the prompt until the specified number of termination characters have been entered and after that the system asks for the next special character table name. When all special character tables have been entered, the system outputs:

END OF SYSTEM GENERATION
it then creates the configuration file and returns to the BCP. Example

Listing of the Dialogue on the Typewriter

```
***MASR EXTENDED MODE VERSION 01 ***
   DATE:83,8,30
   TIME:17,00
FCL:%%B2LP DSK2=/C2
FCL:BYE BATCH
BCP::JOB USID=MASGEN
BCP:%%CONGEN
SPECIFY CPU TYPE:
P859
          SPECIFICATION OF DEVICES

DEV: TY10,6
```

```
NUMBER OF LINES/PAGE: 0
DEV: LP07,23
PROGRAMMED CHANNEL? N
NUMBER OF LINES/PAGE? 40
DEV: CD02,17
DISK TYPE (1215,1216): 1215
DEVICE 22 ON THIS CONTROLLER ? Y
DEVICE 12 ON THIS CONTROLLER ? N
DEV: CC16,16
DISK TYPE (40,80,150,300): 40
DEVICE 36 ON THIS CONTROLLER? N
DEV: MT04,19
DEVICE 14 ON THIS CONTROLLER? N
DEV: CR06,21
DEV: DM18,8
NUMBER OF LINES: 1
SAME DEFINITION FOR ALL LINES? Y
LINE NUMBER: 0
DEVICE TYPE (DY, TY, LP, TK): DY
NB OF CHAR PER LINE: 80
NB OF LINES PER PAGE: 24
ECHO MODE? Y
RESTART TIME FOR PAGE HANDLING(0 TO 255 MN): 2
TIME OUT(0 TO 255 MN): 7
PARITY(N=NOPAR, O=ODD, E=EVEN): E
DEV: MF03,18
DEVICE 13 ON THIS CONTROLLER? N
DEV: EN
USER INT: EN
                    DEFINITION OF SYSTEM MACHINE

STACK SIZE (/200 (NORMAL) TO /800): /200
RTC FREQUENCY (50,60): 50
NB OF CHAR FOR SYSTEM DYN AREA : 0
NB OF SOFTWARE LEVELS (MULT OF 15) : 120
FLOATING POINT PROCESSOR ? N
TDFM ? N
MAX NB OF BUFFERS FOR TDFM : 12
NETWORK ACCESS? N
CORE LKM OPTION? Y
LKM 17 CORE RESIDENT? Y
LKM 52 CORE RESIDENT? N
LKM 55 CORE RESIDENT? Y
LKM 56 CORE RESIDENT? N
LKM 57 CORE RESIDENT? N

                    DEFINE SYSTEM FILE CODES

FCD: 1,TY10
FCD: 2,LP07
SPOOLED DEVICE ? Y
FCD: 3,CR06
SPOOLED DEVICE ? Y
FCD: /E0,TY10
FCD: /EF,TY10
NUMBER OF STOP BIT (NORMAL VALUE 2): 2
FCD: /C0,CD22
FCD: /C1,CC16
FCD: /C2,CD02
FCD: /C3,MF03
```

```
FCD: EN
NAME OF SUPERVISOR DAD : SUPERV
FILE CODE OF DISK : /CO
SWAPPABLE PROGRAMS? Y
FOR SWAP DAD, GIVE NOW
FILE CODE OF DISC: /CO

       DO YOU WANT DATACOM (Y/N)? N

               END OF SYSTEM DEFINITION

BCP::EOB
END OF BATCH
```

Listing of the Generated File

```
      IDENT CONFIG
MACRO DEVICE
$MACRO DEVICE;
$DEV  !DNAM=TY,ADDR=10,INT=06,PAGE=00;
$DEV  !DNAM=LP,ADDR=07,INT=23,PAGE=40;
$DEV  !DNAM=CD,ADDR=02,INT=17,TYPE=20;
$DEV  !DNAM=CD,ADDR=22,INT=17,TYPE=21;
$DEV  !DNAM=CC,ADDR=16,INT=16,TYPE=22;
$DEV  !DNAM=MT,ADDR=04,INT=19;
$DEV  !DNAM=CR,ADDR=06,INT=21;
$DEV  !DNAM=DM,ADDR=18,INT=08,PAGE=024,ECHO=ECHO..,PAR=EVEN.,        *
   RSTIME= 2,TMEOUT= 7,CHPL=80,DTYP=DY,LNNB=0,MXLB=1,DADR=18;
$DEV  !DNAM=MF,ADDR=03,INT=18;
$MEND;
EOS
MACRO SYSMAC
$MACRO SYSMAC;
$HARD !CPU=P859;
$SELECT  !MONITR=MAS8,STACK=0200;
$RTC  !FREQ=50;
$DYNAREA !SIZE=0000;
$MAXPRO !NBPROG=120;
$DUMPSA;
$PMDUMP;
$EDFM   !MAXBUF=12;
$CORLKM !L17=Y,L52=N,L55=Y,L56=N,L57=N;
$SYSFC  !FCOD=01,DNAM=TY,ADDR=10,LNB=0;
$SYSFC  !FCOD=02,DNAM=LP,ADDR=07,SPOOLD=F3;
$SYSFC  !FCOD=03,DNAM=CR,ADDR=06,SPOOLD=F2;
$SYSFC  !FCOD=E0,DNAM=TY,ADDR=10,LNB=0;
$SYSFC  !FCOD=EF,DNAM=TY,ADDR=10,NSBIT=2,LNB=0;
$SYSFC  !FCOD=C0,DNAM=CD,ADDR=22,LNB=0;
$SYSFC  !FCOD=C1,DNAM=CC,ADDR=16,LNB=0;
$SYSFC  !FCOD=C2,DNAM=CD,ADDR=02,LNB=0;
$SYSFC  !FCOD=C3,DNAM=MF,ADDR=03,LNB=0;
$SUPERV !DADNAM=SUPERV,FCOD=C0,DNAM=CD,ADDR=22;
$SWAP !FCOD=C0,DNAM=CD,ADDR=22;
$MEND;
EOS
MACRO    DEVBIS
$MACRO DEBIS;
$MEND;
EOS
MACRO    DTC
$MACRO DTC;
$MEND;
EOS
:EOS
```

Step 9 - Generating the MAS System Modules

The BCL catalogued procedure %%GENMAS is invoked for this phase. The command format is:

```
    %%GENMAS [CONF=filename][,SYS=mac]
```

filename	is the name of the configuration file. The default value is CONFIG.
mac	the macro file to be used for generation. The default is TABLC1, which is used for TMS and for a system without data communication. For DATEM2, the value must be DTCTB1.

Remarks The generated modules are kept on the MASGEN Userid.

Step 10 - Assembling and Cataloguing the MAS Tables

The BCL catalogued procedure %%ASMAS is used for this phase. The invocation command is:

```
    %%ASMAS [LIST=b][,OUSI=u]
```

b is YES or NO, default is NO Assembly listing required.
u is the userid of the starter pack; the default value is MASGEN

Remarks
Before assembling the generated MAS modules, a possibility exists to alter some system default values, which cannot be specified during CONGEN. These values are contained in the CVT (communication vector table) in the generated module T:CVT. The values that can be altered are:

Max. number of Foreground segments	- CVT displacement: /34 (default: 9)
Max. number of scheduled labels	- CVT displacement: /32 (default: 7)
Max. number of filecodes	- CVT displacement: /36 (default: 50)
Max. number of blocking buffers	- CVT displacement: /38 (default: 10)
Swap minimum resident time (sec)	- CVT displacement: /4A (default: 3)
Number of linecodes (datacom)	- CVT displacement: /76 (default: 20)

For a non-MAS 8 system (MAS 6, MAS 7) the object modules produced will be catalogued in one of two object libraries:
MASOB - in userid MASGEN
LASTOB- in the system userid of the starter disc pack.
For MAS 8 systems (MAS 8.0, MAS 8.50) all produced modules are catalogued inthe library MASTAB in Userid MASGEN.

Step 11 - Link Editing the System

The BCL catalogued procedure %%OLEMU is used for this purpose. The format of the invocation command is:

```
    %%OLEMU [USID=u][,DAD=d]
```

u is the userid of the generated system. Default is MASUP.
d is the DAD filecode of the generated system. Default is /F2.

Remarks A copy of the generated system is kept in the MASGEN library under the name SUP, type LM if a non-extended mode and MASR, type LM if anextended mode system has been generated.
The file SYSMAP, type UF, is catalogued on the system disc and also listed. It contains the Link Edit map of the generated supervisor.
The specification of the parameter DAD = /F0 is allowed for users having only one disc.

The OLEMU procedure links a supervisor. For non-extended mode systems all modules to be linked are contained in two or three libraries:

MASOB	- MAS object library
MASTAB	- Generated MAS modules (MAS 8.50)
LASTOB	- Modules that has to be linked at the end of the monitor.

For extended mode systems, the modules to be linked are contained in a lot of libraries:

MASTAB	- Generated modules
MASLIB	- Monitor subroutines
EXECOB	- Hardware interrupt routines, LKM handlers
IOCOOB	- General I/O modules
IOCSOB	- Divers
TMSOB	- TMS modules
DT2OB	- DATEM2 modules
HCPROB	- HDLC procedure modules
SUPVOB	- System programs
FMSOB	- Disc I/O modules
TDFMSG	- TDFM segment
TDFMOB	- TDFM core resident part
TDFROT	- TDFM root
SVCOB	- Core resident LKM modules (17, 52, 55, 56, 57)
INIOB	- Monitor initialization modules
DIVROB	- Dump and Spool modules
ACROB	- Network access modules (only for AMS systems).

The following syntax notations are used in this manual in order to make MAS as simple as possible to understand.

EXAMPLES

Several examples appear in this manual. Where an item is described as a part of a larger piece of coding it may be shown thus:

```
    .
    .
    .
    .
    %%PROC A,B,C
    .
    .
    .
```

This indicates that a PROC statement would be found amongst other statements or text.

FORMATS

Brackets

Two types of brackets are used to describe formats:

```
    []  brackets enclose optional items.
    {}  brackets enclose items in two ways:
             {A,B,C}
        Any number, but at least one of the items in the list must be coded
        (separated by commas.)
```

A vertical line is also used to indicate alternatives, thus:

    {A | B}

means either A or B. (The spaces around the bar are not significant.)

The following are examples of the use of brackets:

    {W,X,Y,Z}

Any number, but at least one of the items must be coded; each item separated by commas.

    [{W,X,Y,Z}]

If desired any number of items may be coded, each item separated by commas.

    {ABC | XYZ | {1,2,4,8}}

One of the three items must be coded. The item 1,2,4,8 is itself a list. If this item is to be coded any number, but at least one of the items in the list must be coded. Items are separated by commas.

Full stops (...) appear within formats to indicate the further occurrence of an item which has already been described.

Spaces

MAS is largely free form. However, spaces are significant in certain instances:

- Within formats spaces are shown as < > or [ ]. The use of < > indicates that at least one space must be coded. The use of [ ] indicates that if desired at least one space may be coded.
- Spaces and commas are not interchangeable. Where commas are shown as applicable they must be coded.

Commands

The following rules apply to the construction of SCL/FCL commands:

- Each command must be contained in one record, that is, one line on the console, one punched card, one paper tape record, one sequential disc record, etc.
- Each record must contain only one command.

Each command has the same basic format:

- A three character ASCII mnemonic code identifying the command type.
- One or more blanks.
- Parameters.

Parameters:

- May be optional.
- Are positional; position is implied by coding commas as separation characters. (Trailing commas are not necessary.)
- Are always positive integer decimal or hexadecimal values.
- Must not be separated by or contain blanks.

Procedure parameters:The maximum line lengths in procedures are for S:PROC/F:PROC 72 characters and for B:PROC 80 characters.
Key-word parameters may have a maximum length of 4 chaarcters, both in B:PROC and S:PROC/F:PROC.
Parameter values may be up to 8 characters in S:PROC/F:PROC and up to 40 characters in B:PROC. The maximum length of a procedure name is always 6 characters.

Each record must be terminated by the standard end-of-input record identifier:
<u>CR</u> for the console.
<u>Blanks</u> up to column 80 for the card reader.
<u>Sector</u> and <u>displacement words</u> (written by the sequential access method) for a disc file.

GENERAL

A program in any machine, be it system, foreground or background, may issue a request for monitor services by means of the LKM instruction, but the validity of these requests and their subsequent execution vary according to the type of machine issuing the request. In addition, within any one machine where a particular LKM is permitted, there sometimes exist variants of these LKM instructions which modify their execution.

The LKM requests are described in the order of the DATA n which follows the LKM instruction. If there is no indication of the machine to which the LKM request applies, it applies to both foreground and background machines.

LKM 1 - I/O Requests

Purpose
To initiate an action on, or retrieve information about, a device or file.

Applies to: Foreground, Background.

Calling Sequence

```
    LDK   A7,L      (or LDKL if bits in 1st character are set)
    LDKL  A8,M
    LKM
    DATA  [-]1
    [DATA  N]
```

Where L = Request Order Code (see below)
M = Event Control Block (ECB) Address
N = Scheduled Label Address.

Request Order Codes:

bit 0 = 1	Non capital letters (a-z) converted to capital ones (AMA8 and ASCU4Z).
bit 1 = 1	An address of an SCT (special character table) is recorded in word 5 (ECB+10) of the ECB (AMA8 and ASCU4Z).
bit 6 = 1	Timeout period is specified in location ECBHD (word 6) of the ECB (for teletypes etc.)
bit 6 = 0	Default value of timeout defined at Sysgen is to be used.
bit 8 = 1	Implicit Wait: The requesting program will be put into a wait state until the operation is terminated.
bit 8 = 0	No Implicit Wait: Control is returned to the calling program as soon as the request is recorded.
bit 9 = 1	User Error Action: The requesting program will process all abnormal or error conditions. The hardware status is returned in this case.
bit 9= 0	System Error Action: The system performs the standard error action and returns an error status to the calling program.

Bits 10-15 of A7 define the function required:

/00	Get device/file description
/01	Basic Read
/02	Standard Read
/05	Basic Write
/06	Standard Write
/0A	Direct Read (Disc File)
/0B	Direct Write (Disc File)
/11	Direct Read (DAD or Disc Unit)
/15	Direct Write (DAD or Disc Unit)
/10	Replace a bad track
/12	Replace a bad track (CDC disc)
/13	Seek to track zero (CDC disc)
/18	Write home address and premark the track (CDC disc)
/30	Get Information about a Filecode
/22	Write EOF Mark
/26	Write EOS Mark

For Magnetic Tape and Cassette the following extra codes are available:

/16	Skip Forward to EOF Mark
/24	Write EOV
/31	Rewind (also DFM file)
/32	Fast Search forward to tape mark (cassette only)
/33	Skip 1 block backwards
/34	Skip 1 block forwards (MT only)
/35	Fast search backward to tape mark (cassette only)
/36	Skip backward to EOF mark
/37	Lock (cassette only)
/38	Unlock.

For Flexible Disc the following request order codes are available:

/21	Read VTOC
/25	Write VTOC
/2D	Door Lock
/2E	Door Unlock
/2F	Write Deleted Data Address Mark
/3A	Compound Read
/3B	Compound Write
/3C	Search Key with Mask
/3D	Write Deleted Data Address Mark and Verify
/3E	Search Key
/3F	Write Sector and Verify.

For all these functions MAS is compatible with present systems, the DFM/TDFM functions remaining unchanged. DADs can be accessed via filecodes /F0 - /FF, they can be read by any user program in the batch machine, but written only by the BCP, EDF and Librarian processors. Foreground users can access their DADs and discs directly.

The physical disc units (filecodes /C0 - /CF) are only accessible to foregroundprograms and the Librarian processor. The orders /11 and /15 are used to read and write a sector of the disc. Great care should be taken to use direct disc access.

<u>The ECB Layout</u>

An ECB is a 5, 6 or 7 word table containing required parameters, but for those functions not requiring words 6 or 7, these words may be omitted. For get information order codes (/00 and /30), the ECB is considerably longer.

bits	0	1	7	8 15
ECB 0	E	L		Filecode
ECB 2	Start Address of Record Area			
ECB 4	Requested Record Length			
ECB 6	Effective Length			
ECB 8	Returned Status			
ECB 10	See below			
ECB 12	See below			

ECB -2 may be used to contain the address of an ECB to whose event this I/O operation is linked (multiple wait; see LKM 2). On completion of this I/O operation, the event bits of all chained ECB's will be set.

For Flexible Disc the layout is as follows:

```
bits     0 1                   7 8                      15
ECB 0   |E|L|                   |        Filecode         |
ECB 2   | Buffer Address (containing delete pattern       |
        | if order /3D is used)                           |
ECB 4   | Requested Length                                |
ECB 6   | Returned Effective Length                       |
ECB 8   | Returned Status                                 |
ECB 10  | No. of first sector to be read/written          |
ECB 12  | zero                  | Timeout value           |
```

Explanation of Tables

ECBFC (byte 0): E and L (bits 0 & 1) refer to event handling (See 'Wait for Event' request, LKM 2).
(byte 1): Bits 8-15 contain the filecode.

ECBBF (bytes 2 & 3) : Start Address of the record buffer area.

Note: For flexible disc the following apply:
- For request code /3D (delete) the buffer should contain the delete pattern.
- for request codes /3C and /3E the buffer should contain the search key block.

ECBRL (bytes 4&5): The length in characters of the record to be written, or the maximum length of record to be read. If an odd number of characters is read, MAS will round up to an even number by adding 1.

ECBEL (bytes 6&7): The actual length in characters which has been transferred. This is returned by the Monitor upon completion of the operation.

ECBST (bytes 8&9): The status returned by the Monitor upon completion of the operation. These are described below under 'Returned Status'.

ECBSC (bytes 10&11):
a) For Direct Access on Disc Files (DFM) or DAD devices, it specifies the relative sector number, from the beginning of the file or device.
b) For 'Disc Physical Unit' (filecodes /C0 - /CF), it specifies the cylinder number.
c) For flexible disc, this is the absolute sector number except in the case of search orders /3C and /3E, for which the user leaves it blank and MAS returns the actual sector number found in a successful search operation.
d) For AMA8 and ASCU4Z devices, this is the address of the SCT (special character table, when bit 1 in the order is set.

ECBHD (bytes 12&13):
a) For 'Disc Physical Unit' (filecodes /C0 - /CF), it specifies the Head Number (byte 12) and Physical Sector Number (byte 13) for the transfer.
b) For such devices as displays, teletypes and floppy disc, the timeout period is specified in byte 13, byte 12 being set to zero, when bit 6 in the order is set.

Returned Status (ECBST)

a) Zero: The operation terminated satisfactorily.

b) Positive: The operation was completed, but the following conditions were encountered:

1	EOF encountered (Read).
2	EOS encountered (Read).
4	Data Error.
8	Incorrect Length.
/10	End of tape, end of media, request done.
/20	Beginning of tape.
/40	End of tape reached, but the current record has been read or written (warning signal).
/80	EOV mark detected.
/100	No data on MT or cassette.

c) Negative (bit 0 set):
Bit 1 = 0: Bits 2-15 indicate the hardware status.
Bit 1 = 1:

/C001	Filecode illegal or not assigned.
/C002	Device attached to another program.
/C008	Buffer address or requested length invalid.
/C010	Function unknown or incompatible with the device or file, or invalid sector number for Floppy Disc.
/C020	Write protection on Disc File.
/C040	End of medium: current operation aborted.
/C080	Timeout.
/C100	Disc Queue overflow.
/C200	Dynamic Buffer overflow; no disc blocking buffer free.
/C400	Blocking overflow (No free granule).

Notes: Status code /C010 is returned when trying to write to cassette with a length greater than 256, whereas status code 0 is returned when trying to write to magnetic tape with a length greater than 4096.

Order code /30

The request order /30 (Return Information About a Filecode) is processed by a disc-resident program; the user only has to specify the filecode in ECBO. The information is returned as follows:

ECBBF Device Name:

DY	display
TK	cassette
MT	magnetic tape
LP	lineprinter
CR	cardreader
DK	disc
PL	plotter
PR	papertape reader
PP	papertapepunch
FL	0.25m data floppy
DD	DAD device
DL	logical disc file
EF	extended (TDFM) file
NO	file code assigned to NO device
TY	typewriter

ECBRL Maximum Record Size (in characters).

ECBEL

a) For non-disc devices:

Bits

0-9 These may be set but are not significant to the user. A description of these (device dependent) bits may be obtained from the Trouble Shooting Guide, in the DWT chapter.

10-15 Contain the device address.

b) For Disc Files (DFM):

Bits 8-15 contain the DAD filecode (/F0 - /FF).

c) For DAD devices:

Bits 8-15 contain the disc filecode (/C0 - /CF) on which the DAD is located.

ECBST Status of the operation:

Zero = OK

/C001 = Illegal Filecode.

/C008 = Invalid Buffer Address and/or Requested Length.

ECBSC:

a) For Disc File (DFM):

Bit No.	Meaning if set
0	-
1	Write Protected
2	Source
3	Object
4	Undefined File (User File)
5	Load Module
6	Temporary Disc File (Catalogued file if not set)
7	Direct Access
8	Sequential Access
9	EOF mark written
10	-
11	-
12	Consecutive File
13	-
14	-
15	-

b) For DAD:

Bits 0-7 No. of Interlaces

Bits 8-15 No. of sectors/granule.

c) For AMA8 and ASCU4Z

If bit 0 = 1: linenumber in bits 1-15 (0-7 for AMA8, 0-3 for ASCU4Z).

For extended files and filecode assigned to NO device only the device name is filled in and the rest of the ECB is set to zero.

Order Code /00 - Get Device/File Description

This order has been introduced as an extension to order code /30, since further details concerning the characteristics of a device or file are required for some system processors.

The user enters the event bits (E & L in the table) and the filecode in ECB0 as usual before issuing the request. The required information is returned as follows:

For information common to all devices:
ECB2: bits 11-15 contain the assign type:
- 0 The filecode is assigned to a physical device
- 2 The filecode is assigned to a DFM file
- 4 The filecode is assigned to a DAD device
- 6 The filecode is assigned to a TDFM file.

ECBST: Returned Status.

The contents of the other locations depend on the device type. For non-TDFM devices or files, the ECB consists of 15 words numbered by byte (0, 2, 4, ... 28), which contain the following information:

a) For a physical device:

bits	0	1	7	8 9	10 15
ECB0	E	L			Filecode
2	Reserved			Type = 0	
4	Device Name				
6	Reserved				Device Address
8	Status				
10	Reserved				Device Type
12	Best (Max.) Length				
14					
16					
18	Linenumber +/8000 (AMA8 and ASCU4Z)				

Explanation

Device Name is 2 ASCII characters, as for order /30.

Device Address occupies the least significant 6 bits.

Device Type occupies the least significant 6 bits, thus:

/00	Typewriter
/02	Display
/04	Card Reader
/08	Paper Tape Reader
/0C	Paper Tape Punch
/10	Line Printer
/14	Plotter
/18	Magnetic Tape
/1C	Cassette Tape
/20	X1215 Removable Disc
/21	X1215 Fixed Disc
/22	CDC Disc (400 cyl., 5 heads)
/23	CDC Disc (800 cyl., 5 heads)
/24	CDC Disc (400 cyl., 19 heads)

/25	CDC Disc (800 cyl., 19 heads)
/26	X1216 Removable Disc
/27	X1216 Fixed Disc
/28	Fixed head disc
/29	CDC-CMD 16M removable disc
/2A	CDC-CMD 16M fixed disc
/2B	CDC-CMD 48M fixed disc
/2C	CDC-CMD 80M fixed disc
/2F	Floppy disc.

b) For Line Printers:

ECB14	No. of remaining lines (in binary) on the current page.
ECB16	No. of lines/page (binary).

c) For Disc Devices:

ECB14	The VTOC sector size in characters
ECB16	Absolute sector address of VTOC
ECB18	No. of tracks/cylinder
ECB20	Address of the Bad Track List sector
ECB22	No. of sectors/track
ECB24	No. of interlaces in the first DAD, containing the VTOC
ECB26	Pack Volume No. (binary)
ECB28	Type (only for CDC-CMD) lay out: Bit 0 = 0: removable part Bit 0 = 1: Fixed part Bit 10-15: device type.

d) For a DFM File:

	0	1 7	8 15
ECB0	E	L	Filecode
2	Reserved		Type = 2
4	'DL'		
6	DAD Filecode		
8	Status		
10	Flags as ECBSC in order /30		
12			
14	GRANTB addr in the DAD		
16	Relative Current Sector Number		
18	Addr of Current Sector in the DAD		
20	Rel. Highest Sector No. in the DAD		
22	Sector Size in characters		
24	No. of sectors per granule		
26			
28			

e) For a DAD device:

	0 1 7	8 11 10 15
ECB0	E | L |	Filecode
2	Reserved	Type = 4
4	Device Name = 'DD'	
6	Disc filecode (/Cx)	
8	Status	
10	No. of sectors/track	
12	Sector length in characters	
14	No. of cylinders of the DAD	
16	Addr of the 1st cylinder of the DAD	
18	No. of sectors/granule	
20	No. of tracks/cylinder	[illegible]
22	No. of interlaces	
24	)	
26	) DAD Name	
28	)	

f) For a TDFM file:

	0 1 7	8 15
ECB0	E | L |	Filecode
2	Reserved	Type = 6
4	'EF'	
6	DAD Filecode of Descriptor File	
8	Status	
10	Disc pack number (binary)	
12	)	
14	) DAD name	
16	)	
18	)	
20	(Userid	
22	(	
24	)	
26	)	
28	(Extended File name	
30	)	
32	) Maximum number of records (4 char)	
34	)	
36	) Creation date (4 char)	
38	)	
40	) Current number of records (4 char)	
42	)	
44	) Number of deleted records (4 char)	
46	)	

Order Codes /10 and /18 - Replace a Bad Track

This function is only used by the Librarian Processor when a bad track is detected during a Premark operation; its purpose is to flag the track as bad and replace it by a good one. It will modify the 'Bad Track Table' kept at the beginning of the disc and in memory, and it will premark the replacement track in the same format as that of the bad one.

Before issuing the LKM, the cylinder number and head number of the track to be replaced are entered into the ECB thus:

ECB0		Disc Filecode (/Cx)
2	unused	
4	Record Length in characters	
6	unused	
8	unused	
10	Cylinder No.	
12	Head No.	No. of ꞏecords in track

After successful completion of the operation, the system returns the cylinder and head numbers of the replacing track in the same locations, i.e. in ECB10 (cyl. No.) and ECB12 (head No. left byte).

Besides the status code in A7, the following are returned in ECB8:

Zero	Successful termination
/C001	Invalid filecode (not disc)
/C010	Invalid order (requesting program not Librarian Processor)
/C020	Overflow of Bad Track Table, but bad track is flagged
/C021	Too many bad tracks (more than spare tracks)
/C023	Track capacity error (too many records and/or recd. too long)
/C200	Dynamic area overflow - but the bad track is flagged.

Floppy Disc Order Codes

See Appendix E, Peripheral Drivers.

Write Home Address (CDC Disc)

This order uses the same ECB layout as for order /12.

Value of register A7 after LKM 1

When A8 contains a valid ECB address (within the program boundaries) the value of A7 is unchanged.
When A8 is invalid, A7 contains: -1.

LKM 2 : Wait For Event

Purpose
To synchronize tasks within a program or between one or more programs.

Calling Sequence:
```
    LDKL  A8,ECB
    LKM
    DATA  [-]2
    [DATA  Scheduled Label]
```

The Event Word pointed to by A8 is modified on termination of the awaited operation. The format is:

```
0 1                          15
_____________________________
|E|L|________________________|
```

E=0	Event has not occurred
E=1	Event has occurred. E is always set to 1 after completon of a LKM 2
L=0	Single Event
L=1	Chained event; when the event occurs, it implies that the system must set the other event, the ECB address of which is given in (A8) - 2 (See 'Set Event' request).

Effect
When a program issues this wait request and the event has already occurred, control is returned immediately to the requesting program. If the event has not occurred, the requesting program will be suspended until the event does occur. The event can either be a single one or an OR of several, allowing the requesting program to wait until any event among a list of events has occurred.

Remarks
If bit 15 of A8 is set when this command is issued by a swappable program, it will be swapped out until the awaited event occurs.

Returned Status:
When control is returned to the requesting program, the reply in A7 is as follows:

A7 = 0	The program was in a wait state.
A7 = -1	The ECB address (or one of a chain of addresses) is invalid.
A7 = 1	The event is already set.

LKM 3 - Exit

Purpose

This depends on whether the request is issued by a foreground or background program. In a foreground machine the purpose is to terminate a task, while for the background machine it can be used to request a return to MAS or to condition the execution of subsequent commands in the Job Stream when batch processing.

Foreground Programs

Calling Sequence:

```
     LKM
     DATA 3  N.B. Scheduled labels are ignored if used with LKM 3.
```

Effect:

If issued by a scheduled label routine, then the next scheduled label for this task is entered. If there are no more scheduled label routines for this task, control is returned to the main program at the instruction following the last scheduled label interrupt.

If the request was issued by a foreground program and uncompleted scheduled label routines exist, the program is suspended until all such scheduled label activity has terminated.

If no scheduled label activity remains when this request is issued by a foreground program, the activation queue for the program is examined and the oldest entry, if any, becomes an eligible task. If, however, the activation queue is empty, subsequent action depends on the manner in which the program was declared, e.g.:

for	LOD:	Nothing happens.
-	REP:	Nothing happens.
-	SWP:	The swap image on disc and the pages it occupies in memory are freed. The initial core image on disc remains.
-	RON:	The pages in memory are freed, the core image on disc remains.

Middleground Programs

When middleground programs exit, they are disconnected and their memory pages freed. The swap image on disc (if any) is deleted, even though the program may be reactivated later.

Background Programs

An LKM 3 request can be issued by a scheduled label routine on termination, or by a background program in order to return control to MAS. The background program may be the Batch Control Processor (BCP), another type of processor, or a user program.

Calling Sequence:

```
    [LDK    A7,L]
    [LDKL   A8,M]
    LKM
    DATA    3
```

L = A BCP Severity Code or Exit Code (see below).
M = The address of a Job Parameter Table (JPT) if the requesting program was the BCP.

Effect:
If the requesting program is the BCP, A7 is not examined, but the JPT pointed to by A8 is accessed. This will indicate either that the :EOB has been encountered and that batch processing is to terminate, or it will contain details of the next program to be executed (see Part 3, The Batch Machine).

If the requesting program is another Processor, or a user program, A8 is ignored and A7 is examined. The following bits have significance:

Bit 8 = 0	No post mortem dump is produced.
Bit 8 = 1	The program has aborted and a post mortem dump is produced in accordance with the value of the DUMP keyword specified on the BCL RUN or Processor call command, i.e.:

DUMP=ALL:	Both system and background machines are dumped.
DUMP=PROG:	Only the background machine is dumped.
DUMP=NO:	No dump is produced.

Bits 9-15 These contain an exit code, or severity code, which is used by the BCP to determine subsequent processing of the job stream. The value ranges from 0 to /7F according to the severity of the error leading to the exit command. The following conventions should be adopted by user programs:

Code	Meaning
0	No error; normal termination.
/01-/2F	Minor Errors (see Chapter 8).
/30-/3F	Errors which require subsequent user programs to be ignored.
/40-/4F	Errors preventing the execution of a Linkage Edit (e.g. a compilation error causing no object code to be produced).
/60-/6F	Language Processor errors.
/70-/7F	Serious errors requiring further batch processing to be terminated (e.g. disc overflow, dynamic area overflow).

If the requesting program is attached to a scheduled label routine, control returns to the interrupt point in the main task, unless there is more outstanding scheduled label activity to be completed. In this case all such activity is completed before the return is made, the program being suspended, if necessary, while awaiting completion of the service on which the scheduled label routine is dependent (e.g. an I/O request).

If the exit code is /FFFF, the execution of any part of the program is suspended. It has the same effect as LKM 46 (Abort) except that the activation queue, if any, is re-started as soon as any pending events have occurred.

On exit of any foreground or background program, the exit code, i.e. the value of A7 upon LKM 3, is printed on filecode /01.

LKM 4 - Get Dynamic Buffer

Purpose
To acquire work areas, scratchpad areas, or I/O buffers; used mainly by re-entrant programs which must safeguard data against corruption when interrupted.

Calling Sequence
A7 is loaded, either with zero or the length of buffer required (expressed in characters), and the sign bit is set as follows:

bit 0 = 1:	The program is to be suspended until buffer space becomes available.
bit 0 = 0:	Control is to be returned to the requesting program immediately if no buffer space is available.

The following statements are then issued:

```
    LKM
    DATA  [-]4
    [DATA M]       where M is the address of a scheduled label routine.
```

Effect
1) A7 originally non-zero

If the request is successful, A7 is zero on return to the calling program while A14 contains the address of word zero of the acquired buffer, whose layout is:

Byte	Contents	
-6	Chain Word	
-4	The original contents of A14	
-2	The length of the Buffer (L)	
0	First word of the buffer	<- (A14)
2	.	
4	.	
.	.	
L-2	Last word of the buffer.	

N.B. The contents of A14 are restored when the buffer is released.

If the request is unsuccessful and the sign bit of A7 was originally zero, then control is returned to the calling program with A7 = 1.

If the request is unsuccessful and the sign bit of A7 was originally 1, the program is suspended until buffer area becomes available. In the case background programs, if this period exceeds the TIME parameter specified in the program's Job Parameter Table, then the program will be aborted.

2) A7 originally zero

In this case the effect is different for background and foreground machines:

a) Foreground Machines: No dynamic buffer is obtained, but:
   - for memory-resident programs A7 returns with the end address of the segment containing the program.
   - for disc-resident programs A7 returns containing the address of the last word in the last page allocated to the program.

Remarks
Dynamic areas are allocated in Segment 0 of the foreground machine. Programs in Segment 0 must not address the dynamic area using virtual addresses.

b) Background Machines: Again, no dynamic buffer is acquired, but A7 returns containing the highest address in the background machine.

Remarks

1) A background program which issues an LKM 4 or assumes that it can use virtual addresses to access memory beyond the load module must be activated by a BCL RUN or non-standard processor call command in which the SIZE parameter specifies enough pages additional to the load module to accomodate the additional areas required.

2) For memory-resident background programs, dynamic buffer areas are allocated from the pages exclusively reserved by the SCL DCB command, which are not being used by the currently active background program.

3) For disc-resident programs, dynamic buffers are allocated from the common dynamic loading area, shared by all machines, to a maximum of 16 pages.

4) If the value in A7 is set to zero, the system returns the highest address in the segment in A7

On return, the following error status may be returned in A7:
A7 = -2: chain link in dynamic area destroyed.

LKM 5 - Free Dynamic Buffer

Purpose   To release the buffer areas acquired by the LKM 4 request.

Calling Sequence

A14 should be pointing to word zero of a buffer obtained as a result of a previous LKM 4 request. If it is not, it must be set up to do so.

```
LKM
DATA   [-]5
[DATA L]    where L is the address of a Scheduled Label Routine.
```

Effect

If the buffer is successfully freed, the contents of A14 are restored from location -4 of the buffer (where they were saved by the LKM 4 request), and A7 returns with a status code of zero.

If A7 =-1 when control is returned, then A14 was found to contain an invalid buffer address when the request was issued.

If A7=-2, the chain in the dynamic area is destroyed.

Remarks

If the dynamic area was in an 'overflow state', i.e. there is at least one program suspended waiting for buffer space, then this program will restart when the LKM 5 request is actioned.

LKM 6 - Pause

Purpose
To output a message on the operator`s console and to suspend the calling program pending an operator`s restart (RS) command.

Calling Sequence

```
    Load A7 with the message length in characters.
    Load A8 with the address of the message buffer.
    LKM
    DATA [-]6
    [DATA    L]   where L is a Scheduled Label Routine address.
```

Effect
The program is suspended and the message that appears on the operator`s console (System Machine Filecode /EF) has the following format:

```
    `MACHINE m PRO: p PSE'
    `console message'
```

where `m' is the machine name
`p' is the program name
and `console message' is a string of up to 72 ASCII characters, the first two of which should be the control characters /0D0A or spaces (since MAS will substitute /0D0A for the 2 leading spaces). These control characters can also be embedded in the message to introduce newlines, since they have the effect of positioning all the characters following them in the message text at the beginning of the next line.

A status code is returned in A7 as follows:

A7 = 0:	Request was successful.
A7 = -1:	The address in A8 was invalid.
A7 = -5:	Dynamic Area overflow; the message could not be sent and the program has not been suspended.

Note: Swappable programs will be swapped out.

LKM 7 Keep Control on Abort

Purpose
To retain control in the event of MAS deciding to abort a program or after an LKM 3 (Exit) has been issued. Instead, control is returned to an address specified by the program issuing this request.

Calling Sequence
```
    Load A7 with a 'User Abort Label'
    Load A8 with the 'Abort Control Block Address'
    LKM
    DATA   [-]7
    [DATA  L]
```

Where: L is a Scheduled Label Routine Address.

- 'User Abort Label' is the address to which control is returned in the event of an abort condition arising.
- 'Abort Control Block' is a four word area set up by MAS when an abort condition occurs subsequent to the issuing of this request. The format of this control block is:

	0 – 7	8 – 15
0	Zero	Abort Code
2	Program Status Word (PSW)	
4	Instruction Counter	
6	(A8) when the program was aborted	

Effect
When this request is received MAS alters the program's control table (PCT) so that whenever an abort condition arises subsequently, control will be returned to the address specified in A7, the User Abort Label.

When an abort or exit condition arises and control is transferred to the Abort Label address, MAS enters the Abort Control Block address in A8 (all other registers are saved) and the instruction counter is loaded with the User Abort Label. These values are also entered in the Abort Control Block.

The abort codes in the Abort Control Block have the following meanings:

/01	Too many Scheduled Labels
/02	Invalid Instruction.
/03	Memory Protect Violation.
/04	Dynamic Area Corrupt.
/05	Not Used.
/06	Operator Abort.
/07	Too Many Blocking Buffers Requested.
/08	Disc Overflow.
/09	Disc Queue Overflow.
/0A	Memory Overflow During Program Load.
/0B	BCL RUN: TIME Exceeded.
/0C	BCL RUN: PRINT Exceeded.
/0D	BCL RUN: PNCH Exceeded.
/0E	Floating Point Error
/0F	LKM 46 Issued (Abort)
/10	:JOB/:EOJ Read by Batch program.
/11	I/O Error in loading the program root.
/12	I/O Error in loading an overlay segment of this program.
/13	Fatal error in spooling the I/O request.
/14	Debug error.
/15	Debug fatal error exit.
/16	Program aborted by another program in the same machine via LKM 76.

Remarks

1) A re-entrant program has one Abort Label for each task, i.e. for each PCT. The LKM 7 request only affects the Abort Label of the issuing task.

2) If a program issues several LKM 7 requests, only the last one is effective.

3) A program does not have to re-issue an LKM 7 after an abort, unless it is desired to alter the effect; multiple aborts will be handled by the most recently specified User Abort Label.

4) If the Abort Code is /06 (Operator Abort), the program must ensure that it is aborted by issuing an LKM 3 with an appropriate severity code in A7.

5) Immediately on entering the Abort Label routine, an LKM 31 (Cancel Retain Control on Abort) request should be issued, as a loop would result if an abort condition occurred within the routine itself.

6) Abort codes /0B, /0C, /0D and /10 apply only to the background machine.

7) The implementation of this request under MAS is slightly different from that under other systems, as the contents of A7 and A8 have had to be interchanged. Scheduled Label processing requires that A8 points to the Abort Control Block instead of to the Label, as with the DOS/BOS systems.

Returned Status

A7 = 0	Request Recorded.
A7 = 2	System Machine Dynamic Area overflow; the request cannot be recorded.
A7 = -1	Invalid address(es) in A7 and/or A8.

LKM 8 - Datem II or TMS Request

Purpose

To request Datacommunications I/O operations.

Calling Sequence

```
    LDK       A7,R
    LDKL      A8,A
    LKM
    DATA      [-]8
    [DATA     L]
```

where: R is a request order
A is the address of a Line Control Block (LCB)
and L is a scheduled label address.

Request orders

(A7) =	1	(A)	+	Read with Echo
	2			Read without Echo
	3	(A)	+	Read with Echo - with Timeout
	4			Read Without Echo - with Timeout
	5			-
	6			Write
	7			Write with Timeout
	C			-
	D			Change line definition
	E			Get line definition
	10			Stop the exchange
	11		*	Disconnect the line
	12	(S)		Search pattern
	13		*	Wait for a call
	14	(A)	-	Accept data
	15			Set timeout.

(S)	=	Synchronous Lines only
(A)	=	Asynchronous Lines only
*	=	Switched lines only
+	=	Full duplex lines only
-	=	Not possible with AMA8 Control Unit under multiplexed input.

Note : All these requests are serviced without wait for completion. For asynchronous lines, if register A7 contains a negative number it means that the user needs extra characters following the termination (automatic request with synchronous lines).

LCB Structure

	0	7	8	15
0	E		Line Code	
2	Buffer Address			
4	Requested Length			
6	Effective Length			
8	Status Code			
10	Edit/Convert Table Name			
12			Timeout Value	

Table Explanation

Word 1 Buffer Address = Address of the memory area where the transmission block is to be found.

Word 2 Buffer Length = Length of the buffer. This is a number of characters 1 < n < 32767 (decimal).

Word 3 Transmitted Length. This information is returned by the Monitor on service completion and provides the user with the effective length of the transmitted block.

Word 4 Service Status. This information is returned by the Monitor. It will be meaningful only when the event bit is set (it must not be modified until the event bit is set).

No bit set	no error detected.
Bit 0 set	logical line busy
1	Disconnected line (switched lines)
2	Illegal filecode (no correspondence between provided filecode and system physical lines).
3	Illegal request: . Incompatible order . Negative block length
4	Character(s) lost
5	End of carrier detection
6	Timeout request may not be served: . no room in table . timeout value out of range (SYSGEN option)
7	Buffer overflow
8	Transmission stopped
9	Power failure (option)
10	Time over
11	Break detection (Asynchronous lines only)
12	Command refused
13	Parity error (hardware detected)
14	Throughput error
15	Modem not operable.

Word 5 (U) Terminator/Edit Table Address. This word is to be filled with the Terminator/Edit Table address if a check is to be done, or with zero. Its contents must not be modified before service completion.

Word 6 (U) Timeout Value. This word provides the interval of time after which the service is posted complete. The contents of this word will be meaningful only with order numbers 3 and 4 (Read with Timeout), 15 (Set Timeout Control), and 7 (Write withTimeout). In the special case of orders 4 and 7, with Synchronous Transmissions, the time control will be inhibitedwhen the first character is input. Unit of timeout control is the tenth of a second. On option the contents of this word
may be checked against a max. Time Control value introduced at SYSGEN time.

Remarks

If editing or terminating characters have been defined at SYSGEN, the SCT name must be entered in word 5 of the LCB.

Programs using LKM 8 must be memory-resident.

For more details see P800 Datacommunications User Manuals.

LKM 10 - Connect A Program to a Timer or to the Clock

Applies to: Foreground Programs only.

Purpose
To cause a foreground program to be activated after a given interval and/or at regular intervals afterwards, or to activate a program at a specific time of day and optionally at regular intervals afterwards.

Calling Sequence

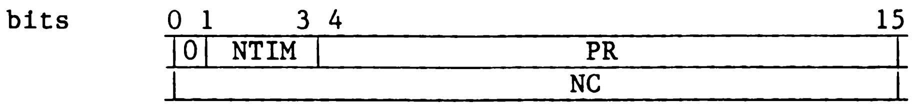
```
    A7 is loaded with the address of a Program Name Block.
    A8 is loaded with the address of a Parameter Block.
    LKM
    DATA      [-]10
    [DATA     L]
```

Where: L is the address of a Scheduled Label routine.

: The Program Name Block is a three-word area containing the Program Name in up to 6 ASCII characters, left justified and space-filled to the right.

: The Parameter Block has a format which varies according to the desired effect of this request:

a) Format 1 - Connect to a Timer



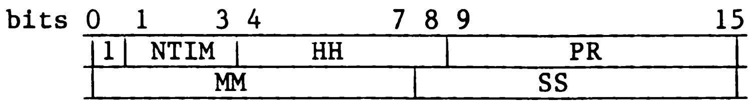
Where NTIM is the Timer Number:

0 - Hour
1 - Minute
2 - Second
3 - 100 MS
4 - 20 MS
5 - Non-Standard clock

NC is the number of cycles before the first activation (Range: 0 - 32767).

PR is the pulse rate or reactivation rate, from 0 - 2047. When PR is 0, only one activation will occur and then automatic disconnection occurs.

b) Format 2 - Connect to a Clock



Where NTIM is the Timer Number, as for Format 1.

HH/MM/SS is the absolute time in hours, minutes and seconds at which the first activation will occur.

PR is the reactivation rate, ranging from 0 - 127. If PR is zero, only one activation occurs (as for Format 1).

Effect:
LKM 10 may be used instead of an FCL CNT command to connect, to a timer or clock, a program in the same foreground machine as that in which the issuing program is running.

If Format 1 is used, the program specified in the Program Name Block will be activated after NC periods of length NTIM have occurred, and will be re-activated every PR periods of length NTIM.

If Format 2 is used, the specified program will be activated when the system clock time reaches the specified hour, minute and second, and will be re-activated every PR periods of length NTIM.

Returned Status
One of the following status codes is returned in A7 following execution of the request:

0	Connection has been performed.
/D	Invalid address in A7 and/or A8.
2	Dynamic area overflow.
-2	Timer unknown (NTIM was not in the range 1-5), or, if 5, the bare machine does not have the non-standard clock hardware option.
-3	Program unknown, i.e. not connected to a level or not defined for this machine.
-4	The program to be connected was a middleground program.

Remarks
1) If a scheduled label is used with LKM 10, it is executed immediately MAS has created the timer activation table entry, and then the next sequential instruction in the issuing program is executed. Thus, the execution of the scheduled label routine is quite independent of the activation or execution of the program connected to the timer and does not imply that the connection has been made.

2) The program to be connected must be known to the system and reside in the same (foreground) machine as the connecting program.

3) When the program is activated by the clock or timer by means of this LKM, A3 will have been set to zero. This provides a means of distinguishing activation by an LKM 10 from activation by FCL RUN or FCL ACT. In addition, the value in A4 can be used to accumulate statistics during the development of such a program:
 A3 = 0, A4 = -1 : Normal activation
 A3 = 0, A4 = -3 : Queued activation.

The contents of A4 can be used to check whether the execution time of the program is compatible with the timer interval, for if A4 is -3 when the attempt is made to activate it, it means:

a) Execution time is longer than the timer interval.

b) The program has a too low priority.

c) The program is also being activated by LKM 12, FCL RUN or FCL ACT commands and these activations are disrupting the activation of the program by timer or clock.

d) If the program requires dynamic buffers, it is having to wait for dynamic area to be released.

e) The program issues LKM`s which require the System Machine dynamic area and this is in heavy demand, causing the program to wait.

f) Another program is competing for devices required by this program.

g) The program is waiting unnecessarily for events; the use of event chains, multiple user record areas for LKM 1`s without implicit waits and scheduled label routines would reduce execution time.

LKM 11 - Disconnect a Program from a Timer or Clock (Foreground Only)

Purpose
To disconnect the program from one or more timers.

Calling Sequence

```
    A7 is loaded with the address of a Program Name Block.
    A8 is loaded with the Timer Number.
    LKM
    DATA [-]11
    [DATA    L]
```

Where: L is the address of a Scheduled Label routine.

'Program Name Block' is a 3-word area in which the user enters the name of the program to be disconnected (left justified and with trailing spaces as necessary).

'Timer Number' is as specified for LKM 10, except that if A8 contains /FFFF the specified program will be disconnected from all the timers to which it is currently connected.

Effect
The current activation of the disconnected program and all activations in its activation queue, together with all future activations by FCL RUN, FCL ACT, LKM 12 etc., will still be processed. Only new activations from a timer or clock are stopped by LKM 11. The program remains connected to a level.

Returned Status
The contents of A7 when control is returned to the calling program are:

0	Disconnection performed.
-1	Invalid address in A7.
-2	Timer unknown, or the program is not connected to a timer or clock.
-3	Program unknown.

Remarks
1) LKM 11 may be used instead of an FCL DST command to disconnect from a timer or clock a program running in the same foreground machine as the issuing program.

2) If a Scheduled Label routine is specified with LKM 11, it will be performed immediately MAS has updated its timer control tables. A return is then made to the issuing program with A7 set as above.

LKM 12 - Activate a Program (Foreground Only)

Purpose
LKM 12 can be used instead of an FCL ACT command to cause a new activation queue entry to be created for a program in the same foreground machine as that in which the issuing program is running.

Calling Sequence

```
    A7 is loaded with the address of the Program Name Block.
    A8 is loaded with the address of a two-word Activation Block.
    LKM
    DATA [-]12
    [DATA    L]
```

Where: L is the address of a Scheduled Label routine.

`Program Name Block' is a three-word area in which the name of the program to be activated is entered (left justified and space filled to the right).

`Activation Block' is a two-word area, the first word of which is not used by the system, but can be used as an event word for task synchronisation. The second word of this block is a parameter word, the contents of which are transferred to A4 of the activated program.

Effect
The following information is transferred to registers of the program to be activated:

To A3, the contents of A3 of the calling program.
To A8, the contents of A8 of the calling program.
To A4, the contents of word 1 of the Activation Block.
To A14, zero.

Remarks
1) The program to be activated must have been defined by an FCL RON, SWP, REP or LOD command for this foreground machine and have been connected to a level by an FCL CNL command or by an LKM 20 given from within this foreground machine. Alternatively, it must be a middleground program, in which case, after creating the activation queue entry, MAS will search the directories of the first Userid in all the DAD`s defined for this foreground machine in the order /F0 to /FF. The program will then be loaded into memory as a swappable program and connected to the lowest unconnected software level.

If the activated program is re-entrant a new entry is created in the Monitor Table and the program can be started immediately. It is automatically connected to the next free level below that specified when the re-entrant program was loaded and connected to a level.

3) Task synchronisation can be specified by using a scheduled label with the LKM 12. The scheduled label routine will be given control when the activated task has exited. Note that if the activated task has been aborted, bit 15 of A8 will be 1 when the scheduled label routine is entered. Meanwhile, the next instruction in the activating task will be executed. When the activating task reaches a point where it cannot proceed until the activated task has completed, it must wait. This is done using the Event Word pointed at by A8 when the LKM 12 was given, as follows:

- Program 1 issues an LKM 12 and the Event Word pointed at by A8 has zero in bit 0. Bit 1 is set to 1 if this event is chained to other events, otherwise it is zero.
- When Program 1 reaches a point where it cannot continue until Program 2 has finished, it issues an LKM 2 with A8 pointing to the event word used by the LKM 12.
- When Program 2 starts, it receives:
    - A3 and A8 of Program 1 at the time it issued the LKM 12.
    - A14 set to zero.
    - in A4, the contents of the second word pointed at by A8 in Program 1 when it issued the LKM 12.
- Just before Program 2 Exits, it sets bit 0 of the Event Word pointed to by A8 to 1 by issuing an LKM 18 (Set Event).
- The scheduled label routine (if any) specified by Program 1 when it issued the LKM 12 will then be performed. When that exits, the scheduled label routine (if any), specified by Program 1 when it issued the LKM 2, will be performed. When that exits the next sequential instruction after the LKM 2 in Program 1 will be executed.

4) If the activated program is in another segment, the parameters and parameter block must be stored in the dynamic area of Segment 0, otherwise memory-protect violation can occur.

Returned Status

One of the following codes will be returned in A7 when execution of the requesting program is resumed:

0	Program activated or an activation queue entry has been created.
-1	Invalid address in A7 and/or A8.
-3	The program has not been defined for this machine or (in the case of middleground programs) cannot be located in any of the First User DAD Directories.
-4	System Dynamic Area overflow; a new activation queue entry cannot be created or no space to create a PCT for a middleground program.
-5	The program has not been connected to a level, or there are not enough free pages to load a middleground program.
-6	The program has previously been aborted and can only be activated by an FCL ACT or RUN command.
-7	Too many reactivations for a re-entrant program, or there are no more levels available for a re-entrant program.
-8	No software level available (middleground programs).
-9	I/O error during activation of a middleground program.
-10	D:CI (swap DAD) overflow (middleground programs).

LKM 13 - Switch Program Levels (Foreground Only)

Purpose
To modify the relative priority of up to 30 programs.

Calling Sequence

```
    LDK      A7,0
    LDKL     A8,P
    LKM
    DATA     [-]13
    [DATA    L]
```

where:	L	is the address of a scheduled label routine.
	P	is the address of a three word parameter block containing the program name, left justified and space filled.

Effect
For the purposes of this request, the software levels to which programs are attached may be divided into groups of 30 levels (30-59, 60-89 etc.). Only the levels within the group to which the named program belongs are modified. If no other program exists apart from the one named in the parameter block, then no change occurs; however, if there is an eligible program of higher priority than the named program, then the priorities of these two programs are interchanged.If there are more than two programs, a cyclic shift occurs. The program with the highest priority is re-assigned to the level of the program with the lowest priority in the group; each other program is re-assigned to the level of its next higher program in the group.

Returned Status
The following status codes are returned in A7 on completion of this request:

0	Completed
1	Unknown program name
2	Program not connected to a level
-1	Invalid address in A8.

LKM 14 - Attach a Device or a File

Purpose
To obtain exclusive use of a device or file via its filecode.

Calling Sequence
```
    LDK       A7,M
    LDKL      A8,N
    DATA      [-]14
    [DATA     L]
```

Where M is a Wait Flag with the following values:

M=0	Control is returned immediately to the requesting program if the device/file is unavailable.
M≠0	The program is suspended until the required device or file becomes available.
N	is the address of a word containing the filecode in bits 8-15. The filecode defines the device or file to be attached. A disc or a DAD may not be attached in this way.
L	is the address of a Scheduled Label Routine.

Returned Status
A7 contains one of the following codes when control is returned to the calling program:

0	The device or file has been attached.
-2	A8 contains an invalid address or the filecode is not assigned or an attempt is made to attach a DAD, a disc or a TDFM file.
-3	Device is already attached to another task. This status code only applies if A7 was zero when the request was made.
-4	The device to be attached is spooled.

Remarks
1) Using a Scheduled Label routine only affects the return address after the request has been activated.

2) For a re-entrant foreground program which is performing several tasks concurrently, the device or file can only be attached to one of the tasks at a time.

3) Attachment of a disc file is only valid within one machine.

LKM 15 - Detach a Device or File

Purpose
To detach a device or file which has already been attached to the requesting program.

Calling Sequence
```
    LDKL      A8,N
    LKM
    DATA [-]15
    [DATA     L]
```

Where N is the address of a word containing the filecode in bits 8 - 15.
L is the address of a Scheduled Label routine.

Returned Status
A7 contains one of the following codes when control is returned to the calling program:

0	Device or file detached.
-2	A8 contained an invalid address, or filecode not assigned or the device or file does not exist or is not attached to this program.
1	The filecode is not attached to a device or file.

LKM 17 - Get Time and Date

Calling Sequence

```
    LDK       A7,M
    LDKL      A8,N
    LKM
    DATA [-]17
    [DATA     L]
```

Where: M = 0 if time is to be returned in ASCII format.
        M ≠ 0 if time is to be returned in binary format.
        N = Address of a Time Block.

The Time Block is a 6-word area having the following format:

	A7=0	A7≠0
0	Day	Hours
2	Month	Minutes
4	Year	Seconds
6	Hours	Tenths of Seconds
8	Minutes	Fiftieths of Seconds
/A	Seconds	Pulse Rate.

This Time Block is completed by MAS before control is returned to the calling program.

The Pulse Rate is zero if the standard clock is used, otherwise it is the number of non-standard pulses per standard pulse as determined at Sysgen when the non-standard clock was generated.

Returned Status
When control returns to the requesting program A7 will contain one of the following codes:

0	OK
-1	A8 contained an invalid address.
1	A clock interrupt occurred during the processing of this request which caused a change of day to occur. The Day entry in the table is the previous day and will have bit 0 set to 1. The time however is correct. Another LKM 17 can be issued to get the correct date.
2	The date has not been initialized.
3	The clock has not been initialized.
4	Neither date nor clock has been initialized.

LKM 18 - Set Event

Purpose
To inform MAS that an event has occurred, so that the system can restart all programs that are suspended awaiting that event.

Calling Sequence
```
    LDKL     A8,N
    LKM
    DATA [-]18
    [DATA     L]
```

Where:	N	is the address of word 1 of an Event Block.
	L	is the address of a Scheduled Label routine. The scheduled label will be performed before the next sequential instruction in the calling routine when control is returned to the requesting program after completion of the LKM.

The 'Event Block' is a two word table having the following layout:

```
bits        0 1                                         15
           ___________________________________________
           |    Address of linked Event Block (only if L=1)  |
(A8)->     |E|L|                                         |
           -------------------------------------------
```

Where:	E	is the Event Bit and is set by the Monitor to indicate that the event has occurred.
	L	is zero for a single event and 1 for a chained event. If L is zero then word 0 is unused, since in this case the event block is the last or only block in the chain.

Effect
For a single event MAS sets bit 0 of the word whose address is contained in A8 (i.e. word 1 of the Event Block).

For a chained event MAS will set bit 0 of the word pointed to by A8 and pick up the address of the next event block from word 0 of this Event Block. The chain is followed until the chained event bit (L in the diagram above) is found to be zero.

Returned Status
The status code returned in A7 upon completion of this request is either:

0	Event bit(s) set.
or -1	Invalid address in A8.

Remark
The address in A8 is a 16-bit relative or logical address. Care must be taken that the address given in A8 is not coincidentally the same as that in the calling sequence of another event handling procedure in a different segment of the same machine, otherwise the result will be unpredictable.

LKM 20 - Connect a Program to a Level (Foreground Only)

Purpose

To connect a program to a software level; this is required before the program can be activated. This request can be used instead of the FCL CNL command.

Calling Sequence

```
    LDK      A7,N
    LDKL     A8,M
    LKM
    DATA     [-]20
    [DATA    L]
```

Where: N is the level number to which the program is to be connected. It must be numerically less than the number of the idle task and must not already have a program connected. For a re-entrant program the level number specifies the minimum software level to which any task performed by the program is to be connected. MAS will connect any new task to the lowest unconnected software level above this minimum.

: M is the address of a Program Name Block, which is a three word area containing the Name of the program to be connected, to a maximum of 6 ASCII characters, left justified and space-filled to the right. This program must have been previously defined and must not be already connected to a level.

: L is the address of a Scheduled Label routine.

Returned Status

On return to the requesting program A7 will contain:

0	Connection made
-1	Invalid address in A8
-2	Program is already connected
-3	Program is unknown
-5	Level too high.

LKM 21 - Disconnect a Program from a Level (Foreground Only)

Calling Sequence

```
    LDKL      A8,N
    LKM
    DATA [-]21
    [DATA     L]
```

Where: N is the address of a three word Program Name Block, in which the name of the program to be disconnected is entered in ASCII characters, left justified and space-filled to the right.

: L is the address of a Scheduled Label Routine.

Effect:

The program is disconnected from its level and also from any timers that may be connected to it.
When the program has not been connected to a level no action is performed and no error status is given.

Returned Status

When control is returned to the requesting program (which occurs when the request and any associated Scheduled Label routine has been completed), A7 may contain one of the following codes:

0	Disconnection performed or program not connected.
-1	Invalid address in A8.
-2	Program is active (running, in wait state or aborted).
-3	Program unknown.

LKM 22 - Wait for a Given Time (Foreground Only)

Purpose
To suspend a program until either a specified time has elapsed or an event bit in an event block or chain of event blocks has become set.

Calling Sequence
```
    LDKL      A8,N
    LKM
    DATA [-]22
    [DATA     L]
```

Where: L is the address of a Scheduled Label Routine.

- N is the address of word 1 of an Event Block having the following layout:

```
    bits       0 1 2 3 4                                   15
    word -2   | Address of linked Event Block (if L = 1)    |
    word 0    |E|L|                                         |     <-(A8)
    word 2    |0|NTIM|                                      |
    word 4    |     NC                                      |
```

In this table:

L	is set if this is one of a chain of event blocks, in which case the address of the next is given in word -2.
E	is the event bit.
NTIM	is the length of the wait time unit (bits 1-3, bit 0 is zero): 0 = hours 1 = minutes 2 = seconds 3 = 100 ms 4 = 20 ms 5 = 1 non-standard clock period.
NC	is the number of periods, defined by NTIM, which must elapse before the program is reactivated.

Effect
The issuing program is suspended until either the specified time has elapsed, or an event bit in the chain of event blocks (if any) has become set.

Returned Status
When the program suspension ends, control is returned to the scheduled label associated with the LKM 22 (if any) and upon completion of this, to the calling program. The status code returned in A7 when the program suspension ends may be one of the following:

0	Wait has ended.
/D	Invalid address in A8.
-2	Invalid timer number.
2	System dynamic area overflow.

LKM 23 - Assign a Filecode

Purpose:
To assign filecodes to devices, temporary disc files, catalogued files or other filecodes.

Calling Sequence

```
    LDKL      A8,N
    LKM
    DATA [-]23
    [DATA     L]
```

Where L is the address of a Scheduled Label Routine
N is the address of an Assign Block.

The format of the assign block depends on the type of assignment required:

1) Type 0 Assign a Filecode to a physical Device

bits	0	1	2 7	8 15
word 0	Assign Type			Filecode
1	Device name			
2	S	L	Device address	
3	Linenumber			

Where: Assign Type = 0 (in bits 0-7 of word 0)
Filecode occupies bits 8-15 of word 0.
Device Name is expressed as two ASCII characters; see Chapter 3.
Device Address is expressed in binary unless the S bit is 1, in which case MAS will assign the first free device of the required type which it encounters in its device tables.
For devices connected to AMA8 or ASCU4Z bit L can be set to 1, in which case the linenumber has to be specified in word 3: linenumber.

2) Type 1 - Assign a Filecode to a Temporary Disc File

bits	0	1 7	8 15
Word 0	Assign Type		Filecode
1	DAD code (/F0 - /FF)		
2	C	Nr of granules	
3	Not used		
4	Not used		
5	File type		

Word 0, bits 0-7: Assign Type = 1
Word 0, bits 8-15 = Filecode to be assigned.
Word 1 = File Code of the DAD on which the file is to be created, or zero (see below).
Word 2, bit 0 = Granule Type: = 0; consecutive granules.
= 1; non-consecutive granules.
Word 2, bits 1 - 15: No. of granules to be allocated to the file.
Default: 1 granule.
Word 5: File Type, in two ASCII characters; one of:

UF	User File
SC	Source Module
LM	Load Module
OB	Object File.

Notes: If the DAD Filecode (word 1) is entered as zero, the effect depends on whether the requesting program is background or foreground:

a) Background; the file will be created on the DAD specified on the BCL :JOB command.

b) Foreground; the file will be created on DAD /F0 or the defaultDAD set by the JOB command or LKM 73

: If the file is to be created by a Direct Write LKM 1, the number of granules must be allocated at Assign time.

3) Type 2 - Assign a Filecode to the latest Version of a Catalogued File

bits	0	1	2 – 7	8 – 15
Word 0		P	Assign Type	Filecode
1	DAD File Code (/F0 - /FF)			
2	File Name (characters 1-2)			
3	File Name (characters 3-4)			
4	File Name (characters 5-6)			
5	File Type			

Assign Type	= 2, but bit 1 may be set (giving an Assign type of /42), in which case MAS will set the Write Protect Flag in its tables and thus prevent other tasks writing to this file until the background job or foreground task has ended.
DAD Filecode	Either the DAD in which the file is contained or zero. If zero, then: a) for foreground programs: DAD /F0 or default DAD. b) for background programs: the DAD specified on the BCL :JOB is assumed to contain the file. The file is always searched in the current JOB userid, so the DAD filecode must contain zero or the current JOB DAD.
File Name	is contained in words 2-4 and consists of up to 6 ASCII characters, left justified and space-filled to the right.
File Type	Is one of the standard mnemonics; see above (Type 1 Assign).

Note: For foreground machines, the library of the directory of the first entry in the DAD catalogue is scanned to locate the file, the default DAD and Userid set by the JOB command or LKM 73 are taken into account.

4) Type 3 - Assign a Filecode to Another Filecode

0 – 7	8 – 15
Assign Type	Filecode 1
Filecode 2	

Assign Type = 3
Filecode 2 contains a filecode to which Filecode 1 is to be assigned, but note:

a) Filecode 2 must already be assigned and not by a Type 3 assignment, i.e. a filecode cannot be made equivalent to another which is itself equivalent to a third.

b) Filecode 1 must not be a permanent filecode of the batch machine, i.e. one declared by an SCL FCD command.

c) If Filecode 2 gets a new assignment, Filecode 1 automatically gets the same assignment.

5) Type 4 - Assign a Filecode to a Specified Version of a Catalogued File

The Assign Block is as for Assign Type 2, with an additional seventh word containing the version number, a value ranging from 0 (the latest version) to 7 for the oldest version. The only other difference is the Assign Type which becomes 4.

6) Type 5 - Assign a Filecode to Another Users Disc Catalogued File

	0 7	8 15
word 0	Assign Type	Filecode
1	DAD Code	
2	Filename (characters 1-2)	
3	Filename (characters 3-4)	
4	Filename (characters 5-6)	
5	Filetype	
6	Version	
7	Userid (characters 1-2)	
8	Userid (characters 3-4)	
9	Userid (characters 5-6)	
10	Userid (characters 7-8).	

- Assign Type = 5
- DAD Code The Filecode of the DAD on which the file is to be created. If this is zero, then:

  a) Foreground programs; DAD /FO or the default DAD is used.
  b) Background programs; the DAD specified on the BCL :JOB command or the DAD that the BCP evaluated as the JOB DAD. The DAD catalogue is then searched for the USERID specified in words 7 - 10 of the Assign Block. The User Directory is then searched for the Filename, Filetype and Version specified in Words 2 - 6.

- The Filetype is one of the standard mnemonics given for Assign Type 1.

first Userid in the DAD. The default is taken, when the first word of the Userid
field is left zero.

7) Type 6 - Assign a Filecode to a TDFM Descriptor File

	0 7	8 15
word 0	Assign Type	Filecode
1	DAD Filecode	
2	Filename (characters 1-2)	
3	Filename (characters 3-4)	
4	Filename (characters 5-6)	
5	Zero	
6	Zero	
7	Userid (characters 1-2)	
8	Userid (characters 3-4)	
9	Userid (characters 5-6)	
10	Userid (characters 7-8).	

Assign Type = 6
DAD Code = The Filecode of the DAD on which the file is to be created (see Assign Type 1).

Remarks

a) Foreground Machines:
If this Filecode has already been assigned (by an SCL FCD or FCL ASG command or an LKM 23 or LKM 33 request), that assignment is replaced. The user must ensure that no other program in this machine will require to use the old assignment for this filecode.

b) Background Machines:
The assignment only affects the JOB Filecode Table maintained by MAS. When the next BCL :JOB command is read, the JOB Filecode Table is reset to the values originally specified by means of the SCL FCD commands when the background machine was defined.

c) When a filecode is assigned, re-assigned or deleted, whether by FCL or BCP command or by LKM request, the system does not check whether or not it is being used.

d) Before assigning the filecode, the system will delete the old assignment if any, then try to make the new assignment. Thus when the assign cannot be done, the old assignment no longer exists, and the User has to issue a new request to re-establish the old assignment if necessary.

e) The assignment of the filecode remains unchanged until the next assignment is requested, either by another assign request or by a control command.

f) When an assignment is made to a TDFM file, only the descriptor file need to be specified. The subfiles (KEY and DATA files) are assigned automatically. The subfiles are searched via the volumenumber of the disc where they are created. This volumenumber is recorded in the descriptor file. When a user mounted two discs with the same volumenumber, the subfiles may be searched on the wrong disc, which can lead to 'strange' errors.

Returned Status

When control is returned to the requesting program one of the following status codes will be returned in A7:

Value	Meaning
0	Assignment made
1	Disc I/O Error
2	Dynamic Area Overflow; filecode table cannot be created.
3	DAD unknown
4	Device unknown or specified devicename-deviceddress not existing.
5	Required granules unavailable
6	Filename unknown
7	Second filecode unknown or already assigned by equivalence
8	Permanent Filecode must not be assigned
9	Invalid Assign type
/A	Userid not found in DAD catalogue
/B	Illegal Filetype (not SC, UF, OB or LM)
/C	Illegal Filecode (/C0-/CF or /F0-/FF)
/D	Filecode already assigned to an attached device or file or illegal address in A8.
/E	Too many filecodes (more than system default or MFC command value)
/F	Too many granules requested for the sector size (for non-consecutive files) or greater than 32K sectors (consecutive files) or specified number of granules negative.
/1D	Unauthorized assignment made to a file.
/8002	I/O pending on the filecode to be re-assigned (i.e. file code is in use)

/100	Userid of one subfile unknown
/101	Name of one subfile unknown
/102	Dynamic area overflow in System Machine
/103	One pack not on line
/104	DAD of one subfile unknown
/105	DAD of one subfile not assigned
/106	ECB not completely in user program
/107	Unknown filecode for DAD of descriptor
/108	File already assigned in another machine with non-identical DAD filecodes for subfiles
/109	File protected, but session is without security back-up
/110	File protected in back-out mode, but back-out file not on-line
/111	FCT entry of Disc of Descriptor not found
/112	File already assigned, but no entry in FCT for DAD of descriptor.
/113	Filecode was assigned to a still accessed file.

Note: Even if the assignment request is rejected, the old assignment for this job will have been deleted.

LKM 24 - Delete a Filecode

Purpose

To request the system to remove a filecode which has been previously assigned. The entries of the filecode are freed in the various tables of the system.

Calling Sequence

```
    LDKL      A8,N
    LKM
    DATA [-]24
    [DATA     L]
```

Where: L is the address of a Scheduled Label Routine
N is the address of a Delete Word. This is a single word having the following format:

0 7	8 15
Zero	Filecode to be deleted

Effect

a) Foreground Machines:
The filecode is removed from the Filecode Table for this foreground machine and in addition, if the filecode was assigned to a disc temporary file, the granules allocated will be freed.

b) Background Machines:
The filecode is deleted from the MAS Filecode Table for the current background job, but not from the MAS Filecode Table for the background machine. If the filecode was assigned by an SCL FCD command when the background machine was defined, this filecode will be restored when the next BCL :JOB command is read by the BCP.

If the filecode was assigned to a disc temporary file, the granules allocated will be freed for use by other files.

If the filecode was unknown, the system returns as if it was deleted.

If the filecode was a permanent one, it is set to delete, but the FCTe (file code table) is not freed.

Returned Status

When control is returned to the requesting program, A7 may contain one of the following codes:

Code	Meaning
0	Operation completed
1	I/O error while reading the file to free the granules (temporary disc file), however the filecode will have been removed.
/C	Invalid filecode (/C0-/CF)
/17	Filecode is used by input spool.
/8001	The filecode to be deleted is a DAD filecode with one or more logical files assigned to it.
/8002	The filecode is in use by some pending I/O.
-1	Invalid address in A8.

LKM 25 - Read Unsolicited Key-in

Purpose

To read input from the operator's console (filecode /EF of the system machine) without having to use an LKM 1, which would block the operator's console until the message has been keyed in. The message for an unsollicited key-in is input after a control panel interrupt and is transported by the system to the program, expecting the key-in. The program can either wait for the message (via a LKM 2) or handle it via a scheduled label.

Calling Sequence

```
    LDKL      A8,N
    LKM
    DATA [-]25
    [DATA     L]
```

Where: L is the address of a Scheduled Label Routine.
   -     N is the address of word 0 of an Event Block, the format of which is:

bits	0 1 2 15
Word -1	Address of next ECB in the chain (if L=1)
0	E L
1	Buffer Address for operator message
2	Maximum message length
3	Actual Message Length (set by MAS)
4	Reserved
5	Identity Characters

Where: E   is the Event Bit which is set by MAS when the key-in occurs.
   -     L   is the chained event-block indicator. Zero if this is the last or only block in a chain of Event Blocks; if set to 1, word -1 of the block contains the address of the next block in the chain.

   -     'Actual Message Length' in word 3 of the table will contain either the character count of the input message or zero. If zero, it indicates that an LKM 26 has been issued cancelling the Key-In request.

   -     The 'Identity Characters' in word 5 are those used in the operator KI message to remove the LKM 25 request. They serve to identify which LKM 25 the message refers to when a there are several LKM 25 requests outstanding.

Effect

The request is recorded by the system and placed in a queue.

The user may use an LKM 2 (Wait for an Event) to suspend his program until the key-in occurs, or he may use a Scheduled Label routine with the LKM 25. In this latter case, the scheduled label will be started as soon as the key-in occurs; in the meantime processing is not suspended.

Remarks

1) If the program issues several LKM 25 requests with the same identity characters, they will be serviced on a first-in first-out basis. As one is serviced, it is removed from the queue, word 3 of the associated event block is updated and the event count is decremented.

2) If word 3 of the event block is zero on return, it means that the request was cancelled by means of an LKM 26. In such a case, any scheduled label routine associated with the LKM 25 is started immediately.

3) An LKM 25 must not be issued by a re-entrant program, since the KI message does not identify the PCT required.

4) A task cannot Exit if it has an outstanding LKM 25 request; the request must first be cancelled, either by an LKM 26 or an operator`s KI command.

<u>Returned Status</u>

A7 = 0	Request recorded.
A7 = -2	Invalid address in A8 or in the Event Block
A7 = 2	Dynamic area overflow; request not recorded.

LKM 26 - Cancel 'Read Unsolicited Key-in' Request

Purpose

To cancel an LKM 25 request (Read Unsolicited Key-in). This may be necessary, for example, since the latter request can be issued asynchronously and if no key-in is received, the program will be unable to Exit.

Calling Sequence

```
    LDKL      A8,N
    LKM
    DATA [-]26
    [DATA     L]
```

Where:	L	is the address of a Scheduled Label routine.
-	N	is the address of an Event Block, the format of which is as described for the LKM 25 request.

Effect

The request is removed from the queue, the event count is decremented and word 3 of the Event Block is set to zero. The Scheduled Label routine associated with the LKM 25 is then performed.

Return Status

A7 = 0	Request cancelled.
A7 = -1	Invalid address in A8.
A7 = 1	No previous LKM 25 for this Event Block issued by this task.

LKM 27 - Load an Overlay Segment

This request is generated automatically by the LKE Standard Processor for overlaid programs.

Calling Sequence

```
    LDKL      A8,N
    LKM
    DATA      27
```

Where: N is the address of a Load Block, the layout of which is as follows:

	0 7	8 15
0	Event Byte	Ascendant No.
2	Load Address of the Segment	
4	Sector Address of the Segment	
6	Effective Length	

Returned Status

Upon completion of the request, control is returned with one of the following status codes in A7:

A7 = 0	Segment has been loaded
A7 = -1	Invalid address in A8
A7 = -16	I/O error during load, the program is aborted with abort code /12.

Remark

The sectors of the overlay segment are loaded from the original load module. Every time a new granule has to be accessed the GRANTB is read in. So some performance can be gained when the load module is a consecutive one, because in that case no GRANTB reading is to be performed.

LKM 28 - Set Timer (Foreground only)

Purpose
To cause an event to occur after a predetermined time.

Calling Sequence
```
    LDKL      A8,N
    LKM
    DATA      [-]28
    [DATA     L]
```

Where: L   is the address of a Scheduled Label Routine.
   -     N   is the address of word 0 of an Event Block, the layout of which is as follows:

```
          0 1    3 4                     15
Word -1  | Address of linked ECB (if L=1) |
      0  |E|L|                            |
      1  |0|NTIM |                        |
      2  |              NC                |
      3  |              RT                |
```

Where: E   is the Event Bit, set by MAS when the specified time interval has elapsed.
   -     L   is the `linked block′ indicator; it is zero if this is the last or only Event Block in the chain, otherwise 1.

   -     NTIM is the Timer Number, as follows:

0	=	hours
1	=	minutes
2	=	seconds
3	=	100 ms
4	=	20 ms
5	=	non-standard clock.

   -     NC   is the number of NTIM time periods required to elapse before the Event Bit is set.
   -     RT   is the `reset timer′ indicator. It is set to zero by MAS when the Event Bit is set and set to 1 by an LKM 29 (Reset Timer) request which specifies this Event Block.

Effect
After `NC′ cycles of `NTIM′ units of time have elapsed, MAS sets the event bit `E′ and the linked event bits in any chained blocks, as indicated by the value of L.

Remarks
1) The issuing task is not suspended when this request is recorded, but when the event occurs any scheduled label routine associated with the LKM 28 is started.

2) The issuing task may follow the LKM 28 with an LKM 2 (Wait for Event) request using the same event block. If both LKM`s use scheduled label routines, the following sequence of events occurs when MAS sets the event bit:
   a) The scheduled label for the LKM 28 is performed
   b) The scheduled label for the LKM 2 is performed
   c) The next sequential instruction in the issuing program is performed.

It can be seen that the event block may be used to define an OR of the timer and another operation, for example an I/O operation. This is particularly useful in data communications programming, to determine either that an I/O operation is complete or that the terminal concerned has timed-out. For instance, suppose that an LKM 1 (I/O request) is issued and the I/O event block is chained to another event block. An LKM 28 is then issued to signal the end of the timeout period and the event block of this LKM 28 is chained to the same event block as that to which the I/O request is chained.

This latter event block therefore defines an OR of the timer and the I/O requests. When either completes, the event bit will be set.

An LKM 2 is issued (specifying this common event block) when execution of the program reaches a point beyond which it cannot continue until either the transfer is completed or the terminal is to be considered inoperable. If the I/O operation is completed, then the scheduled label associated with the LKM 1 is performed; if the device times-out, then the scheduled label routine associated with the LKM 28 is performed. When control is returned to the calling program, therefore, one of these two scheduled label routines will have been performed.

3) When the LKM 28 request is recorded, the event count is incremented. It is decremented again when:
   a) the specified time has elapsed.
   b) an LKM 29 (Reset Timer) request is received before the time has elapsed.

In both cases the event bit will be set and any associated scheduled label will be started. The value of RT (Word 4 of the Event Block) indicates which of these two cases applies, since RT = 1 if reset by an LKM 29 and zero if the specified time has expired.

Returned Status
When control is returned to the next sequential instruction to the LKM 28, A7 will contain one of the following status codes:

0	The request is recorded.
/D	Invalid address in A8.
-2	The Timer Number (NTIM) is invalid.
2	System dynamic area overflow.
-4	Set timer request done by a middleground program.

LKM 29 - Reset Timer

Purpose

To cancel the effect of an LKM 28 (Set Timer) request.

Calling Sequence

```
LDKL     A8,N
LKM
DATA     [-]29
[DATA    L]
```

Where: L is the address of a Scheduled Label Routine.

- N is the address of word 1 of an Event Block, the format of whichis described under LKM 28 - Set Timer.

Effect

The event count is decremented, the event bit is set in word 1 of the Event Block and word 4 of the Event Block is set to 1. Any associated scheduled label routines are then performed before control returns to the instruction following the LKM 29.

Returned Status

A7 = 0	Reset performed
A7 = -1	Invalid address in A8
A7 = 1	No previous LKM 28 (Set Timer) request has been recorded for the specified timer.

LKM 30 - Queue Handling Request

Purpose
To allow the system to take over the routine maintenance of user queues of data areas. It relieves the user of the need to chain sequential message buffers, for example, or to rechain as processing is completed or new messages are received.

Calling Sequence
```
    LDK       A7,N
    LDKL      A8,M
    LKM
    DATA      [-]30
    [DATA     L]
```

Where: L is the address of a Scheduled Label Routine.
- M is the address of an Event Control Block.
- N is a function code indicating the service required. The function code may have the following values:
  - 1 Put an area into the queue.
  - 2 Get the next area in the queue.
  - 3 Cancel the `Get Next' request.
  - 4 Release the current area in the queue.

The format of the ECB pointed to by A8 varies according to the function requested.

1) Put an Area into the Queue

		0 15
ECB	0	Not Used
	1	Buffer Address
	2	Number of Queues
	3	Status
	4	QNAM1 (3 words)
	5	being the name of
	6	the first queue
	7	QNAM2
	.	.
	.	.
	.	.
	.	.
	N	QNAMn

Where: Word 1 contains the address of the buffer which is to be placed in the queue. These buffers are described below.

- Word 2 specifies the number of queues into which the buffer is to be placed.

- Word 3 contains a status code, returned by the system after processing the request. These are detailed below.

- Words 4-6 contain the first Queue Name as 6 ASCII characters, left justified and space-filled to the right.

- Words 7-N contains Queue Names 2-`n', indicating the names of the queues into which the buffer is to be placed.

2) Get the Next Area in the Queue

ECB	0	1	2–15
0	E	L	Event word
1	Buffer Address		
2	Not Used		
3	Status Code		
4	Queue Name,		
5	up to 6 ASCII		
6	characters.		

Where: E is the Event Bit, set when MAS has processed the request.
- L is the Link Indicator, set to 1 if this ECB is linked into a chain of ECB`s.

- Word 1: the address of the first word of user data. This is returned by MAS when E is set.

- Word 3 contains the status returned by MAS upon completion of the request (see below).

- Words 4-6 give the queue name from which the next data area is to be retrieved.

3) Cancel `Get Next' Request

ECB	0 – 15
0	Not used
1	Not used
2	Not used
3	Status Code
4	Queue Name for
5	which the Get Next
6	request was issued

The status codes returned by MAS when this request has been completed are described below.

4) Release the Current Area in the Queue

ECB	0 – 15
0	Not used
1	Buffer Address
2	Not used
3	Status Code
4	Not used
5	Not used
6	Not used

Where: Word 1 contains the address of word 0 of the buffer to be released.

- Word 3 will contain a status code inserted by MAS when the operation is complete. These status codes are described below.

Format of the Buffer Area

The memory areas to be placed in the queues handled by the LKM 30 requests have the following format:

```
    Word -2        Reserved for MAS and LKM 4
         -1   F    Length of this data area
          0   First word of user data            <-------------|
          1          |                                         |
          -          |                                         |
          -          |                                         |
          n   Last word of user data                           |
Entry for A   Pointer to this word in next buffer              |
Queue 1            in this queue                               |
          B   Pointer to start of User Data (word 0) --------->|
Entry for A'  as for A                                         |
Queue 2   B'  as for B                             --------->|
          C   Number of Queues (QCNT).
```

The buffer shown above assumes that the area has been placed in two queues, QUEUE1 and QUEUE2. Location C (QCNT) therefore contains 2.

The two words A and B are repeated as many times as necessary, so that there is one pair for every queue in which this area has been placed by the system.

The user must not change any of the data outside words 0 - n until the buffers are released (QCNT = 0), as these areas are used by MAS in the queue handling process.

All the user has to do, before issuing his first PUT request, is to ensure that there are enough words in each buffer area to contain two words for each queue entry, plus one word for the QCNT and two words at the beginning of the buffer.

In the diagram showing the buffer format, if:

F = 0	The buffer belongs to the dynamic area and must be released as soon as it has been processed.
F = 1	the buffer is not to be released even when it has been completely processed.

Processing

The buffers are chained and removed from the queues on a first-in, first-out basis. For each machine several queues may be used, each of which is identified by its name (unique within one machine) and entered into the system's control tables when the first request is made naming that queue. It is not necessary to declare it at the time the machine is specified.

Put in Queue Request (A7 = 1)

This request is used to chain the buffer to the last one already in the queue. If the queue name is not known to the system, MAS will create a new entry in its Queue Table. The user must enter the queue name in his ECB together with any other queue names to which this buffer belongs, and with the total count of queues.

This request is always processed immediately, and the calling program resumes processing as soon as the request is entered and the status code has been returned.

Get Next Area in the Queue Request (A7 = 2)
This request indicates to MAS that the previous area has been completely processed (either by the issuing program or another within this machine using the same queue name). The QCNT of the previous area is decremented, and if it reaches zero and if F is also zero it will be released. Then the next buffer will be obtained.

If the queue is empty, the request will be recorded (even if the queue name is unknown, in which case a new entry will be made in the system's Queue Table), but the issuing program will not be suspended. The user can check for the completion status by means of an LKM 2 (Wait) using the ECB or, better still, a scheduled label routine.

When the 'Get' request is recorded the event count is incremented; it will be decremented when the requested area is retrieved.

Cancel 'Get Next Area' Request (A7 = 3)
This is used to cancel the effect of the previous 'Get Next Area' request, and it also indicates that the current area is ready for release. The system will then decrement the queue count.

Release the Current Area in the Queue (A7 = 4)
This is used to indicate that the area last retrieved by a 'Get Next' request has been processed and is to be released. QCNT will be decremented.

Returned Status
The status codes returned by MAS in word 3 of the ECB have the following meanings:

0	Request completed.
-1	Invalid address in the control block.
-2	Buffer cannot be released (because, for example, it is a data area within a user program).
1	'Cancel Get Next' received. This status code is returned either to the scheduled label routine associated with the 'Get Next' request, or to the LKM 2 associated with the ECB used by the 'Get Next' request.
2	A 'Cancel Get Next' request has been issued, but the queue of 'Get Next' requests is empty.
3	The buffer is too short to contain all the chain pointers.
4	The 'Get Next' request has been queued (queue is empty).
5	Dynamic area overflow. Request not performed.
6	Current area has been released.
7	'Release Current Area' request invalid.

In A7, -1 is returned when A8 contained an invalid address or when A7 contained an invalid order (<1 or >4).

Remarks
The buffers belong to the user machine and are accessible by user programs. They can be acquired either by LKM 4 (Get Dynamic Buffer) requests or included as data areas within a program. The user may overwrite the data parts of the buffer areas (words 0 - n in the diagram above), but the other areas should not be overwritten until the buffer is released. The only exception is bit 0 of word -1 (bit 'F' in the diagram), which the user may set to 1 in order to prevent the automatic release of a dynamic area buffer when the queue count (QCNT) becomes zero.

LKM 31 - Cancel `Keep Control on Abort'

Purpose

To cancel the effect of the previously issued LKM 7 (Keep Control on Abort) request.

Calling Sequence

```
    LKM
    DATA      [-]31
    [DATA     L]
```

Where   L is the address of a Scheduled Label Routine.

Effect

MAS removes the `User Abort' label from the program`s PCT (Program Control Table) and returns to the program.

Returned Status

```
    A7 = 0    Request performed.
    A7 = /15  No previous LKM 7 was recorded.
```

LKM 32 - Set/Reset File Attributes

Purpose

To change the attributes of a file or the version number that will be used as the highest version in any `Keep File' requests.

Calling Sequence

```
    LDK       A7,N
    LDKL      A8,M
    LKM
    DATA      [-]32
    [DATA     L]
```

Where: L is the address of a scheduled label routine.

- N is a number defining the function required:

A7 = 0	Set specified bits to 1
A7 = 1	Reset specified bits to zero
A7 = 2	Set highest version number.

- M is the address of a File Descriptor Block, the format of which is as follows:

```
    Bits     0          7 8            15
    Word 0   |           | DAD filecode |
         1   | Userid (characters 1-2)  |
         2   | Userid (characters 3-4)  |
         3   | Userid (characters 5-6)  |
         4   | Userid (characters 7-8)  |
         5   |    Filename (char. 1-2)  |
         6   |    Filename (char. 3-4)  |
         7   |    Filename (char. 5-6)  |
         8   |         Filetype         |
         9   |MASK|           |Version  |
    Bits     0   3 4          12 13    15
```

Where: `DAD Filecode' (word 0, bits 8-15) is in the range /F0 - /FF.

- `Userid' is up to 8 characters, left justified and space-filled to the right.

- `Filename' is up to 6 characters, left justified and space-filled.

- `Filetype' specifies the type of the file (SC = source, etc.).

- `MASK' (word 9, bits 0-3) is a pattern of bits representing the file attribute flags which it is required to set or reset (see LKM 40, Keep File).

- `Version' (word 9, bits 13-15) represents a file version number which is to be used as the highest version number, for each Keep File request processed, for the whole library.

Effect

If A7 = 0	The bits corresponding to the MASK are set to 1.
If A7 = 1	the bits corresponding to the MASK are reset to zero.
If A7 = 2	the version number (in word 9, bits 13-15) is treated as the highest version number when processing Keep File Requests for the whole library.

Returned Status

A7 may contain one of the following values when control is returned to the calling program:

0	Request performed.
-1	Invalid address in A8.
1	I/O error.
3	DAD not assigned.
6	File not catalogued in the directory.
9	Filecode not assigned to a DAD.
/A	Unknown Userid.
/1D	For background only: the specified Userid is not that of the JOB. Only the owner or the System Manager (JOB USID=SYSTEM) may modify the file attributes.
/1E	The calling program is not the Librarian processor (Background only).
/1F	Invalid order in A7.

LKM 33 - Check and Assign a Filecode

Purpose

To check whether a filecode has been assigned and if possible to re-assign the filecode to a temporary disc file. This request is used by various processors to check whether the temporary output file has been opened or not. For example, the Assembler uses it for checking and assigning temporary object output file (/D5).

Calling Sequence

```
    LDKL     A8,N
    LKM
    DATA     [-]33
    [DATA    L]
```

Where: L is the address of a Scheduled Label Routine.
N is the address of an Assign Block having the following layout:

Bits	0 - 7	8 - 15
Word 0	1	Filecode
1	DAD Code (/F0-/FF)	
2	Nr and Type of Granules	
3	Not Used	
4	Not Used	
5	File Type.	

Word 0:	-	the left byte is set to 1 (assign type)
	-	the right byte contains the filecode to be checked.
Word 1	-	the filecode of the DAD on which the file will be assigned (if not assigned); if zero, then the DAD of the JOB command will be used (background) or DAD code /F0 or the default DAD (foreground).
Word 2	-	bit 0 indicates the type of granules: bit 0 = 0, consecutive granules are allocated. bit 0 = 1, non-consecutive granules are allocated.
	-	bits 1-15 contain the number of granules required (if this is entered as zero, 1 granule is allocated).
Word 5	-	The file type; one of the standard mnemonics: SC Source File; OB Object File; LM Load Module; UF User File.

Effect

If the filecode specified in the Assign Block is already assigned, or has been assigned to a temporary disc file and EOF sector has not been written to this file, then the request has no effect. Or the old assignment is deleted and MAS reassigns the filecode according to the parameters given in the assign block.

Returned Status

When control is returned to the requesting program one of the following status codes will be found in A7:

0	Assignment done.
1	I/O error on disc.
2	Dynamic area overflow; entry cannot be created in the filecode table.
3	DAD unknown or invalid.
5	Disc overflow: requested granules unavailable.
9	Invalid assign type (not equal to 1).
/B	Invalid file type.
/C	Invalid filecode (background machine only).
/E	Filecode table overflow.
-1	Invalid address in A8.

LKM 34 - Check and Write an EOF on a File

Purpose

To ensure that an EOF mark has been written on a file and that the file is closed.

Calling Sequence

```
    LDKL      A8,N
    LKM
    DATA [-]34
    [DATA     L]
```

Where -	L	is the address of a Scheduled Label routine
-	N	is the address of a word containing the filecode of the relevant temporary file in bits 8-15.

Effect

If an EOF mark has already been written to the file, nothing happens. If an EOF mark has not yet been written, one is written into the next sector of the file.

Returned Status

Upon return to the calling program, A7 will contain one of the following status codes:

0	Request performed
1	Disc I/O error
4	Filecode unknown (not assigned)
/C	Invalid filecode (not assigned to a disc file)
/11	Catalogued file
-1	Invalid address in A8.

Remarks

This request can be used instead of an LKM 1 (type /22) when it is not known if the file is closed.

LKM 35 - Get a Program's Characteristics

Purpose
To obtain information about the calling program.

Calling Sequence
```
    LDKL      A8,N
    LKM
    DATA      [-]35
    [DATA     L]
```

Where: L is the address of a Scheduled Label Routine.
N is the address of a table to which the system will copy the Program Control Table (PCT) and other relevant details such as the Job Parameter Table (background machines) or machine details (foreground machines). This table must be long enough to contain all the information transferred; 64 words in the case of foreground machines and 76 in the case of the background machine.

The format of this table is as follows:

Word	Bit	Contents
0	0-15	Address of next PCT in the chain, or zero.
1-3		Program Name (6 ASCII characters, left justified and padded with spaces).
4		Program start address.
5-7		Reserved.
8	0	1 if system program, otherwise zero.
	1	1 if Exit has been issued but not completed.
	2	1 if Scheduled Labels outstanding.
	3	1 if Read-Only program.
	4	1 of Swappable Program.
	5	1 if Memory-Resident Program.
	6	1 if Middleground Program.
	7	1 if Re-entrant Program.
	8	1 if Program swapped.
	9	1 if Background Program.
9	0	1 if Program not connected.
	1-15	Software Level to which it is connected.
10		Address of ECB on which main program is waiting.
11-13		Addresses of Scheduled Labels to be performed when this task exits.
14		Address of the activation queue for this program.
15		Event count for this program.
16-19		Reserved.
20	0-7	Zero.
	8-15	DAD filecode where the program resides.
21	0	1 if program load module is consecutive.
	1-15	Address of GRANTB sector of the load module; unused if the file has consecutive granules.
22		The number of pages in the region containing this program.
23-24		Reserved.
25		Program Load Address.
26-27		Reserved.
28		Address of the Parameter Block for LKM's 7, 25 and 57.
29		For a re-entrant foreground program, the maximum number of simultaneous activations allowed.
30		Reserved.
31-33		Relevant to disc-resident programs only.

For the foreground machine the table continues as follows:

Word	Bit	Contents
34		Reserved.
35-37		Machine Name (6 ASCII characters, left justified).
38-44		Reserved.
45		Machine Status:
	0	Zero if event occurred.
	1	Zero.
	2	1 if System machine.
	3	1 if Foreground machine.
	4	1 if Background machine.
	5	1 if SM command received and BYE not received.
	6	1 if filecode /E0 defines an interactive device (TY, CRT).
	7	1 if catalogued procedure being used.
	12	1 if no middleground program allowed.
	13-15	Return Code used by FCL.
46		MCT address.
47-52		Reserved.

For the background machine the table continues thus:

Word	Bit	Contents
34-37		Userid (8 ASCII characters, left justified and padded with spaces).
38		Maximum execution time (TIME).
39		Maximum number of print lines (PRINT).
40		Maximum number of punched records (PUNCH).
41		Current execution time.
42		Current number of printed lines.
43		Current number of punched records.
44	0-7	Filecode of the DAD containing the current program.
	8-15	JOB DAD filecode.
45		Directory address of the Userid in JOB DAD.
46	0	1 if JOB Userid is SYSTEM.
	2	EOJ of current job received.
	4	1 if /E0 is assigned to an interactive device (TY or DY).
	5	1 if filecode /02 is assigned to the same device as /E0: input commands from /E0 do not need to be listed.
	6	1 if /02 is assigned to the same device as the error recovery file. It is unnecessary to send the error message to both.
	7	1 if the filecode /E0 is assigned to the same device as the error recovery file. In interactive mode the processor does not have to reprint the erroneous command on the error recovery file before reading the correction.
	9	1 if pure batch processing mode. Zero if interactive mode; error recovery filecode (word 53) contains the filecode where the correct command can be obtained.
	10	1 if BCP is currently executing.
	11	1 if the LIB processor is currently executing.
	12	1 if a catalogued procedure is being used. 0 if the command input is /E0.
46	14-15	00 if DUMP=NO) 01 if DUMP=ALL) Set by the BCP. 11 if DUMP=PROG)
47	8-15	Value of the 'ABCD=' parameter on :STP command.
48		Address of the current BCL command in BCP CCT.
49	0	1 if the load module for BCP has consecutive granules.
	1-15	Disc address of the BCP.

50	0	1 if the program to be activated by the BCP has consecutive granules.
	1-15	Disc address of the program to be activated by the BCP.
51	0-7	Abort code of the program just completed.
	8-15	Exit code for the last BCL command.
52	0-7	Filecode used by the BCP to read the current BCL command.
	8-15	`CODE=' parameter value of most recent :STP command.
53	0-7	Filecode defining the ERR device for recovery of command errors.
	8-15	Highest exit code received in this step.
54		System directory address (USERID=SYSTEM, DAD=/F0).
55		No. of pages required to load the BCP.
56		No. of pages required to load the program.
57-75		Save area used to transmit the registers from a background program to the BCP when an abort occurs. Contents are: P-register Program Status Word General registers A1 to A14 Floating-point registers FR1 to FR3.

Returned Status

A7 = 0	Request complete.
A7 = -1	Invalid address in A8.

LKM 36 - Signal the Start of a JOB (BCP Only)

Purpose
This request is used exclusively by the BCP when a BCL :JOB command is received, to request MAS to initialize the JOB tables.

Returned Status
A7 = -3 The requesting program is not the BCP.

LKM 37 - Keep Control on Floating Point Error

Purpose
To avoid the automatic abort of a program if a floating-point error occurs.

Calling Sequence
```
    LDKL     A7,A
    LDKL     A8,N
    LKM
    DATA     [-]37
    [DATA    L]
```

Where: A is the address within the calling program to which control should be returned by MAS when the error occurs.

- L is the address of a Scheduled Label routine.
- N is the adress of a four-word control block in which MAS will return details of the program's status when the error occurred. The format of this block is:

word	0	Status
	2	PSW
	4	Instruction Counter
	6	Contents of A8 when the error occurred.

Effect
Control is returned to the address specified in A7 in the event of a floating point error occurring.

Returned Status
When control is returned to the calling program, or to any associated scheduled label routine, upon completion of this request A7 will contain one of the following status codes:

0	Request recorded.
-1	Invalid address in A7 or A8.
2	System dynamic area overflow; request not recorded.
/16	Floating point hardware option not present on the bare machine.

Remarks
1) Apart from A8, no other register will have been changed when the specified error exit is taken.

2) If a program issues more than one LKM 37, only the last applies.

3) The effect of this request is cancelled explicitly by the LKM 38 request, or implicitly when an error occurs, in contrast to LKM 7 (Keep Control on Abort).

4) The program must cancel this request before exiting, using the LKM 38 request.

LKM 38 - Cancel 'Keep Control of Floating Point Error'

Calling Sequence

```
    LKM
    DATA      [-]38
    [DATA     L]
```

where L is the address of a Scheduled Label routine.

Effect

The effect of any previously issued LKM 37 is removed; after this request has been recorded by the system, the normal error action will occur when a floating point error is detected.

Returned Status

A7 = 0	Done
A7 = /15	No previous LKM 37 has been issued by this program, or it has already been cancelled.

LKM 39 - Get Machine Options

Purpose

To obtain information about the resources of the bare machine.

Calling Sequence

```
    LKDL      A8,M
    LKM
    DATA      [-]39
    [DATA     L]
```

where: L is the address of a Scheduled Label Routine.
M is the address of a Bare Machine Block (BMB) provided by the user program.

Effect

Upon return, the relevant fields in the Bare Machine Block (BMB) will have been filled as follows:

Word	Bit(s)	Contents
0	0-15	No. of characters in the BMB, excluding this word.
1	0	Set to 1 if floating point instructions are used by more than one program or task, i.e. if MAS must save floating point registers. This bit is altered by the SCL commands FON & FOF.
	11	Set to 1 if the machine has the floating point option.
2		No. of free pages in the bare machine.
3		Max. No. of programs in the bare machine.
4-10		Reserved.
11		Device type, e.g. CR, LP etc.
12	0-7	Filecode, if disc device.
	10-15	Device Address.
13		No. of lines per page if device is LP, or Volume Number if disc.
14		Reserved.

Words 11 - 14 are repeated as many times as required to describe all the devices in the machine. The last word is zero.

Returned status

A7 = 0	Block filled
A7 = -1	A8 contained an invalid address
A7= 1	Block too small to contain all information, however it is filled till it was full.

LKM 40 - Keep File

Purpose

In the background machine, this request is used to prevent a temporary disc file from being released and to enter it into the user directory. In the foreground machine, the purpose is the same; the request may also be used in place of the FCL KPF command to keep a temporary disc file created by another program of the same foreground machine.

Calling Sequence

A8 is loaded with the address of a File Block, the format of which is described below, then:

```
LKM
DATA      [-]40
[DATA     L]
```

where L is the address of a Scheduled Label Routine.

The format of the file block is as follows:

```
    bits  0                 7 8          15
Word 0   |                   |  Filecode   |
     1   |)                                |
     2   |)            Userid              |
     3   |)                                |
     4   |)                                |
     5   |)                                |
     6   |)           File name            |
     7   |)                                |
     8   |            File Type            |
     9   |A|B|C|D|          Zero           |
    bits  0 1 2 3
```

The Filecode (word 0, bits 8-15) must have been assigned to a temporary disc file.

The Userid (of 1-8 ASCII characters) exist in the DAD in which the temporary file has been created. If the first word of the Userid is zero, then for background machines, the Userid of the JOB command is used, while for foreground machines, the first Userid in the DAD or the default DAD-Usid is used.

The File Type (word 8) is a two-character ASCII code such as:

UF	User File
SC	Source File
LM	Load Module

but note that only the LIB processor may issue an LKM 40 for a load module.

Word 9 contains flag bits (bits 0-3), shown in the table as A, B, C and D; the significance of each bit is:

A = 1 if the file is to be unshared.
B = 1 if the file is to be write protected.
C = 1 if the file is to be a system file.
D = 1 if the file is to be an invisible file.

Returned Status

A7 = 0	File catalogued.
1	I/O Error.
2	System Dynamic Area Overflow.
4	Unknown Filecode.

9	Filecode not assigned to a disc file.
/0A	Userid unknown.
/10	Directory overflow: no more entries can be created.
/11	This filecode is assigned to a catalogued file.
/12	Invalid File Type.
/13	Keep file was requested by a background program for a file on a DAD which was not the :JOB DAD
/14	There was no Userid on the DAD (foreground only)
/1D	All versions of the file to be kept existed already. As no new version could be created, the file with the highest version number had to deleted, but this can only be done by the owner or by the system user which was not the one that issued this keep file.
-1	Invalid address in A8.

Remarks

1) When the file type is OB (object), the system does not alter the entry in the directory; it assumes that this has already been created. Thus, this file type must not be used in this request by user programs, but only by the LIB Processor.

2) The file to be catalogued is implicitly the latest version of the file (version 0). The version numbers of all other entries in the directory, having the same filename and file type, are incremented and any version greater than the maximum, created in this way, is automatically deleted and its granules released.

3) If a scheduled label routine is specified with this request, it will be executed before returning to the instruction following the LKM 40. However, all I/O operations will be allowed to terminate before this return is made.

4) If a status code of 1 (I/O error) is returned, it is possible that some or all of the version numbers have been incremented and that the old maximum version has been deleted.

5) If a status code /10 (Directory full) is returned all existing version of the file are incremented, but there will be no version 0 as the file to be kept could not be inserted in the directory.

LKM 41 - Delete a File

Purpose

To remove a catalogued file from the directory. The granules belonging to the file are released and can be re-used immediately, so the user must not delete a file that is still in use.

Calling Sequence

```
    LDKL      A8,B
    LKM
    DATA      [-]41
    [DATA     L]
```

where: L is the address of a scheduled label routine.
- B is the address of a File Block, the layout of which is as follows:

```
              0             7 8               15
Word 0       |               |   DAD Filecode   |
     1       | )                                |
     2       | )            Userid              |
     3       | )                                |
     4       | )                                |
     5       | )                                |
     6       | )           File Name            |
     7       | )                                |
     8       |          File Type Code          |
     9       |          Version Number          |
```

- Userid is 1 to 8 ASCII characters, left justified and padded with spaces. The default value is the Userid of the JOB (background), or the first Userid of the DAD or the default DAD-Usid (foreground). Only the LIB processor may specify the Userid.

- File Name is 1 - 6 ASCII characters, left justified and padded with blanks, and is the name of the file to be deleted.

- Filecode is that of the DAD in which the file is located. If zero, /F0 is assumed for the foreground machine, while the DAD of the JOB command is used for the background machine. This entry must be zero for user programs; only the LIB Processor may specify this filecode.

- File type code can be:

UF	User file
SC	Source file
LM	Load Module
OB	Object File (LIB processor only)
EF	File managed by TDFM.

- 'Version number' is in the range 0 - 7, and specifies which version of the named file is to be deleted.

Effect

A file with the specified filename, type and version number will be deleted from the directory of the Userid and DAD specified. If these are not specified, the appropriate default values will apply. Any other entries in the directory having the same filename and type, and with a version number greater than the deleted file, will have their version numbers decremented by 1.

Remarks

1) It is the user's responsibility to ensure that the file is not in use, since the system will delete it even if the machine still has one or more filecodes assigned to it.

2) An implicit wait is caused by LKM 41; control is not returned to the issuing program until all I/O operations on the directory have been completed. Any scheduled label specified in the LKM 41 requests will also be completed before this return is made.

3) Only the LIB Processor may specify the DAD filecode and Userid (words 0 - 4 of the file block); for all user programs these entries should be zero.

Returned Status

One of the following status codes will be returned in A7 upon completion of this request:

-1	Invalid address in A8.
0	Operation completed.
1	Disc I/O error: the file may or may not have been deleted and other version numbers may have been updated.
2	No Userid in the DAD.
6	File not found.
9	Words 0 - 4 of the file block were non-zero and the calling program is not the LIB Processor.
/12	Illegal File Type.
/1C	File Type is 0B, but the calling program is not the Librarian.
/1D	System flag is set for this file; only the Librarian may delete it.
/1E	DAD filecode set, but the calling program is not the Librarian.

For the LIB Processor and Foreground programs:

3	DAD Unknown.
9	Filecode is not assigned to a DAD.
/A	Userid unknown.

LKM 42 - Initialise BCP (BCP Only)

Purpose
The request is used by the BCP to initialize the Monitor tables prior to reading a BCL command. It is not available to user programs, and a status code of -3 is returned in A7 if the request is issued by any such program.

LKM 43 - Allocate Permanent Granules (LIB Only)

Purpose
This request is used only by the Librarian, to allocate permanent granules on a DAD (for example, for a Userid Directory). It is not available to user programs, and a status code of /1E is returned in A7 if the request is issued by any such program.

LKM 44 - Release Permanent Granules (LIB Only)

Purpose

The request is used only by the Librarian, to release granules allocated by LKM 43. It is not available to user programs, and a status code of /1E is returned in A7 if the request is issued by any such program.

LKM 45 - Dump Memory

Purpose

Used by any program to print the contents of the program's registers and areas of memory, on the device defined by filecode /02 of the user machine.

Calling Sequence

```
    LKM
    DATA      45
    DATA      first address to be dumped
    DATA      last address to be dumped
```

Effect

The effect is as for the SCL/FCL DUM commands; the memory area specified and the contents of all registers are dumped onto the device with filecode /02.

Notes

1) There is no returned status; if either address is invalid, or the second is less than the first, then no dump is performed.
2) No scheduled label may be used with this request.
3) No registers are modified.
4) If the postmortem dump option was not selected at system generation time, this LKM will not work.

LKM 46 - Abort the Program

Purpose

Used by the main program or a scheduled label routine to stop execution and to ignore all outstanding scheduled label routines.

Calling Sequence

```
    LKM
    DATA      46
```

Effect

For a foreground program, unless it has issued an LKM 7 (Keep Control on Abort), a message is sent to filecode /01 of the foreground machine in order to allow the user to request a dump. The program is abandoned, currently active tasks will not exit and no new tasks will be started.

For the background machine, if an LKM 7 has been issued, control is transferred to the user Abort Label and an Abort Code of 06 is placed in the Abort Control Block. It is then the user's responsibility whether or not to abort by issuing an LKM 3 (Exit) with an exit code and post mortem dump flag in A7. If he has not issued an LKM 7, the post mortem dump flag is set and a dump is performed according to the DUMP parameter on the BCL RUN or BCL Processor Call command which activated the program.

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

LKM 47 - User Written Request

Purpose

To allow the user to incorporate his own LKM requests into the system without the necessity of recreating the Monitor's LKM table.

Calling Sequence

```
    LDKL      A7,N
    LKM
    DATA      [-]47
    [DATA     L]
```

where: L is the address of a Scheduled Label routine.
- N is a one word entry point reference in which:
  bits 0-5 contain the relative entry point within the segment.
  bits 6-15 contain the segment number, within the D:MASG file of the System DAD, in which the user-written LKM is stored.

These segments are independent load modules which the user incorporates into the MAS segments file D:MASG of the DAD SUPERV. They are executed in the system machine under the program name X:MASG. They are loaded into the System Transient Area and entered at the entry point specified in A7.

Upon entry, the following registers will have been set up:

A1	Address of the PCT for X:MASG.
A2	Return address to X:MASG, used to call a subsequent segment.
A5	The User PCT address.
A6	The user Scheduled Label address (if any), otherwise zero.
A14	Zero.

Effect

The calling program is suspended and its event counts are incremented. Before returning to the calling program, the user-written LKM must:
- Decrement the event counts.
- Return the status to the calling program.
- Remove the suspension from the calling program by resetting bit 14 of word 7 of the calling program's PCT.
- Start the scheduled label (if any).
- Exit.

See P800 Programmer's Guide 3, Vol. IV: Trouble Shooting Guide, for a complete description of the system tables.

Remarks

1) The size of each module is a maximum of 2039 words; if the user-written LKM exceeds this, it must be contained in more than one segment.

2) The next segment may be loaded from the previous one by:
   a) setting A3 to contain the relative entry point in bits 0-5 and the segment number in bits 6-15.
   b) issuing an ABR A2 instruction (A2 contains the address of X:MASG)

3) A segment may contain more than one entry that can be called, indicated by the value given in the first 6 bits of A7. The segment must start with the addresses of each entry in the segment, because X:MASG branches indirect to the fist location of a segment, according to the entry number.
   X:MASG (the disc resident segment loader) checks whether the called segment has already been loaded, if so, it is not loaded again from the segment file. However, this means that a user written LKM must be coded re-usable, that is, each variable must be initiated, because one cannot be sure that the value assigned to the variable when it was declared has not been changed.

Insertion of a segment in the D:MASG file

After having linked the load module containing the new segment, the load module must be moved to the temporary load module file /D6 (or /L):

```
LIB
CDF fnam,LM,ONAM=/L,ODAD=/Fx
LEN
```

Then the D:MASG file has to be assigned and a utility has to be invoked (OVLGEN). This utility inserts the new segment in the D:MASG file:

```
ASG FCOD=/40,FNAM=D:MASG,USID=MASUP,DAD=/F0
RUN PROG=OVLGEN
RELOC,SEGNR: 300,xx
```

The OVLGEN program asks for the relocation (always 300) and the segment number, i.e. the place in the D:MASG file where the new segment is to be inserted. The segmentnumber must not be an existing one in the MAS system. Because the number of segments used by MAS differs from release to release, no value can be given here. The segmentnumber consists of two hexadecimal numbers without preceding slash (/).
When the MAS system resides on a CDC disc, the D:MASG file must be copied to a DAD called D:MSEG. This is done as follows:

```
ASG FCOD=/40,FNAM=D:MASG,USID=MASUP,DAD=/F0
RUN PROG=COMASG
```

Running the COMASG utility, DAD filecode /F2 must be assigned to a DAD on the disc on which the D:MSEG DAD resides.

Useful MAS modules

Some standard modules reside on the library MASOB in Usid MASGEN on the starter pack with might be of some help, programming an LKM 47.

A:USIN

The module A:USIN initiates the registers for the LKM. It must be declared as external in the LKM routine and called using A15:

```
    CF   A15,A:USIN
```

The module performs the following actions:

- Load A5 with the PCT (program control table) of the calling program.
- Load A7 and A8 with the values of A7 and A8 of the calling program.
- Load A12 with the MCT (machine control table) of the machine where the calling program resides.
- Load A13 with the CVT (communication vector table) address.
- Load A14 with a stack address. The stack is the module C:STK, available in the MASOB library. Subroutines used by the LKM should be called, using A14.

The MMU of the calling program is loaded.

A:USR1

This module performs the end actions for the LKM. It must be declared as external and called with an absolute branch:

```
    ABL  A:USR1
```

The module performs the following actions:

- restore of the calling program registers.
- start of the scheduled label (if any).
- unsuspend the calling program. The program was suspended (on LKM) by the LKM interrupt routine I:LKM.
- decrement the event counts (incremented by I:LKM).
- store the MMU of the calling program into its PCT.
- exit from X:MASG (the program under which the user written LKM was running)

C:SVA7

When the user written LKM must return a status to the calling program, it must be loaded into the calling program's A7 register.
The LKM may store the status into the external C:SVA7 (1 word), which is loaded into the user's A7 in A:USR1. If no error occurred C:SVA7 should be set to 0.

R:CUAD

This module checks whether an address supplied by the user is in the calling program. It must be declared as external and its calling sequence is:

```
        LDKL   A4,retadd
        <load A1 with the addres to be checked>
        ABL    R:CUAD
retadd  RF     INVADD
```

R:CUAD expects A4 to be loaded with the return address. When the addres to be checked is not in the calling program, R:CUAD returns to the address loaded in A4. When the address is valid, R:CUAD returns to the address in A4 plus 2.Register A1 must contain the address to be checked.
An invalid address should cause setting of A7 to -1 (/FFFF) in the calling program.

R:DMAS and R:DMLS

When the LKM needs a buffer, e.g. for reading a disc buffer, it can be reserved in the LKM's segment, but it can also be asked in the System Dynamic Area.
To ask a buffer in the System Dynamic Area, use the module R:DMAS:

```
        LDKL   A1,length
        CF     A15,R:DMAS
```

The length in A1 must be supplied in characters, the address of the obtained buffer is returned by R:DMAS in A1. If no buffer is available, A1 is set to zero. The programmer should take care not to use space outside the buffer, because this will abort the system with error code /09.

When the obtained buffer is not needed anymore, R:DMLS has to be called to release it. This releasing is obligatory as not relasing might cause System Dynamic Area overflow. R:DMLS is called:

```
        LD     A1,bufad
        CF     A15,R:DMLS
```

The address in A1 must be an address returned by a previous R:DMAS call. If not, the system will abort with error code /09.
Both R:DMAS and R:DMLS must be declared as externals.

Core resident LKM's

Described above is the implementation of a disc resident LKM. Also core resident LKM's can be included in the system. In this case, the module T:LKM has to be adapted. This module is delivered on the starter pack. To include a core resident user LKM, an entry in T:LKM must be filled with:

```
    DATA     /8000
    DATA     usrlkm
```

where: /8000 is an indidcation the a core resident LKM is involved.
usrlkm is the address of the user written core resident LKM.
The core resident LKM should return via the Dispatcher (external M:DIS1) with in A4 zero, or, when there is nothing new to dispatch via the external MLRRTN. Both routines, M:DIS1 and MLRRTN expect 8 registers (A1-A8) in the A15 stack, which are stored there by the general LKM interrupt routine. So these registers may be used in the LKM. without saving.

LKM 48 - Assign a Linecode

Purpose
To assign a filecode to a datacommunications device or to another linecode.

Calling Sequence
```
    LDK      A7,0
    LDKL     A8,A
    LKM
    DATA     [-]48
    [DATA    L]
```

where: L is the address of a scheduled label routine.
A is the address of an assign block, the layout of which is as follows:

```
    bits  0 1          7 8                   15
  word 0   |Assign Type  | Linecode to be assigned|
       1   |Device Name or linecode to which it  |
           |is to be assigned                    |
       2   |S|       Device Address              |
       3   |           Line Number.              |
```

Description of Assign Block
Word 0: 'Linecode' specifies the filecode to be assigned to the line.
'Assign Type' can be:
0: Assign a linecode to a physical device.
3: Assign a linecode to another linecode.

Assign Type 0
Word 1: 'Device Name' is one of the following:

S2	= SLCU2
S4	= SLCU4
A4	= ALCU4 or ALCU2
A8	= AMA8A or AMA8C
NO	= No Device.
H1	= HLVCU
L6	= LSM16
SA	= SALCU
HV	= HLVCUZ
SZ	= SLCUZ

Word 2: 'Device Address' should be given in binary. If S = 1, the filecode is assigned to the first operable device of the requested type encountered in the Monitor tables (i.e. in the order declared at SYSGEN), whatever its address and line number.

Word 3: This location is used only for AMA8 devices, and gives the line number in binary, from 0 to 7.

Assign Type 3
The linecode given in Word zero is made equivalent to that in Word 1. Words 2 and 3 are unused, but must still be reserved within the calling program.

Returned Status

A7 =0	Assignment OK
-1	Invalid address in A8
2	System Machine dynamic area overflow
4	Unknown Device
7	2nd linecode is not assigned (type 3)
9	Invalid assign type
/C	Invalid linecode (=0)

Note

Before assigning the linecode, the system first deletes the old assignment if any. If this request fails, therefore, the old assignment will have been deleted.

See P800M Datacommunications User Manuals.

LKM 49 - Delete a Linecode

Purpose
To delete the linecode of a datacommunications device.

Calling Sequence
```
    LDK       A7,0
    LDKL      A8,P
    LKM
    DATA      [-]49
    [DATA     L]
```

Where: L   is the address of a scheduled label routine.
        P   is the address of a parameter word, thus:

```
    bits  0          7 8                15
         |            |linecode to delete |
```

Returned Status
On return to the calling program, A7 will contain one of the following values:

0	Linecode deleted.
-1	Invalid address in A8.
/C	invalid linecode (=0).

See P800M Datacommunications User Manuals.

LKM 50 - Submit a Job to the BCP (Foreground Only)

Purpose
To start a job running in the background machine by issuing a request from a user program. The JOB to be submitted must first have been stored by a program in a catalogued DFM file in a DAD also known to the background machine.

Calling Sequence
```
     LDK       A7,N
     LDKL      A8,JDB
     LKM
     DATA      [-]50
     [DATA     L]
```

where: JDB is the address of a block defining a job description file on disc.
- L is the address of a scheduled label routine.
- N = 0: Submit a Job
  = 1: Get Information about a Job.

Job Description Block Format

Byte bits	Contents 0 – 7	8 – 15
0		DAD filecode (default is /F0)
2	)	
4	(	
6	(Userid (default is first userid of the DAD)	
8	)	
/A	)	
/C	(Filename	
/E	)	
/10	Filetype (must be UF)	
/12	Version number (default is 0).	

Effect
On receipt of this request, the JOB referred to is retrieved from the file described in the job description block and entered in the job queue for the BATCH machine.

Remarks
This LKM is actioned via the spooling system for the card reader, and so it is essential that spooling is initiated for the card reader before this request is given. The address of the submitted file and the control block address of the DAD, where it resides are transferred to the DAD D:SPCR and from there submitted to the spooling queue in the System Dynamic Area. Because the submitted file itself remains on the user's DAD, the file must not be altered before the submitted JOB is finished.
In the spool DAD, one sector is reserved for submitted job, so the maximum number of jobs in the submit queue is (D:SPCR sectorlength-10)/4.

If in A7 the value 1 has been specified, status always a status unequal to zero is returned.

Returned Status

The following status codes are returned in A7:

0	OK
-1	Invalid Address in A8
2	User DAD Not Assigned
4	I/O Error on User DAD
6	User ID Not Found
8	Unknown File
10	Card Reader Not Spooled
12	I/O Error on DAD D:SPCR
14	Job Queue Overflow.
16	Job completed or unknown (N = 1).
18	Job running (N = 1).
20	Job queued (N = 1).

LKM 51 - Spooling Request (BCP Only)

Purpose
This request is used by the BCP to initiate or end spooling activity. It is illegal from a user program.

Calling Sequence

```
    LDK       A7,N
    LKM
    DATA      51
```

where: N is 0 at the start of a spooled JOB and 1 at the end of a spooled JOB.

Effect
At the start of a JOB the BCP zeroises A7 and initiates the JOB.

At the end of a JOB, A7 is set to 1 and the disc space previously occupied by the spooled JOB is freed. A status code is returned in A7 on completion of this request, as follows:

0	OK, or no spooled device
1	EOB encountered
-3	LKM 51 was not issued by the BCP.

LKM 52 - Send or Receive a Letter

Purpose
To provide a channel of communication between programs in the same or different machines.

Calling Sequence

```
    LDK      A7,F
    LDKL     A8,M
    LKM
    DATA     [-]52
    [DATA    L]
```

where: L is the address of a scheduled label routine.

F is a function code which may have one of the following values:
- 1 Send a letter
- 2 Receive a letter.

M is the message area from which the letter is transmitted (function code 1) or into which it will be transferred (function code 2). The format of this area varies with the function code.

Function Code 1 (Send):

Byte	Contents
0	Unused
2	Mailbox Name (2 ASCII characters)
4	Length of letter in characters
6	Unused
8 on	Message to be sent.

Function Code 2 (Receive):

Byte	Contents
0	E
2	Mailbox Name (2 ASCII characters)
4	Length of message block
6	Length of letter received, in characters (set by MAS)
8 on	Message of length specified in byte 6.

- E is the event bit; set when the letter is received.

Note: If the message length exceeds the value specified by the user in byte 4, it will be truncated.

Effect

Function Code 1 (Send)
If the mailbox has not already been created when an LKM 52 request is issued, MAS creates it in the system machine dynamic area. The user does not create the mailbox. As many mailboxes (each known by its unique name of 2 ASCII characters) may be created as the system dynamic area has space for.

As letters are received by a mailbox, they are placed in a queue and requests for these letters are serviced on a first-in first-out basis. Similarly, if a mailbox is empty when requests for letters are received these requests are also queued and serviced in the same way, as and when letters are received.

Function Code 2 (Receive)
The request is serviced when either:
- the request reaches the head of the queue, or
- there is at least 1 letter in the mailbox.

Synchronisation can be effected using a scheduled label routine or by an LKM 2 (Wait for Event) request on the event bit `E'.

<u>Returned Status</u>
On completion of this request, one of the following status codes is returned in A7:

0	Request completed
-1	Invalid address in A8
-2	Incorrect length (zero or negative)
-3	Dynamic area overflow
-4	Invalid function code in A7.

LKM 53 - Conditional Dump

Purpose

To provide a means of obtaining a dump of the user program area or buffers when the bits set in a mask word match the corresponding bits in a flag word. Both mask and flag word are provided by the user.

Calling Sequence

```
    LKM
    DATA      53
    DATA      M
    DATA      F
    DATA      L
    DATA      A
    DATA      B
```

where:	M	is the address of the user's mask word
	F	is the address of the user's flag word
	L	is a 6-character label, printed at the head of the dump
	A	is the address of the first location to be dumped
	B	is the address of the last location to be dumped.

Effect

If the MASK word is non-zero, a logical AND operation is performed between it and the flag word. If the result is equal to the MASK, a dump is performed.

Returned Status

If the MASK word is zero, or if any of the addresses are invalid, no dump occurs. No status code is returned.

Remarks

If the postmortem dump option was not selected at system generation time, this LKM will not work.

LKM 54 - Request or Release a Device (Background Only)

Purpose
This depends on the function code loaded into A7 before the request is issued, but the request provides the same facilities as the BCL commands ROI, REQ and REL.

Calling Sequence
```
    LDK      A7,F
    LDKL     A8,P
    LKM
    DATA     [-]54
    [DATA    L]
```

where: F is the order, in the range 0 - 2, with the following significance:
- 0 Request operator intervention
- 1 Request a device
- 2 Release a device.

P is the address of a parameter block of 38 words, with the following layout:

bits	0 – 7	8 – 15
0		Filecode
1	Device Name (only if A7 = 1)	
2 on	72 characters of message to be sent to the operator.	

and L is the address of a scheduled label routine.

Effect
This varies with the function code:

Function Code = 0 (Request Operator Intervention):
A message having the following format is sent to the operator's console:

    `message' dnda, THEN RS, PLEASE

dn is the device name
da is the device address.

The program is then suspended awaiting operator action, and is resumed when the operator enters `RS' on the console.

If the device is a spooled output device, the message is not sent immediately, but is output when the file is unspooled.
For spooled device the message is:
    'message' dnda, THEN TYPE IN SP DNDA,C

Note: This request is only actioned for non-disc devices.

Function Code = 1 (Request a Device):
A message having the following format is output on the operator's console:
    MOUNT ON `message' dnda, THEN RS, PLEASE

The program is suspended awaiting operator action, until `RS' is entered on the console. The device is attached to the background program and the filecode is assigned to it.

Note: This request is used for MT and cassette.

Function Code = 2 (Release a device)
The device is detached from the batch program, but the program is not suspended. The following message is sent to the operator's console:

    DISMOUNT `message' dnda, PLEASE

Returned Status
One of the following status codes is returned in A7 upon completion of this request:

0	OK
-1	Invalid address in A8
-4	Function code in A7 invalid
-5	Invalid `dn' specified (function code 1 - not MT or TK)
-6	No free operable device (function code 1)
1	I/O error (spooled device)
2	Dynamic area overflow
7	Filecode not assigned (function codes 0 or d 2)
8	Filecode not assigned to a physical device, or assigned to NO device (function codes 0 or 2)
9	Filecode assigned to a disc (functions 0 or 1), or filecode not attached to this program (function 2).

Furthermore the status codes returned by the LKM 23 (assign) apply when order code 1 is specified.

Remarks
The filecode in bits 8-15 of word 0 of the parameter block is the one already assigned to the device (functions 0 and 2), or which is to be assigned by MAS (function code 1).

Trailing blanks in the message are not output and unprintable characters are blanked.

LKM 55 - Semaphore Handling (Foreground Only)

Purpose
To provide a method of synchronising foreground tasks.

Calling Sequence
```
    LDK     A7,F
    LDKL    A8,S
    LKM
    DATA    [-]55
    [DATA   L]
```

where: F is a function code having one of the following values:

0	Declare a semaphore
1	Cancel a semaphore
2	Decrement a semaphore by 1 (P-operation)
3	Increment a semaphore by 1 (V-operation).

S is the address of a one- or two-word block, containing in the first word the 2-character semaphore name. In the case of function code 0 (Declare a Semaphore), a second word is used containing the initial value of the semaphore.

L is the address of a Scheduled Label Routine.

Effect
1) Function Code 0 (Declare a Semaphore):
This request must be made before a semaphore can be used.

The system sets up a semaphore block in the system dynamic area and, in the case of the first semaphore declaration for a machine, enters its address in location /36 of the MCT. Subsequent semaphores are chained from this first one, thus:

```
MCT      S1       S2       S3
-----    -----    -----    -----
         @S2      @S3      0        Chain Address
         S1       S2       S3       Semaphore Name
         +1       0        -2       Semaphore Value
@S1      0        0        @PCT1    PCT chain of suspended
-----    -----    -----    -----     programs.

                  PCT1          PCT2
                  -----         -----

                  @PCT2         0
                  -----         -----
```

S1, S2 and S3 are three semaphores; S3 has 2 suspended programs queued.

2) Function Code 1 (Cancel a Semaphore):
The semaphore block is removed from the system dynamic area.

3) Function Code 2 (Decrement a Semaphore): The P-operation.

The current value of the semaphore is decremented by 1. If the new value is negative, the calling program will be suspended and put in a queue of all such programs which have issued request type 2 causing the semaphore value to go negative. The absolute value of the semaphore indicates the number of programs suspended.

4) Function Code 3 (Increment a Semaphore): The V-operation.

The current value of the semaphore is incremented by 1. If the new value is negative or zero (i.e. there were tasks suspended and queued), the first task in the queue will be reactivated.

The Use of Semaphores

The most usual application of semaphores is in cases of competition for resources where deadlock or over-allocation may occur. For example, if two tasks both request a resource held by the other and are suspended waiting for it to become free, neither can free the resource which it holds and which is required by the other. The result is deadlock. Similarly if two independent tasks allocate a resource according to the following algorithm:

```
If resource A is free,
then assign it.
```

and they are both competing for the same resource, they may both coincidentally get a TRUE response to the first statement and proceed to attempt to allocate the resource.

These difficulties can be overcome by using P- and V- operations on a semaphore. Suppose, in the second example above, that resource A is allocated a semaphore of initial value 1, then a safe algorithm would be:

```
P(s) (i.e. decrement the semaphore).
If resource A is free,
then assign it.
V(s) (i.e. increment the semaphore).
```

Any competing task, attempting to perform a P-operation on S after the first task has performed one, will be suspended until the V-operation is performed.

If a number of independent foreground tasks have a pool of buffers available, the Get Buffer and Release Buffer operations can be effectively synchronised by declaring a semaphore of initial value equal to the number of buffers available.

If the sequence adopted by each task is:

```
P(s)
Get Buffer
Process
Release Buffer
V(s)
```

then as soon as the semaphore becomes negative following a P(s) operation, the issuing task will be suspended until another task releases a buffer and performs a V-operation upon the semaphore.

Returned Status

The following status codes are returned in A7 upon return to the issuing program:

Function Code 0 (Declare a Semaphore):

0	Completed
-1	Invalid address in A8
1	Invalid function code in A7
2	Semaphore already declared.
3	Dynamic area overflow.

Function Code 1 (Cancel a Semaphore):

0	Completed
-1	Invalid address in A8
1	Invalid Function code
5	Unknown Semaphore
4	Semaphore cancelled, but there was a program suspended.

Function Code 2 (Decrement a Semaphore - P Operation):

0	Completed
-1	Invalid address in A8
1	Invalid function code in A7
5	Unknown semaphore
6	Semaphore has been cancelled (the program is not suspended).

Function Code 3 (Increment a Semaphore - V Operation):

0	Completed
-1	Invalid address in A8.
1	Invalid function code in A7.
5	Unknown semaphore.

LKM 56 - Request/Release Pages (Foreground Only)

Purpose
To allow programs within any one foreground machine to acquire and share named working areas.

Calling Sequence
```
    LDK      A7,C
    LDKL     A8,B
    LKM
    DATA     [-]56
    [DATA    L]
```

where: L is the address of a scheduled label routine.
- B is the address of a parameter block.
- C is the request order code, which may have one of the following values:

0 or /8000:	request pages
1:	release pages
2:	connect pages to the program.

Request Pages (A7 = 0 or /8000)
If request order 0 is used, an immediate return to the program is made with a status code in A7. If /8000 is used and there are insufficient pages to satisfy the request, then the program will be suspended until the required number of pages becomes available.

The parameter block addressed by A8 has the following structure:

0	Page group area name (2 ASCII characters)
2	Relative address of page area
4	Page size (/1000)
6	Number of pages to be allocated
8-38	16-word copy of MMU registers.

The name of the group of pages to be allocated must be unique within a machine.

The relative address is the logical address within the program to which the pages will be connected. It must be a page boundary (0, /1000, /2000 up to /F000). Only the 4 leftmost bits are significant.

The page size is fixed at /1000 (i.e. 2K words), but the default value is zero.

The last 16 words of the table are filled by MAS and will contain, on return to the calling program, a map of the adjusted MMU registers, showing the new page structure.

Remarks
- If the user sets the 'relative address' entry in the table to -1, the system will search for the first unused MMU register, starting with zero. If no free register exists, the pages will be connected to virtual address zero.

- Allocated pages are assigned to consecutive virtual addresses in ascending order, modulo 32K.

- If request order /8000 is used and the program is suspended due to insufficient free pages, an entry is created in the system area to preserve the coordinates of the pages to be allocated. All suspended programs are reactivated as sufficient pages become free.

Returned Status
One of the following status codes will be returned in A7 on completion of this request:

0	Request Successful
-1	Invalid Address
1	Invalid Request Order
2	Already Declared
3	Dynamic Area Overflow
4	Insufficient Free Pages
5	Invalid Number of Pages (more than 15).

Release Pages (A7 = 1)
For this request the parameter block addressed by A8 consists of one word, containing the name of the pages block to be released.

Note: after this request has been processed, any attempt by this or any other program to access the released pages will result in a page fault error.

Returned Status
The following code is returned in A7 on completion of this request:

0	Satisfactory completion
-1	Invalid Address
1	Invalid Order
6	Unknown page group name.

Connect Pages to a Program (A7 = 2)
The parameter block addressed by A8 has the following format:

0	Page Group Name
1	Relative Address
2	Check
3	Number of Pages to be Connected.

where:
- Page Group Name is the name of a group of pages, which must already have been allocated by Request Pages (order 0 or /8000).
- Relative Address is the address within the requesting program, to which the pages group is to be connected.
- Check may have the values 0 or 1. If 0, the Relative Address in Word 1 must be the same as the Request Pages request; if 1, the addresses may differ. If Check is 1 and Relative Address is -1, the system searches for the first free MMU register and connects from there. The actual address is returned in Word 1. Connection is refused if not enough MMU registers are free.
- Number of Pages to be Allocated must be the same as in the Request Pages request.

These pages must have been previously allocated by another program (using request order 0 or /8000).

Returned Status
Upon return to the calling program, A7 will contain one of the following status codes:

0	Satisfactory completion
-1	Invalid address in A8
1	Invalid order
4	Insufficient free pages
5	Invalid number of pages (differs from that given at request time).
6	Unknown page group name.
7	Invalid relative address (differs from that given at request time).

LKM 57 - Connect/Disconnect a Secondary Load Module

Purpose
This request enables the calling program to modify its MMU tables to include or exclude the named secondary load module. While the module is included it may be accessed by the calling program.

Calling Sequence
```
    LDKL      A8,A
    LKM
    DATA      [-]57
    [DATA     L]
```

where:
- L   is the address of a scheduled label routine.
- A   is the address of a six word control block with the following lay-out:

word	0 – 15
0	SLM name
1	to be connected
2	load address
3	SLM name
4	to be disconnected
5	not used

One can connect (word 0#0) or disconnect (word 3#0) a secondary load module. When both word 0 and 3 are unequal to zero, first the secondary load module specified in words 3 and 4 is disconnected and then the secondary load module in words 0 and 1 is connected.
For the connection of a secondary load module, word 2 must contain its load address. This load address must be equal to the address specified during Link Edit of the secondary load module.

Effect
a) Connect (word0#0)
   The MMU tables of the calling program are modified to allow it to access the secondary load module area. The calling program must disconnect the secondary load module after use. Connection is refused if the MMU pages to which the module must be connected are not free.
b) Disconnect (word 3#0)
   The MMU tables of the calling program are modified to free the pages allocated to a previously connected secondary load module. The pages may then be re-allocated to the same or a different secondary load module.

Remark
It is dependent on the type of the program, which pages can be used to load a secondary load module.Background programs are loaded from page 0 onwards, disc resident foreground programs are also loaded from page 0 onwards, but from page 15 downwards, pages for the CMA are reserved. Memory resident foreground progams are loaded just before the CMA, which again occupies the highest pages

Returned Status

On return to the calling program, A7 may have one of the following values:

0	successful.
-1	invalid address in A8.
1	secondary load module not loaded.
2	the secondary load module refers to the ascendant (i.e. the calling program), while the ascendant is a memory resident foreground program. Referring to the ascendant is only allowed, when the ascendant starts at page 0.
3	dynamic area overflow.
4	page(s) already allocated.
5	page(s) already disconnected.

LKM 58 - Wait Multiple (Foreground Only)

Purpose
To enable the user to restart a program when a specified proportion of a total number of events have occurred.

Calling Sequence
```
    LDK       A7,0
    LDKL      A8,B
    LKM
    DATA      [-]58
    [DATA     L]
```

where:	L	is the address of a scheduled label routine.
	B	is the address of the multiple event control block, the layout of which is as follows:

bits	0	1 … 15
word 0	E	Event Word
1	Total Number of Events (N)	
2	Number of Events Awaited (P)	
3	Reserved	
4	Reserved	
5	Address of 1st Event	
6	Address of 2nd Event	
.	-	
.	-	
.	-	
.	Address of Nth Event	
.	P words, reserved for MAS to insert the addresses of the P ECB's	
.		
.		

Effect
The program is placed in a `multiple wait state'. When P events out of a specified N events have occurred, it will be restarted.

Remarks
- The range of P can be expressed as $1 \leq P \leq N$.
- This LKM can only be used in the program`s main sequence.
- A scheduled label routine, if specified, will be started when at least P events have occurred.
- Bit 0 of the multiple wait block will not be set by the system.
- P and N have a maximum value of 64. This limit is imposed by the system in order to reduce the danger of dynamic area overflow, while not imposing too great a restriction on the user.
- When P events already have been occurred, control is immediatily returned to the calling program.

Returned Status
A7 may have one of the following values on return to the calling program:

0	Normal termination
-1	Invalid address in A8
2	The request is issued by a scheduled label
3	Too many elementary events (N > 64)
4	P > N
5	Dynamic area overflow.

LKM 60 - Load/Delete a Secondary Load Module

Purpose
This request loads a secondary load module into memory, or deletes it when it is no longer required.

Calling Sequence
```
    LDK      A7,N
    LDKL     A8,A
    LKM
    DATA     [-]60
    [DATA    L]
```

where:

L	is the address of a scheduled label routine.
N	is the order:
	0 load the secondary load module
	1 delete the secondary load module.
A	is the address of a 5-word (for load) or 3-word (for delete) block, with the format:

0-1	Secondary Load Module Name
2	Not used
3	DAD Filecode
4	Type of Page

The secondary load module name is four ASCII characters, left-justified and space-filled; the module is catalogued in the first Userid of the DAD specified in Word 3.

The DAD filecode specifies the DAD from which the secondary load module will be loaded. It need not be supplied if the module is to be deleted.

The Type of Page is two ASCII characters, `R ′ or `W ′:

`R ′:	the module is read-only;
`W ′:	the module may be modified during execution, so it may be accessed by only one primary load module at any one time.

The type of page need not be specified if the secondary load module is to be deleted.

Effect
a) Load (A7 = 0)
The user must load a secondary module using this request, before connecting it to a primary load module using LKM 57. Loading is not required if the secondary load module is already in memory and has not been deleted.

b) Delete (A7 = 1)
The secondary load module is deleted from memory, if it is not connected to any primary load module.

Returned Status

a) from Load:

A7 = -1	invalid address in A8
0	successful
1	invalid order in A7
2	DAD unknown
3	DAD sector length greater than 512 characters
4	secondary load module already loaded
5	unknown program
6	I/O error
7	secondary load module is segmented
8	too few free pages
9	secondary load module too long
/C	dynamic area overflow
/D	type of page invalid.

b) from Delete:

A7 = -1	invalid address in A8
0	successful
2	secondary load module already deleted
3	secondary load module still connected to a program.

LKM 62 - Check if DAD assigned

Purpose
To check whether a DAD to be used is assigned in some machine. This LKM is used by the Librarian processor to check whether a DAD to be deleted is still in use.

Calling sequence

```
    LDKL     A8,N
    LKM
    DATA     [-]62
    [DATA    L]
```

Where: L is the address of a scheduled label routine.
- N is the address of a nine word bock, the layout of which is as follows:

bits	0 15
word 0	DAD filecode or zero
1	)
2	) DAD name
3	)
4	Disc filecode
5	Not used
6	)
7	) Machine name
8	)

Where: DAD filecode is specified when a check is needed to see whether the specified DAD is assigned to this given filecode. If zero is specified MAS checks whether the DAD is assigned to any (DAD) filecode.
If zero is specified, on return this word contains the filecode to which the DAD is assigned.
- DAD name is the name of the DAD to be checked.
- Disc filecode is the filecode of the disc where the DAD resides.
- Machine name is the name of the machine where the DAD is assigned.

Effect
The specifed DAD is searched in the filecode tables of all machines. If it is found, a status in A7 is returned and the name of the machine is stored in words 6-8 of the control block.

Remark
If the DAD is assigned in more than one machine, or assigned to more than one filecode, only the first occurrence of the DAD is recorded in the control block.

Returned status

A7 = 0	No (not that) filecode is assigned to the DAD in any machine.
A7 =-1	Control block out of the program.
A7 =-2	Word 4 does not contain a disc filecode (/C0-/CF)
A7 =-3	Word 0 does not contain a DAD filecode (/F0-/FF) nor zero.
A7 = 1	The DAD is assigned in the Batch machine.
A7 = 2	The DAD is assigned in a Foreground machine.
A7 = 4	The DAD is assigned in the System machine.
A7 = 8	The DAD is assigned in the machine from which the LKM is issued.

In the latter four cases, the machine name is stored in words 6-8 of the control block and the filecode to which the DAD is assigned is stored in word zero.

LKM 63 - Set Date and Clock

Purpose

To change the date and time as recorded currently in the system.

Calling sequence

```
    LDKL      A8,M
    LKM
    DATA      [-]63
    [DATA     L]
```

Where: L is the address of a scheduled label routine.
- M is a control block containing the new Date and Time, with the following layout:

word	bits 0 — 15
0	Day
1	Month
2	Year
3	Hour
4	Minute
5	Second

The date values must be specified in ASCII format, the time values in binary format. A check is made, whether the date and time are valid.

Remark

Great care should be taken, using this LKM. It affects the whole system and changing date and time can disturb the activation of programs connected to timers in any machine.

Returned status

A7 = 0	Date and time changed.
A7 =-1	Control block out of the program.
A7 = 1	Not existing date.
A7 = 2	Not existing time.
A7 = 3	Value of date does not contain decimal ASCII characters.

LKM 70 - Interface FCL and Middleground processor

Purpose
To read the last command given to the FCL task in the current machine.
This LKM is used by some standard progcessors like NOD, INC, etc. to read the FCL command with which the processor is called.The length of the command is given in word 0, the command itself in the next words (max. 36) of an area, of which the address is given in A8.

Returned status

A7 = 0	Command moved.
A7 =-1	Area out of the program.

LKM 71 - Assign a filecode to a DAD

Purpose
To assign or re-assign a DAD filecode (/F0-/FF) to a DAD, residing on a disc, specified by its filecode (/C0-/CF) or its packnumber.

Calling sequence

```
    LDKL     A8,M
    LKM
    DATA     [-]71
    [DATA    L]
```

Where: L is the address of a scheduled label routine.
- M is the address of a seven word block with the following layout:

bits	0 — 15
word 0	DAD filecode
1	Pack-
2	number
3	Disc filecode
4	)
5	) DAD name
6	)

Where: DAD filecode is the filecode to be assigned to the DAD. If it already exists, the old assignment is deleted.
- Packnumber is the volume number of the disc, where the DAD resides in 4 ASCII hexadecimal characters.
- Disc filecode is the filecode of the disc where the DAD resides. If both packnumber and disc filecode are specified, only the disc filecode is examined.
- DAD name is the name of the DAD to be assigned.

Effect
The DAD is assigned to the specified filecode. If the filecode was already assigned, the old assigned is deleted, however if the LKM is used in the Batch machine, it only affects the current JOB. After End-Of-Job the old assignment is reset.

Remarks
- The DAD of the current Job (in Background) or DAD /F0 (in Foreground and Background may not be re-assigned.
- On return both disc filecode and packnumber are stored in the control block.
- No files may have been assigned to a DAD to be re-assigned, nor programs may have been loaded from it.

Returned status
On return A7 contains one of the following status:

A7 = 0	Assignment done.
A7 =-1	Control block out of the program.
A7 = 1	I/O error, reading the VTOC, or the Bittab of the DAD.
A7 = 2	System dynamic area overflow.
A7 = 3	DAD does not exist on the specified disc.
A7 = 6	Packnumber does not consist of 4 hexadecimal characters, or no disc in the system has the specified packnumber.
A7 = 7	Disc filecode is not /C0-/CF or DAD filecode is not /F0-/FF.
A7 = 9	Disc filecode not assigned, or not assigned to a disc.
A7 =/C	Filecode to be assigned exists already but has not been assigned to a DAD.

A7 =/D	Try to re-assign the Job DAD or DAD /F0.
A7 =/E	Too many filecodes assigned.
A7 =/F	DAD cannot be re-assigned, some programs are loaded from it.
A7=/10	Disc filecode is assigned to a Data Floppy disc.
A7=/8001	Filecode cannot be re-assigned because some files ave been assigned to it.

LKM 73 - Set default DAD and Usid (Foreground only)

Purpose

To change the default DAD and Userid from DAD /F0, first Userid to a user defined DAD and Userid.

Calling sequence

```
    LDKL     A8,M
    LKM
    DATA     [-]73
    [DATA    L]
```

Where: L is the address of a scheduled label routine.
M is the address of a control block with the following layout:

```
    bits        0                  15
word 0         |   DAD filecode     |
     1         | )                  |
     2         | )   Userid         |
     3         | )                  |
     4         | )                  |
```

Where: DAD filecode is the filecode of the DAD where the new default Userid resides.
When Userid is not specified, the first Userid in the DAD is taken.
The name of the first Userid is returned in the control block.

Returned status

A7 = 0	Done.
A7 =-1	Control block out of the program.
A7 = 1	DAD filecode is invalid (not /F0-/FF) or not assigned to a DAD.
A7 = 2	The DAD does not contain a Userid.
A7 = 3	The specified Userid does not exist in the DAD.

LKM 74 Read the default DAD and Userid (Foreground only)

Calling sequence

```
    LDKL      A8,M
    LKM
    DATA      [-]74
    [DATA     L]
```

Where	L	is the address of a scheduled label routine.
	M	is the address of a five word area.

Effect
On return of this LKM, the area M contains the DAD filecode and the Userid, which are default in the current Foreground machine. On return, the layout is the same as for LKM 73.

Returned status

A7 = 0	Done.
A7 =-1	Area out of the program.

LKM 75 Dump system area

Purpose
To move a part of the system area to the user program.

Calling sequence
```
    LDK       A7,0
    LDKL      A8,M
    LKM
    DATA      [-]75
    [DATA     L]
```

Where:	L	is the address of a scheduled label routine.
	M	is the address of a control block with the following layout:

```
    bits       0           15
word 0         | ECBRB      |
     1         | ECBLEN     |
     2         | ECBEB1     |
     3         | ECBEB2     |
```

where: ECBRB is the receiving buffer address.
ECBLEN is the length of the area to be moved.
ECBEB1 is the most significant bits of the emitting buffer address. To be used in future, must be 0.
ECBEB2 is the least significant bits of the emitting buffer address.

Effect
The system area, addressed by ECBEB2 and ECBLEN is used to the user area, from address ECBRB on.

Remarks
Entering this LKM, A7 and ECBEB1 must be 0.

Returned status

A7 = 0	Area is moved.
A7 =-2	On entering this LKM, A7#0.
A7 =-1	Control block out of the program.
A7 = 1	Address in ECBRB is not in the user program.
A7 = 2	ECBEB1 is not 0.
A7 = 4	Error in length. The length is negative, or ECBRB+ECBLEN is greater than /FFFF or ECBEB2+ECBLEN is greater than /FFFF.

LKM 76 Abort another program in the same machine (Foreground only)

Calling sequence

```
    LDKL     A8,M
    LKM
    DATA     [-]76
    [DATA    L]
```

Where: M points to a three word block,containing the name of the program to be aborted.

L is the address of a scheduled label routine.

Effect

The specified program is aborted with abort code /16.

Returned status

```
    A7 = 0    Program aborted.
    A7 =-1    Program name block out of the program.
    A7 =-2    Program not known in this Foreground machine
```

LKM 77 Accept an attention key

Purpose

To interrupt output on a device connected to the AMA8 or the ASCU4Z.

Calling sequence

```
    LDKL      A8,M
    LKM
    DATA      [-]77
    [DATA     L]
```

Where	L	is the address of a scheduled label routine.
	M	is a one word ECB, containing in bits 8-15 a filecode assigned to a physical device for which an attention key request can be issued.

Effect

The attention key request is put in a queue. On receipt of an attention key (the ESC key) the event bit of the ECB is set and the scheduled label (if any) is started. Attention keys are serviced on LIFO (Last-In-First-Out) basis. An attention key request can be cancelled via LKM 78.

Returned status

A7 = 0	Attention key request accepted
A7 =-1	ECB out of the program.
A7 = 1	Filecode not assigned, assigned to NO device, or not assigned to a physical device.
A7 = 2	System dynamic area overflow.
A7 = 3	The filecode is not assigned to a device for which an attention key request is possible.

LKM 78 Cancel the last attention key request

Purpose
To undo the last issued LKM 77

Calling sequence
```
    LDKL      A8,M
    LKM
    DATA      78
```

Where	M	is a one word ECB containing a filecode assigned to an attention key device.

Effect
The last issued attention key request is removed from the queue, the event of the attention key request is set and its scheduled label (if any) is started.

Returned status

A7 = 0	Request done.
A7 =-1	ECB out of the program.
A7 = 1	Filecode not assigned, assigned to NO device or not assigned to a physical device.
A7 = 3	Filecode not assigned to an attention key device.
A7 = 4	There was no attention key request recorded by this program for this device.

LKM 79 Close a spool file (Foreground only)

Purpose
To close a spooled file so that it can be unspooled.

Calling sequence

```
    LDKL     A8,M
    LKM
    DATA     [-]79
    [DATA    L]
```

Where L is the address of a scheduled label routine.
M is a one word ECB, containing a filecode assigned to a spooled file in bits 8-15. The first bit of the control word is an event bit which will be set on completion of the LKM.

Effect
An EOF record is written to the spooled file. The file is put in the unspool queue and is output when it is on the top of this queue.

Returned status

A7 = 0	spooled file closed.
A7 =-1	control word out of the program.
A7 = 1	LKM issued by a Background program.
A7 = 2	no spool command (operator command SP) given.
A7 = 3	filecode not assigned or not assigned to a spooled device.
A7 = 4	I/O error, writing the EOF record on the spool file.
A7 = 5	I/O error, accessing the Spool DAD.

COMASG **Copy the MAS segments file** COMASG

Purpose

MAS is an operating system, which is partly core resident and partly disc resident. While the core resident part can be created by using the Linkage Editor, some special actions must be performed to create or copy the segment file.
Two kinds of segment files exist:
- D:MASG is a DFM file of type UF residing in the directory of the System user. Although the file may have consecutive or non consecutive attributes, there may be no gap between the sectors.
- D:MSEG is a DAD with a sectorlength of 4082 bytes and a length that varies with the release that is used. It is created by the System Generation procedure %%DCICDC.

The D:MASG file is used for non CDC-SMD systems. When MAS needs a segment to read in it calculates the sector number in the D:MASG file where the segment starts and reads the segment into memory at address /300. For one segment, up to 10 sectors need to be read in.
The D:MSEG DAD is used for a system with CDC-SMD discs. These discs have variable sectorlengths and so it is possile to read a segment with one access to the disc.
To copy the D:MASG file from the starter pack to the user pack, the utility COMASG has been developed. This utility is the only way to fill the D:MSEG DAD and can also be used to copy a D:MASG file to another D:MASG file.

Calling sequence

The COMASG utility has to be called as follows:

```
    ASG  FCOD=/40,FNAM=D:MASG[,USID=usid[,DAD=/Fx]]
    RUN PROG=COMASG
```

Filecode /40 has to be assigned to the D:MASG file that has to be copied.
The DAD filecode /F2 <u>must</u> be assigned to the first DAD on the pack whereon the D:MASG file or the D:MSEG DAD to be filled resides.

Warning

COMASG does not give any error messages. The only way to know whether it performed successfully is the time it needs to run (2-3 minutes).

Purpose

To make a fast copy (5-10 minutes) from one CDC-SMD disc to another CDC-SMD disc or to certify such a disc. The COPY program is a stand alone program that can be loaded by answering the MONITOR? question of IPL-DK program with COPY.

Use of the COPY program

When loaded, the COPY program starts a dialogue:
- STANDARD INITIALZATION? (Y OR N)=
Replying '?' or any other character different from "Y" or "N" the standard values are output, being:
    - CP (control panel) INT LEVEL = /07
    - CU (control unit) ADDRESS = /16
    - CU INT LEVEL = /20
    - EMITTER DRIVE 0
    - NOT A BIGD2 CU (so a BIGD)
    - NO PRINT OF ALL STATUSES
    - NO PRINT OUT OF THE MENU (the list of possible actions)

Replying "Y" will avoid to reply on the 7 initialization questions by using the prefixed ones mentioned above. The question 'WAITING FOR ACTION' (see below) is printed immediality.
Replying "N" permits to make ones own initialization by answering the following questions:
- CONTROL PANEL INTERRUPT LEVEL (1 HEXA CHAR)=
  Reply the level of the control panel interrupt (normally 7) in one character. For all replies requiring a numeric input, the user should input hexadecimal characters without preceeding "/".
- CU ADDRESS (2 HEXA CHARS.)=
Reply the address of the control unit (e.g. 16 or 17). The control unit address in the lowest device address of the discs connected to it.
- INT LEVEL (2 HEXA CHARS.)=
Reply the interrupt level of the control unit (normally 10 or 20).
- EMITTER DRIVE NUMBER (0 OR 1)=
A drive number must be entered here. The drive number identifies the pack to be handled, for the COPY function, it is the pack that must be copied to another pack. The disc with the lowest device address is always connected to drive number 0 and the disc with the highest device address is always connected to drive number 1.
- IS IT A BIGD2 CU? (Y OR N)=
Reply "Y" (BIGD2 control unit) or "N" (BIGD control unit). The difference between the two control units is that a BIGD can address directly up to 128KW and a BIGD2 control unit up to 512KW.
- DO YOU WANT A PRINT OF ALL STATUS ERROR (FOR SERVICE PEOPLE ONLY) (Y OR N)=
Reply "Y" or "N", depending on your profession.
- DO YOU WANT A LIST OF POSSIBLE ACTIONS? (Y OR N)=
If "N" is replied, the message 'WAITING FOR ACTION' is output, if "Y", the following list appears:

```
        POSSIBLE ACTIONS
        0= COPY PACK TO PACK
        1= BAD TRACKS LIST
        2= DATA FAULT DETECTION
        3= RESTART INITIALIZATION
        4= NOT YET AVAILABLE
        5= FLAG A BAD TRACK
        6= PACK CERTIFICATION
        7= PRINT VTOC
        TO SELECT ACTION GIVE A CP. INT.
```

This list is followed by:
- WAITING FOR ACTION
which ends the initialization phase. On an erroneous reply of a question in the initialization, the question is re-output. On a valid (for the program) but incorrect (for the user) answer, the initialization can be restarted by action 3 or by a re-IPL.

Waiting for action, the program goes into an idle loop, waiting for a control panel interrupt. When this interrupt is given, the program outputs:
- ACTION=
and the user replies a number from 0 to 7, except 4 because that is not yet available, each starting an action. These actions are described below. Any action can be stopped by a CP interrupt, which causes the ACTION= to re-appear.

ACTION 0: COPY PACK TO PACK

This action permits to copy one disc to another one. Both discs must be CDC-SMD discs and connected to the same Control Unit. The copy takes 5-10 minutes (the Librarian SDD command 1-2 hours). The copy is only possible, when the emitter pack has a MAS structure. Packs to and from can be copied are:
- from 40MB to 40MB
- from 40MB to 80MB
- from 80MB to 80MB
- from 80MB to 40MB if the used space on the 80MB is not greater than 40MB.

To initiate the copy action, the following questions are output:
- RECEIVER DRIVE NUMBER (1 CHAR)=
Reply 0 or 1. It is the number of the unit to which the copy is to be made. If the Emitter drive is 0, the Receiver drive is 1 and vice versa. It is advised to copy from drive 0 to drive 1.
- RECEIVER DRIVE TYPE (40M=1 80M=2) =
Reply 1 or 2.
- IS YOUR PACK ALREADY CERTIFIED? (Y OR N)=
Reply "Y" implies that a certification (action 6) has been made in the past. In this case the copy process starts immediatily. When a bad track is detected on the Receiver pack, the program outputs its coordinates in the message:
- BAD TRACK DETECTED ON CYLINDER = xxx HEAD yy
This message is only an information to check, whether the bad tracks found now are the same as the ones found in a previous certification.
When the Receiver pack was not certified, so the previous question was answered with "N", the following message is output:
- DO YOU TAKE THE RISK? (Y OR N)=
Reply "Y" means that the user takes the risk to work with an uncertified pack. The copy process makes a quick premark before starting the actual copying. The reply "N" stops immediatily the action and the program goes into the 'WAITING FOR ACTION' state.

After these questions, the program prints the volume label of the Emitter pack, asks whether it is the right volume to copy and continues with asking for a volume name, a pack number and the date to construct the volume label of the Receiver pack. Then the copying starts with for each DAD the message:
- COPY OF DAD: xxxxxx
At the end of the copy process, the volume label of the Receiver pack is printed.

ACTION 1: PRINT OF BAD TRACK LIST

For this action, one additional question is output:
- EMITTER DRIVE TYPE (40M=1 80M=2)=
Reply 1 or 2, identifying the capacity of the disc.

Then the action starts. For each Bad Track, the message:
- BAD TRACK DETECTED ON CYLINDER = xxx HEAD NUMBER = yy
is printed. When all Bad Tracks have been printed, or when no Bad Tracks are present, the program turns into the idle state by printing:
- WAITING FOR ACTION

ACTION 2: DATA FAULT DETECTION

This action checks a disc on occurences of Data Faults. The tracks on which a Data fault is found are checked against the Bad Track list whether they are recorded. If not, an error message is printed and the disc must be re-certified. One additional question is output:
- DRIVE NUMBER (0 OR 1)=

Reply 0 or 1. This process performs the same actions as the copy process (action 0), without writing on an output pack. Per DAD the message:
- DATA FAULT DETECTION ON DAD xxxxxx
is printed.

ACTION 3: RESTART INITIALIZATION

This action allows to start the complete initialization of the COPY program, without re-loading.

ACTION 4

This action is not yet available. When called, an error message is output to inform the user about that.

ACTION 5: FLAG BAD TRACK BY OPERATOR

This action flags a track as bad and assigns an alternate track.
Two additional questions are asked:
- CYLINDER POSITION? (3 HEXA CHAR.)=
- HEAD NUMBER (2 HEXA CHAR.)=

With the replies to these two questions, the bad track is fully identified and flagged by the program. The track is read again to check the successfull completion of the operation. If no error is found, the messsage:
- BAD TRACK CORRECTLY FLAGGED
is output, if not the message:
- IMPOSSIBLE TO FLAG THE TRACK, SORRY, BUT THE PACK IS NO LONGER USABLE
is output, followed by the 'WAITING FOR ACTION' message.

ACTION 6: CERTIFY A PACK

For certification the dialogue is:
- IS YOUR PACK ALREADY CERTIFIED? (Y OR N)=

In case of a new pack, the reply must be "N". In case of an already certified pack, the reply must be "Y". When "Y", tracks flagged by a previous certification are kept and recalled as their recovery proceeds. The flagged bad tracks are printed as if they were found during this certification.
- FULL SURFACE MUST BE CHECKED? (Y OR NO)=

If "Y", all tracks of the disc are checked one by one (411 cylinders for 40M and 823 cylinders for 80M discs). If the reply is "N", the certification is restricted to the part of the disc specified in the answers on two questions, asking for the starting and ending cylinder to be certified (smallest part is one cylinder).

- LOOPING MODE? (Y OR N)=

This function permits to check the disc only once (reply "N") or more than once (reply "Y"). The experience proves, that two or three loops may be necessary. That is why it is advised to answer "Y" and to certify packs during night if possible. The certification process can be stopped by pressing the control panel interrupt button. A run indicator is printed at the end of each pass.

After the 'LOOPING MODE' question, the certification starts.

A loop consists of writing, re-reading and comparing each track:

- with pattern /0000 : 4 sectors of 2KW per track
- with pattern /FFFF : 32 sectors of 256 words per track
- with pattern /BFBF : 39 sectors of 205 words per track
- with random pattern : 64 secotrs of 105 words per track.

After certification, the disc is not yet usable. It should first be premarked (by a normal operator PK command). The premark asks whether the pack has been certified or not.

ACTION 7: PRINT VTOC

This action prints the VTOC (volume table of contents) of the disc.

Example:

DAD NAME	NB OF INT	NB OF SECT/GRAN	NB OF SECT/TRACK	SEC.LENGTH
SUPERV	0010	0008	0027	019A
D:CI	0003	0001	0004	1000

All values are output in hexadecimal format.

Purpose

Accessing a device, the hardware can return an error status, like parity error, data fault, etc. MAS retries such errors up to 5 times and, if it is a persistent error, /8000 is added to the status and it is stored in the calling ECB.
If the error is not persistent, no status is put in the calling ECB and the program continues as if nothing has happened. However, it might be very useful to know which device or (for disc) which track gives an error now and then. It might be the cause of a disaster in the future. Therefore, error logging has been developed. It logs the error, which was retried successully, on a file. The information in this file can be printed, and according to the obtained data actions can be taken to prevent real errors in the future.

Error logging in the system

Error logging is only active in the system, when the file D:ERLG, type UF is assigned in the system machine to filecode 33, thus:

```
    ASG 33,DDFx,UF,D:ERLG
```

When during processing a hardware status is received, MAS records the error-information in a block in the System Dynamic Area, if filecode 33 is assigned. Every minute, the clock interrupt routine checks whether there are error logging blocks in the System Dynamic Area and, if so, it activates a MAS disc resident segment to write the block(s) onto the D:ERLG file. As this process is driven by the clock, the RTC must be switched on.
The D:ERLG is cyclic. When it is full, it starts from the beginning. A warning message is output, when the file is nearly full.

ERLOG utility

On the starter pack, a utility called ERLOG is delivered. With this utility one can create the D:ERLG file (which has a special format) and retrieve information from it.
The calling sequence of the utility is:

```
ERLOG
OPT FUNC=func[,DAD=dadfc][,USID=usid][,NBGR=n]
```

where:

func	is the function that should be performed: FUNC=CRE: create the D:ERLG file, FUNC=LIST print the information recorded in the D:ERLG file.
dadfc	is the filecode of the DAD (/F0-/FF), where the D:ERLG file must be created (FUNC=CRE) or where it resides (FUNC=LIST. Default filecodeis /F0.
usid	is the userid where the D:ERLG file must be created or where it resides. Default userid is MASUP.
n	is the number of granules to be assigned to the D:ERLG file, when FUNC=CRE. Default is 1 granule.

Error messages:

```
    INVALID OPTION STATEMENT (does not start with 'OPT ')
    INVALID PARAMETER xxxx
    NO FUNC PARAMETER
    VALUE: xxxx OF PARAM: yyyy IS ILLEGAL
    PARAM: xxxx IS REDUNDANT
    ERROR DURING LKM: xx, STATUS= /yyyy
```

(the utility uses several LKM's and LKM xx returned a status. For the meaning of the status, see Appendix C)

Output of ERLOG

For FUNC=CRE, the output of the utility (when it acted successfully) is:
```
ERLOG FILE CREATED
```
For FUNC=LIST the following items are printed:
- total number of recorded errors
- total number of recorded errors per device address
- information per error

The information per error consists of:
- device name and address
- machine name and name of the program that issued the error-causing access
- date and time when the error occurred
- hardware status
- for disc: cylinder, head and sector number
- number of retries
- control word given to the CIO start command.

Purpose

With FILEXC one can write/read load modules to/from a sequential file or device. FILEXC also handles files with other types (except EF), but these can also be handled by the Librarian.

Calling sequence

```
FILEXC
OPT FUNC=func,FCOD=fc,SYST=MAS,FNAM=fnam,FTYP=ft[,FDES=des][,KEEP=YES]
```

where:

func	is the function that should be performed: FUNC=IN reads the file from the sequential device to disc, FUNC=OUT writes the file onto the sequential device.
fc	is a filecode assigned to a sequential device.
fnam	is the name of a file.
ft	is the file type (UF, SC, OB or LM).
des	is a description to be put into the identification block. The maximum length is 16 characters, only consisting of alphanumeric characters and blanks.

- FILEXC assumes the file to be accessed in the current JOB Usid and DAD.
- KEEP=YES is only applicable when FUNC=IN. If specified, the file just read is kept in the directory of the current JOB Usid and DAD- SYST=MAS indicates
that the utility is executed under the MAS operating
system. The alternative is SYST=DOS.
- FILEXC only works in the Batch machine.

When FUNC=OUT, FILEXC first writes an identification block onto the sequential device (MT or TK). In this block, all parameters specified in the option statement are recorded. These parameters are checked against the option statement specified when the file is to be read in. After the identification block, the file is written.
FILEXC starts with printing a heading containing information about the file to be handled.

Error messages

```
OPTION STATEMENT MISSING
INVALID OPTION STATEMENT
INVALID KEY-WORD
TWICE THE SAME KEY-WORD
= NOT FOLLOWING THE KEY-WORD
KEY-WORD VALUE MISSING
SYST KEY-WORD MISSING
INVALID SYST KEY-WORD VALUE
FCOD KEY-WORD MISSING
INVALID FCOD KEY-WORD VALUE
FCOD FILE CODE NOT ASSIGNED
FUNC KEY-WORD MISSING
INVALID FUNC KEY-WORD VALUE
FDES NOT ALLOWED WHEN FUNC=IN
KEEP NOT ALLOWED WHEN FUNC=OUT
KEEP NOT ALLOWED WHEN SYST=DOS
INVALID KEEP KEY-WORD VALUE
FNAM KEY-WORD MISSING
```

```
INVALID FNAM KEY-WORD VALUE
FTYP KEY-WORD MISSING
INVALID FTYP KEY-WORD VALUE
KEEP NOT ALLOWED WHEN FTYP=OB
INVALID ASSIGNMENT TO A PERMANENT FILE
IMPOSSIBLE ASSIGNMENT TO A TEMPORARY FILE
MAS CONTROL IS NOT BATCH-MACHINE
I/O ERROR WHEN DELETING FILECODE
KEEP FILE OPERATION NOT SUCCESSFULL
EMPTY INPUT FILE
INVALID INPUT RECORD
INCONSISTENT INPUT FILE NAME
INCONSISTENT INPUT FILE TYPE
INCONSISTENT INPUT SECTOR LENGTH
INCONSISTENT INPUT GRANULE SIZE
TOO MANY INPUT SECTORS
INVALID INPUT SECTOR NUMBER SEQUENCE
TOO MANY/FEW INPUT RECORDS PER SECTOR
```

Purpose

IPL-DK is a utility that must reside in the 3rd granule of the disc from which is IPL-ed. It is a stand alone program which is loaded at IPL and gives the possibility to have more than one monitor on the same disc.

Behaviour

After IPL, the utility is loaded and started. It outputs the following questions:

- DAD:

Whether or not this question is output depends on the version of IPL-DK. The reply must be the name of a DAD residing on the IPL-ed disc, (default is the first DAD) or a question mark (?), which results in a print out of all the DAD names residing on the disc and re-outputting of the DAD question.

- MONITOR:

The reply on this question must be the name of a load module residing in the first Userid of the DAD indicated in the DAD question (e.g. SUP, LDMASR, COPY) When MAS8 (so LDMASR) is intended to be loaded, the DAD question must be replied with the first DAD of the disc, the same applies for a DOS monitor. A question mark (?) results in a print out of all load modules residing in the first Userid of the DAD and re-outputting of the MONITOR question. A <CR> letsthe DAD question re-appear.

- LOAD ADDRESS:

Reply the address where the monitor is to be loaded. Default (and normal input) is a carriage return <CR>, being address /0000.

Purpose

LDMASR is a utility to load a MAS8 (extended mode) monitor, because such a monitor cannot be loaded by the normal IPL.

Behaviour

After loading LDMASR asks for the name of the MAS8 monitor to be loaded:

```
    - MONITOR:
```

The reply must be the name of a load module containing a MAS8 monitor, which resides in the first Userid of the first DAD of the disc from which is IPL-ed. The default (<CR>) is MASR.

Purpose

With OVLGEN, one can move a load module, from filecode /D6 (/L) to the D:MASG file. The D:MASG file must be assigned to filecode /40.
A full description of this utility is given in Appendix C with LKM 47.
The OVLGEN utility only runs in the Batch machine.

0.25M Flexible (Floppy) Disc Driver

0.25M Flexible discs connected to a 0.25M flexible disc Control Unit are accessed as a physical device with direct access at sector level. Every sector is available for data storage.

The logical and physical characteristies of the flexible discs are as follows:

- Physical sector length is 128 characters.
- Logical record length using 1 LKM instruction (4 sectors):
  - a) connected to a programmed channel: up to 512 characters
  - b) connected to the I/O Processor: the user may read or write to a maximum of 1 track (3328 chars).

  If a record comprises more than one physical sector, it is defined by the physical address of the first sector in the record.
- A track is 26 sectors, or 3328 characters (not interlaced).
- A floppy disc has 77 tracks (2002 sectors, 256256 characters).
- A sector is defined by a sector number from 0-2001 (the sector number given in word 5 of the ECB used for I/O operations with LKM 1).
- Bad Track conditions are handled by the driver.
- The Floppy Disc Driver package is included during system generation, in the same way as other physical devices, using the mnemonic `FL'.

Floppy Disc Operations

The order loaded into A7 before issuing an LKM 1 determines which of the various possible functions is carried out by the control unit. These can be summarised thus:

Order	Operation
/00	Get extended information about a filecode
/11	Read Sector
/15	Write Sector
/3A	Compound Read
/3B	Compound Write
/3F	Write Sector and Verify
/2F	Write Deleted Data Address Mark
/3D	Write Deleted Data Address Mark and Verify
/3E	Search Key
/3C	Search Key Masked
/2D	Door Lock
/2E	Door Unlock
/30	Return Information about a Filecode.

The ECB Layout

The ECB address is loaded into A8 before issuing the LKM 1 requests. The layout of the ECB is as follows:

	0 7	8 15
Word 0	Event Byte	Filecode
1	Buffer Address	
2	Requested Length	
3	Effective Length	
4	Returned Status	
5	Sector Number	

where:
- `Filecode' is the filecode assigned to the floppy disc unit.
- The `event byte'; bit 0 is set to 1 by MAS, to indicate completion of the operation.
- Buffer Address is the user's buffer address, which can contain or receive data as normal, but can also contain control information (e.g. Search Key or Write Deleted Data Address Mark).
- Requested length is expressed in characters, and should be even (except for search key operations); MAS will round down odd-numbered values.
- Return Status codes are summarised later.
- Effective length is the actual message length, in characters, read or written.
- Sector Number is a relative sector number or sector address, in the range 0-2001.

Read/Write Operations

/11 - Read Sector

From 1 - 4 sectors (each of 128 words) may be read in one operation; thus the maximum record length is 512 characters. The sector number of the first sector in the record determines the start address or sector number specified in word 5 of the ECB.

/15 - Write Sector

From 1 - 4 sectors can be written directly. The requested length should be even and not greater than 512 characters.

/3F - Write Sector and Verify

This order is performed as for /15, but with the additional feature that the disc revolution following the write operation is used to read and verify the record.

/3A - Compound Read

This order allows the user to read up to 3328 characters (i.e. 1 track or 26 sectors) in one disc revolution. The starting sector address quoted in ECB word 5 is the logical start of the track.

The operation is performed correctly with the disc control unit attached to the Input/Output Processor (IOP). If the disc control unit is attached to the Programmed Channel the command is accepted, but more than one disc revolution is required to complete it. The ECB values are as for function /11 (Read), except that the allowed requested length is in the range 2 - 3328.

/3B - Compound Write

This order allows a complete track (26 sectors) to be written in one revolution of the disc, as long as the disc control unit is connected to the IOP. If it is connected to the Programmed Channel, it will require more than one revolution. The requested length is in the range 1 to 3328 characters.

/2F - Write Deleted Data Address Mark (DDAM)

This order allows the user to write a special pattern on the disc. This pattern should be placed in the user's buffer before the LKM 1 is issued. The sector number to which the mark is to be written is entered in word 5 of the ECB, while word 2 of the ECB contains the requested length expressed in characters.

If more than one sector is to contain the DDAM, the requested length is in the range 128 - 512.

/3D - Write Deleted Data Address Mark and Verify

This is carried out as for order /2F, but a further revolution of the disc is used to read back the pattern and verify that it has been written correctly.

Search Key Orders

/3E - Search Key

This order allows the user to search for a pattern of characters which begin at the start of a sector. The search is carried out within and including a start sector number and end sector number, specified by the user.

Either an even or an odd number of characters may be specified as the search pattern. This pattern is placed in the user's buffer, and is immediately followed by the start and end sector number.

The length of the pattern is specified in word 2 of the ECB.

After a successful search, the sector number of the sector containing the pattern is entered by MAS in word 5 of the ECB.

The following diagrams illustrate how the user sets out his buffer before issuing the LKM 1:

a) For an even number of characters:

```
1            2             length of search pattern
3            4             character string = 6
5            6
Start sector number        search limits
End sector number
```

b) For an odd number of characters:

```
1            2             length of search pattern
3            4             character string = 5
5            ignored
Start sector number        search limits
End sector number
```

/3C - Masked Key Search

This is conducted in the same way as the Search Pattern order /3E, except that the user is able to blank or mask certain characters within the pattern string which will be ignored during the search operation. The character positions to be masked are filled with the space character (/20).

The following example shows how the user should lay out his buffer before issuing the LKM 1 request.

Suppose the pattern to search for starts at character 3 of the sector, is 7 characters long and consists of the characters A and B at character positions 3 and 4 and the character C at position 6:

/20	/20	
/20	A	search pattern
B	/20	
C	ignored	
Start sector number		search limits
End sector number		

<u>Other Orders</u>

/2E - Door Lock

This command locks the loading door of the floppy disc unit, preventing manual opening of the door until the Door Unlock command (/2D) is received. The ECB should be initialised as follows:

0	Filecode
1	Buffer Address (dummy value)
2	Requested length (dummy value)
3	Returned Length (unchanged)
4	Returned Status (updated by the system)
5	Sector number (dummy value).

/2D - Door Unlock

This order unlocks the door of the floppy disc drive unit, allowing the operator to open the manual door latch in order to extract the disc. The ECB layout is exactly the same as that of the /2E (Door Lock) command.

/30 - Return Information About a Filecode

This command causes the unit to return a value for Best Length to word 3 of the ECB. The other locations are as described in Appendix C under LKM 1.

<u>Error Messages</u>

If an I/O operation is attempted on a drive which is inoperable (e.g. the door is open), the following message is printed on the console:

```
    PU FLxx,yyyy,RY
```

where:
- xx is the Floppy Disk Address
- yyyy is the status
- RY means Retry or Release Device.

For the retry the operator must remove the cause of the error, when the I/O operation should proceed automatically.

Returned Status for Flexible Disc

If bit 0 is 0 but bit 1 is 1 a hardware error has occurred, and the following bits have significance:

Bit	Meaning
2	Key not found
4	Deleted data mark read
5	Record not found
6	Write protected
10	The request was completed after one or more retries.
11	Program error
12	Incorrect length
13	Data error.

If bit 0 = 1 and bit 1 = 1, an error occurred in the calling sequence. If bit 11 is also set, the request code is invalid or the sector number is incorrect.

1M Flexible Disc Driver

1M (1 megabyte) flexible discs are supported by the FL1MZ and the FL1MB control units. These control units are always connected to the IOP and they can handle 1M flexible discs as well as 0.25M flexible discs.

Disc types

This flexible disc driver supports the following flexible disc types:

- F1 : 0.25M flexible system disc
- F2 : 0.25M flexible data disc (MAS relase 8: type F6)
- F3 : 1M flexible system disc
- F4 : 1M flexible data disc
- F5 : 1M flexible data disc

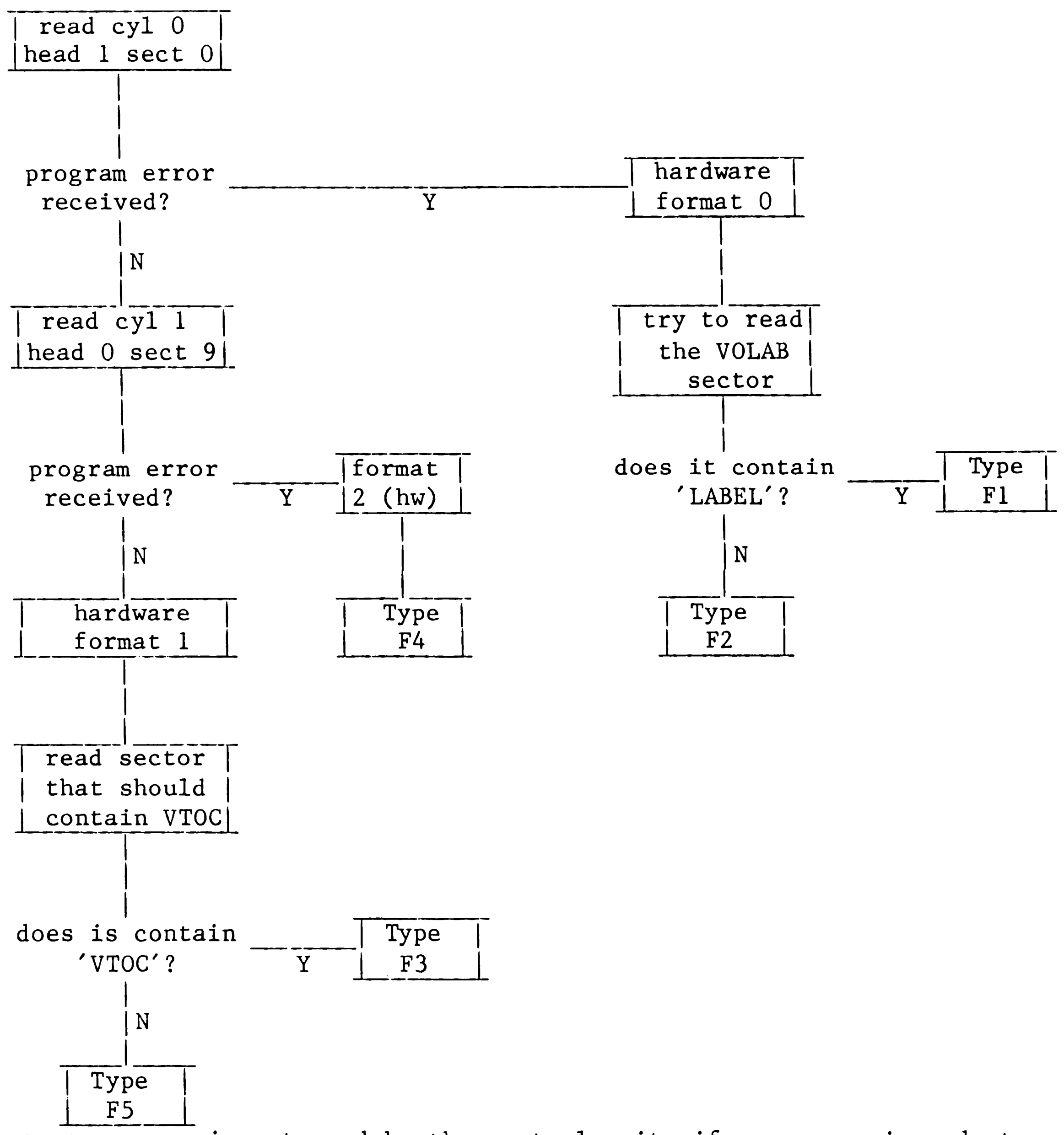
Hardware formats

There are three hardware formats for flexible discs.
The characteristics for each format are:

Hardware characteristics	Format 0	Format 1	Format 2
number of surfaces	1	2	2
number of cylinders	77	77	77
sect/track cyl 0, head 0	26	26	26
sect/track cyl 0, head 1	-	26	26
sect/track other cyls	26	26	8
sectorlength (char) cyl 0, head 0	128	128	128
secl cyl 0, head 1	-	256	256
secl other cyls	128	256	1024
number of useful cyls	74	74	74

The translation of the hardware format into the flexible disc types F1, F2, F3, F4 or F5 is done at IPL time, if there is a flexible disc in the drive, or when a ready interrupt is received.

The assignment of the types is done according to the following scheme:



A program error is returned by the control unit, if an access is made to a non-existing track or to a non-existing sector (i.e a too high sector number). So the first check on program error determines whether there are one or two tracks (surfaces) available on a cylinder and the second check whether there are 8 or more (so 26) sectors available on the track. These two checks identify fully the hardware format.

Orders

For this driver, the following orders are allowed:

- /01 or /11: read; allowed for all types of flexible discs.
- /05 or /15: write; allowed for all types.
- /13 : verify; allowed for all types.
- /21 : read VTOC; allowed for type F3, simulated for type F1.
- /25 : write VTOC; allowed for type F3, simulated for type F1.
- /2D : door lock; allowed for all types.
- /2E : door lock; allowed for all types.

- /2F : write deleted data; allowed for types F2, F4, and F5.
- /3A : compound read; allowed for all types.
- /3B : compound write; allowed for all types.
- /00 or /30: get filecode/device information; allowed for all types.

- For read/write on DAD or disc level, with type F1 and F3 the requested length must be the length of one logical sector. For Data flexible disc, any length is accepted and there is an automatic chaining on next tracks.
- For read/write VTOC, the ECB must be filled as follows: ECBSC must contain 0 and the word ECBHD must contain the physical sector address of the VTOC (this address can be found on displacement /52 of the VOLAB).
- With verify, the command of the buffer in memory is compared with the contents of one or more sectors on the disc. It can be used after a write, to check the written data. The buffer must remain the same between the two commands. If the comparison is incorrect, the status Data Fault is returned.
- Door orders are only executed when the addressed drive is operable and when the 'door lock' option is present. In other cases, the command has no effect.
- Write deleted Data, writes the sectors with deleted data address marks. The ECB must be filled as in the write command.

Bad track handling

The FL1MZ and FL1MB controllers use cylinders 75 and 76 to assign bad tracks. These cylinders cannot be accessed by an application. This is a constraint in the transportability from MAS to other systems (like IBM). Others systems may use cylinders 75 and 76 to put data on, which cannot be retrieved by MAS. This data can even be overwritten, when a bad track is detected by the controller.
The driver itself does not contain bad track handling, so if an error occurs, the flexible disc should be exchanged.

Flexible disc type F1

The type F1 is a 0.25M flexible disc, with an X1215 disc simulated lay-out. No DAD's can be declared on it. The whole disc is seen as containing one DAD, called DADFF1. No VTOC resides on the disc, it is simulated by the flexible disc driver.
The disc type, stored in the VOLAB of the type F1 flexible disc is 'FLD1'.

One logical sector, being 410 bytes, is stored physically in 4 sectors on the flexible disc. An interlace table is maintained in memory (not on disc, because it has no VTOC). The interlace factor is 2.
Interlacing is not performed on logical setors 0, 1 and 2. For logical sector 0, the sectors starting at physical sector 0 are accessed. For logical sector 1 sectors from physical sector 4 are accessed and for logical sector 2, accessing starts at physical sector 12.

System commands like ASG and DUF, on a type F1 flexible disc are possible. However some commands concerning DAD's like DCD and DLD (declare and delete DAD) are not possible, because of the DADFF1 VTOC simulation.
The type F1 is considered by MAS as a disc and so a filecode should be assigned to it during System generation. Thus:

```
        FCD: /Cx,MFyy
```

/Cx is the filecode assigned to the type F1 disc in the System machine.
MFyy is the device mnemonic for the disc (MF) and its device address.
Note that MF is only the device mnemonic during System generation. In the monitor the device is considered to be a disc, with the device mnemonic 'DK'.

A declaration of the disc in a machine declaration is done as follows:

```
    FCD /Cx
    FCD /Fy,/Cx,DADFF1
```

/Cx being the filecode assigned during System generation and /Fy the filecode to be assigned to the (only) DAD on the disc.
Receiving a ready interrupt of a type F1 disc the message:

```
    FLOPPY DATA F1
```

is output by the system.

A type F1 disc must be premarked. The premark asks for the type of flexible disc: FLOPPY TYPE F1 OR F3
For this type, the reply must be F1. Premark does not ask question about the DAD to be created.

<u>Flexible disc type F2</u>

A type F2 flexible disc is a Data disc with hardware format 0. It can only be accessed on physical level. So no ASG a logical file, DUF and other system commands are allowed for this type of disc. Although the disc is recorded as 'DK' in the system, one may assign any filecode to the disc, e.g. ASG /10,DKxx.

Receiving a ready interupt, of a type F2 disc the message:

```
    FLOPPY DATA F2
```

is output by the system.
<u>Flexible disc type F3</u>

The type F3 flexible disc contains a DAD structure. The first DAD starts at cylinder 1 and the lay-out of the disc is as follows:

```
    - Cylinder 0: Physical sectors 0, 1, 2 and 3 are IPL sectors.
    - Cylinder 1: Logical sector 0 contains the volume label.
                : Logical sectors 2-5 contain the catalog of the first DAD.
                : Logical sector 6 is the not used Bad track sector.
                : Logical sector 7 contain the VTOC.
```

The type F3 is a hardware format 1 disc with a X1215 simulated structure. DAD's can be declared, files assigned etc. DAD's have interlace factors, which must not have a common divisor with the number of sectors per track (i.e. 26). Interlace factor 1 is allowed.

The disc type stored in the VOLAB for the type F3 disc is 'FLD2'.

Logical sectors contain 410 bytes. The first cylinder of the disc cannot be accessed. For each access to the disc, the calculated cylinder number is incremented with 1, so logical sector 0 is translated into an access to the first sector on cylinder 1.

Generating the flexible disc, in System generation a /Cx file code must be assigned to the device (see type F1)

Receiving a ready interrupt, the system outputs:

```
    FLOPPY DATA F3
```

when this type of disc was mounted.

A type F3 disc must be premarked and the question:

```
    FLOPPY TYPE F1 OR F3
```

must be answered with 'F3' for this type.

Flexible disc type F4

The type F4 flexible disc is a Data disc with hardware format 2. For this type, the same rules apply as for the type F2 flexible disc. Note that the sectorlengths for cylinder 0 head 0, cylinder 0 head 1 and the other cylinders differ.

Receiving a ready interrupt, the system outputs:

```
    FLOPPY DATA F4
```

when this type of disc was mounted.

Flexible disc type F5

This is a flexible disc with hardware format 1. The same rules apply as for the other data disc types. And of course the message:

```
    FLOPPY DATA F5
```

is printed when this type of disc gave a ready interrupt.

GENERAL

This Appendix describes how the user can incorporate his own user driver into the MAS Monitor. No alterations to existing Monitor routines are required, andthe driver package is then included during the system generation procedure.

PREPARATION OF THE DRIVER PACKAGE

After being coded, the driver is compiled and the object module is stored within the userid MASGEN in the object library MASOB for non Extended Mode systems and in the object library IOCSOB for Extended Mode system. This can be achieved by using the Librarian processor.
A driver consists of two parts:

- the device dependent LKM 1 (I/O) handling, filling the device dependent part of the DWT (device work table) and executing the WER instructions - for devices connected to IOP - and the CIO start instruction.
- the interrupt routine, which handles the interrupts, executes the SST, INR, OTR and CIO stop commands and branches upon end of I/O to the general end of I/O handler of the system.

Normally, both parts are coded in the same module. This module contains two entry points, one for the device dependent I/O handler (called D:<name>) and one for the interrupt routine (called S:<name>).

The driver must be made known to the system by including its DWT in the DWT chain and by filling the address of its interrupt routine in the appropriate interrupt location in the LOCAT module.

The LOCAT module

The incorporation of the driver in LOCAT (so filling the interrupt location) can be achieved by answering the "USER INT:" question in the System generation CONGEN process with <lev>,<name>. The System generation will then at interrupt location <lev> store the address of the external I:<name>. An EXTRN for I:<name> is also generated.

The DWT chain

For each device, a DWT must be incorporated in the DWT chain. For disc devices, also a DCT (dics control table) must be included in the DCT chain, but as the writing of disc drivers needs special knowledge of the MAS system, it is not discussed here.
A DWT can be put in the chain by replacing the module USDWT, which is present in the MASOB resp. IOCSOB library by an own written USDWT module.
The standard USDWT module consists of one instruction:

```
    USDWT     EQU 0
```

As the last chain address in the DWT chain points to USDWT it contains zero (i.e. end of chain) when the standard module is used. Including a user written DWT in USDWT, the last chain address value changes from zero to the address of the DWT to be added, and that is exactly the intention.

[illegible]

EXAMPLE

The lay-out of USDWT, containing one user written DWT is:

```
          IDENT     USDWT
          ENTRY     USDWT
          ENTRY     I:<name>          address stored in the interrupt location in
                                      LOCAT.
          EXTRN     D:<name>          device dependent I/O handler.
          EXTRN     S:<name>          interrupt handler.
I:<name>  MSR       8,A15             save 8 registers in the system stack.
          LDKL      A6,USDWT+2        Fill A6 with the DWT address. Note that the
                                      DWT starts at the <dev name> word!
          ABL       S:<name>          branch to the interrupt handler
USDWT     DATA      NXTDWT            address of the next DWT, if no DWT, 0.
          DATA      '<dev name>'      mnemonic of the device that is included.
          DATA      <dwtda>           flags and device address.
          DATA      <best length>
          DATA      D:<name>          driver address
          ---                         ) For a description of the DWT, the user
          ---                         ) should refer to Volume IV, the
          ---                         ) Trouble Shooting Guide.
          DATA      ---               last word of the DWT
NXTDWT    EQU       0                 or the start of the next DWT.
          END
```

<u>Contents of the DWT</u>

The DWT is described in the Trouble Shooting Guide. Here is indicated:
- the fields that must be filled during DWT definition
- the fields that are filled upon entering the driver
- the fields that must be filled by the driver, returning to the end of I/O handler

<u>Defining the driver</u>

The following fields must be filled, defining the DWT:

-DWTDN	device name
-DWTDA	device address and flags
-DWTBLG	best length
-DWTDRV	driver address
-DWTRY	last 6 bits: the device type
-DWTC:N	address of a word containing /8000 (not busy)
-DWTSST	address of I:<name>+2
-DWTFLG	bit 4: transfer per word (1) or transfer per character (0) bit 5: single device controller (1) or multiple device controller (0) bit 6: device connected to IOP (1) or programmed channel (0)

By the general LKM 1 handler (X:IO) are filled:

-DWTBUF	buffer address in System Dynamic Area
-DWTRLG	requested length to be transferred
-DWTORD	the last 6 bits of the order
-DWTA5	PCT address of the calling program
-DWTA6	scheduled label address in the calling program (0 if no Schlab)
-DWTATT	used internally in X:IO
-DWTDET	used internally in X:IO
-DWTUEC	user's ECB address
-DWTURO	user's order
-DWTNT	used internally in X:IO
-DWTIME	used internally in X:IO

-DWTQUE used internally in X:IO
-DWTFCT address of the FCT entry in the System Dynamic Area

Fields, not used by the general MAS I/O routines can be used freely by the driver. The driver should, when input is received from a programmed channel device, store the current number of received characters in the field DWTEFL. The field DWTBUF must then always contain the address in the buffer where the next character is to be stored. The minimum length of the DWT must be /46 characters (including the chain word). There is no maximum.

<u>Inputs and outputs</u>

On entering the driver from the general I/O handler (X:IO), the DWT is filled as indicated above. Furthermore:

- A6 contains the DWT address i.e. points to DWTDN.
- A4 contains the last 6 bits of the user order
- the MMU of the calling program is loaded.

On end of I/O, when the driver should return to the general end of I/O handler, the following registers must be filled:

- A6 with the DWT address
- A8 with the address of DWTIOB
- A2 with the status received from the SST instruction
- A3 with the effective length.

After filling these registers, a branch must be made to the external R:TUR1:

```
      ABL     R:TUR1
```

<u>MAS routines</u>

Some MAS routines can (or even must) be called from the driver.

- M:DIS1: the dispatcher. This routine has to be called from the device dependent I/O part of the driver to return. M:DIS1 expects 8 registes in the A15 stack, which are already stored there upon entering the driver. The routine has to be called with an absolute branch:

```
      ABL     M:DIS1
```

- M:LRTN: absolute return routine. When the interrupt part of the driver decides not to return to the end of I/O handler (because the I/O has not ended)this routine must be invoked. M:LRTN expects 8 registers in the A15 stack, which are stored there by the I:<name> routine. M:LRTN is invoked by an absolute branch:

```
      ABL     M:LRTN
```

- R:GO62: go to hardware level 62 routine. When an interrupt is received from any device, it enters the interrupt handler on a hardware level equal to the interrupt level of the device. Only interrupts with a lower interrupt level are serviced, as long as the interrupt routine runs at the level entered. Therefore the routine should go as soon as possible to hardware level 62 to allow other interrupts to come. R:GO62 is called, using A15:

```
      CF      A15,R:GO62
```

Note that, entering an interrupt routine, the system is in inhibit mode. The driver must issue an ENB (enable) instruction as soon as possible to let the system re-get the normal execution.

- Execute: hardware instruction routines. To perform hardware instructions, the driver may use some general routines. The routines must be called with an absolute branch, the return address must be loaded in A4. The instruction is constructed in A3 and the driver can supply a control word in A2. A1 is destroyed. Calling sequence:

```
      LDKL    A4,ret
      ABL     routine
ret   ----
```

The routine is S:TIO for a start I/O instruction, S:SST for an SST, H:LTIO for a halt I/O, T:TST for a TST, I:NRIO for an INR and O:TRIO for an OTR instruction. On return the condition register is the condition returned by the hardware instruction.

- C:INPT: check input character. This routine executes the INR instruction for a programmed channel device (input), it checks the received character on special functions (backspace, delete line, end character) and performs a CIO halt instruction if necessary. On input A6 must contain the DWT address, the routine does not return to the driver. Calling sequence:

```
      ABL     C:INPT
```

A MAS manual is not complete without some words about the internal structure of the monitor.
In the monitor there are a lot of LKM routines present. They are invoked by the LKM interrupt handler (I:LKM) via the table T:LKM. This table is generated during Sysgen and may be examined by the one (or two) who wants a somewhat deeper knowledge about this subject. Other information can be retrieved from the description of the LKM 47 (user written LKM) in Appendix C.

Furthermore, the monitor is more or less a Foreground machine called SYSTEM. In a Foreground machine, programs are running and indeed, that is also the case for the System machine. All programs have a PCT (program control table) and are chained decently just like the PCT's in a normal Foreground machine. In the System machine also per Foreground machine a PCT is present to run the FCL task of that Foreground machine.

Programs running in the System machine

A list of the so called system programs may be obtained by giving a MAP command in the System machine. The system programs run on the lowest software level (= highest priority) in the system, except X:IDLE which runs on the highest available software level.
The system programs are:

----- X:IO LKM 1 (I/O) handler -----

X:IO performs all device independent functions of the LKM 1. For the start of an I/O it is entered by I:LKM, for the end of I/O it is entered by the driver.
Functions performed by X:IO are at start of I/O:
- check the validity of the ECB
- fill the DWT (see Appendix F)
- ask an intermediary buffer in the System Dynamic Area for non-disc devices and (if output) move the user buffer to that area.
- set the device to busy. If the device is already busy, create a device request queue.
- open a spool file or transform a request to a spooled device into a request to a file on the spool DAD.
- call the "device dependent" routine:
  - for a request to a physical device: the driver
  - for a request to a logical disc file: disc file management (M:DFM)
  - for a request to TDFM: the TDFM package
  - transform a request to a DAD into a request to a physical disc device.

Functions performed at end of I/O are:
- fill the user's ECB with the returned status and the effective length
- set the device free or take the next entry from the device queue and start it
- move the buffer to the user area (if input) and free the intermediary buffer, if that was asked during start of I/O
- clear the DWT
- set the event on the user's ECB
- start the scheduled label (if any).

----- X:MASG MAS segment handler -----

The disc resident segments of MAS are read in and executed under the X:MASG program. At system initialization, is determined whether a D:MSEG DAD (system filecode /FF) or a D:MASG file is to be read and according to this information, an ECB is constructed, used by X:MASG to read the segments. Note that, if D:MASG has to be read, the read is done at DAD level (/F0). The start sector

of D:MASG and the calculated start address of the segment to be read are added and from that sector, the segment is read. This means that the D:MASG file <u>must</u> consist of consecutive granules, although it needs not to have the "CONS"attribute.
Upon activation of X:MASG, A3 contains in the first 6 bits an entry number in the segment. The segmentnumber is stored in the last 10 bits of that register.

Functions performed by X:MASG are:
- check whether the segment to be called is already in core. If not, read it from disc. The current loaded segment number is stored in the field T:SCUR, which is an entry that can be found in the MAS Link list.
- check the result of the I/O operation to read the segment, if error is returned, abort the system with error /000B, except when the error was "disc not operable". If not operable output the message:
  SYSTEM DISC NOT OPERABLE, RESTART IT
  on system filecode /EF and stop the system until the system disc gives a ready interrupt.
- start the segment by an indirect branch to the entry (present in Link list) T:OVLA+(2*entrynr).

----- X:SWIO swap handler -----

Swapping in MAS is done via this task.
Functions performed are:
- determine whether the swap DAD (system filecode /F1) is a DAD with small (i.e 410 bytes) sectors or with large (4092 bytes) sectors. Construct an ECB according to this information.
- For swap out, write the program to the D:CI DAD, to a file, starting at the sectoraddress recorded in the program's PCT in word PCTSWN. At least, when the program has been declared swappable, for read only programs, writing is not performed. Then free the pages, occupied by the swapped out program.
- For swap in, reserve pages in the Dynamic Loading Area. The number of pages to be reserved is recorded in the program's PCT, word PCTREG. Then read the program into the reserved pages from the D:CI DAD. The address where the program resides is for read-only- or not previously swapped out programs PCTSW1 and for previously swapped out programs PCTSWN. Then start the actions recorded in the program's PCTMOV queue, which are delayed, because the program was loaded.

----- X:ALGR allocate/free granules -----

The (de)allocation of geanules has to be maintained by a separate task, to prevent that two tasks request granules at the same moment. This could lead to allocation of the same granule to more than one file. With a separate task, each system program that requests granules, activates X:ALGR and, if busy, an activation queue is created so that only one request is handled at the time.
On entrance of X:ALGR, A3 contains an entry number:
- 0: allocatate granules
- 2: free granules.

A4 contains the address of a control block. Lay-out:

```
   bit     0 1                        15
 word 0   |E|                        |NC|
      1   |            A4               |
      2   |      DAD address            |
      3   |      # of granules          |
      4   |         status              |
      5-  |    granule address          |
      n   |_____________________________|
```

G.0.2

where:

E	is an event bit, set upon completion of X:ALGR
NC	is an indication for consecutive (NC=0) or non-consecutive (NC=1) granules.
A4	is the address of the control block
DAD	is the address of the DAD control table in the System Dynamic Area. In the DAD control table, a BITTAB is recorded, wherein X:ALGR puts its alterations.
# gran	is the number of granules to be allocated or freed.
status	is zero if the requested number of granules could be allocated and 1 if a DAD overflow (not enough free granules) occurrud.
granad	is for consecutive granules a one word area containing the start address of the granules to be allocated/freed. For non-consecutive files, it is an n-word area (n is the number of granules to be allocated/freed). Each word containing one granule address.

----- X:TDFM segment handler for TDFM -----

X:TDFM performs about the same actions as X:MASG for TDFM segments under Extended Mode systems.
For TDFM, some segment buffers are reserved over 32Kw. X:TDFM checks whether the requested segment has already been loaded in one of these buffers and, if so, starts it. If not yet loaded, the least recently used segment is overwritten by the required segment and started.
X:TDFM reads it segments from the monitor load module (MASR) and <u>not</u> from the D:MSEG/D:MASG segment files.

----- X:USVC user service calls -----

X:USVC is used to start core resident "segments". It is activated with anentry number in A3. This entry number is searched in the table T:RMAC (which is generated during Sysgen) and it branches to the associated label. This task is used to activated core resident LKM's as specified in Sysgen.

----- X:LPxx spool out task -----

This task performs unspooling of files to the lineprinter (xx is the lineprinter's device address). It is activated by a segment that closes the spool file via a write EOF.

----- X:CRxx spool in task -----

This system task reads a file from the spooled cardreader (device address is xx) and writes this file to a file on the spool DAD (D:SPCR).

----- X:DUMP dump memory -----

This task is activated when a dump is required of (a part of) the memory. The dump is made via a system task, because dump commands are handled in disc resident segments (they are not often used). When the dump was made in a segment, under X:MASG no other segments could be activated and because dumping is a time consuming activity, it would block the system.

----- X:RTC real time clock handler -----

This task is only activated, when programs are connected to a timer or clock. It is done in a separate task, because this action is time-bounded and to free the clock interrupt routine.

----- X:OCOM operator command handler -----
X:OCOM is activated by the control panel interrupt routine (I:CTPN), when the INT button is pushed. It outputs an "M:" on system filecode /EF and reads then the operator command and activates the program that handles the given command. Furthermore it is used by system task to print a message to the operator, so that the calling task is not blocked during the time the message is being printed.

----- X:IDLE idle task -----
This task with the highest software level so the lowest priority is always eligible to run, but because of its low priority. It is only running when no other program wants to run, because it is not active, waiting or suspended (on resources, LKM, etc)
This task consists only of two instructions:

```
    X:IDLE    DLC  31        dummy instruction
              RB   X:IDLE
```

MAS abortions

MAS sometimes aborts, due to hardware, software, or even user errors. It is impossible to specify here, what the user should do when a system abort occurs. However some hints can be given.

The stand alone dump

When MAS aborts, a stand alone dump (if generated) is output on the lineprinter. The dump consists of a print out of the current registers, the current MMU register and the whole memory. In such a dump, some area'a can be indicated that may be of help for debugging.
-- P:CUR --
P:CUR is a one word area in the LOCAT module. It is an entry that can be found in the MAS link list. It contains the address of the PCT of the program that is currently being executed. At system abort it might contain the (system) program that caused the abortion. However, as the current program can be interrupted from a driver interrupt routine (which does not change P:CUR) it is not always true.
-- T:SCUR --
This is the current (or last) segment, executed by X:MASG. It is not in the scope of this manual to give a description of all segments (currently 107 arepresent), but a list of the segments with their functions performed can be obtained from the SSS department.
-- The A15 stack --
The A15 stack is (when generated normally) an area in the LOCAT module, ranging from /100 to /300. It is filled from the high address (/2FE) to the low address (/100). A stack overflow results in a system error /0007. The address of the first free word in the stack (i.e. the free word with the highest address can be found in A15).
-- The System Dynamic Area --
In the System Dynamic Area all buffers, control blocks etc used by the system are put. The start address of the SDA can be found in the system's link list, entry SYSDYN. All blocks in the SDA are chained via the first word. When the chain word is odd, the block following it is free, if even, it is occupied.

Error codes

Only abort codes, that occur relatively frequent are discussed here.

--- System error /0000 ---

This error code means: unknown interrupt. A hardware interrupt was received on an interrupt level that was not generated into the system. This level may be found by taking the word addressed by the contents of A15 plus two. This word contains in the first 6 bits the unknown interrupt level.

The same error can occur, when the system branched erroneously to an address in the interrupt locations. In that case, the interrupt level will normally be 61, 62 or 63. No standard debugging for this cause of the error can be given.

--- System error /0009 ---

This error code means: chain in the System Dynamic Area is corrupted. It is only detected, freeing a block from the SDA, when the block to be freed is not in the chain, or when it is already free.

The address of the block that was to be freed can be found as follows: take the current stack address (A15) and add 18. At this address, the address is put, that was to be freed, but that could not be found in the SDA chain.

--- System error /000B ---

The system aborts with error code /000B, when X:MASG has read a segment, but the event returned a status. The status (which is returned by an LKM 1) can be found via the register A8, which contains the address of the X:MASG ECB.

--- System error /0015 ---

This error means: invalid instruction in the monitor. The address of the word, containing the invalid instruction can be found via register A15. The contents of this register plus 24 gives the address of a word, containing the address of the invalid instruction.

--- System error /001B ---

Error /001B means that there was an invalid attempt to free granules in a DAD. The error is given when:

- the address of the granule to be freed is not a multiple of the number of sectors per granule
- the granule to be freed is already free
- the address of the granule to be freed is outside the DAD or zero.

The error is given by the X:ALGR task and the X:MASG task. When the X:ALGR task is involved, information on the error can be obtained from the control block given to X:ALGR (in A4).

No standard information can be given when the X:MASG task is the error causer. Only that A8 will point to an ECB, reading the GRANTB (i.e. the granule addresses to be freed) from the file to be freed.

For detailed descriptions about monitor control blocks, system error codes etc.the user should refer to Volume IV of this series, the Trouble Shooting Guide.

MANUAL COMMENT FORM

Concerns manual	P800M Programmer's Guide 3 – Vol. I	**Originator**	
12NC	5122 991 28375	**Name**	....
including update(s)	–	**Address**	....
			....
			....

Comment (if possible, add a copy of the page(s) affected by the comment, marked with the proposed changes).

Please return this form to SSS Publications,
P.O. Box 245, 7300 AE Apeldoorn, The Netherlands.