

**P800M Programmer's Guide 2**  
**Volume IV: Disc Real Time Monitor**

A publication of  
Philips Data Systems B.V.

Apeldoorn, The Netherlands

Publication number 5122 991 27404

November, 1978

Copyright © by Philips Data Systems, B.V., 1978  
All rights strictly reserved. Reproduction or issue to  
third parties in any form whatever is not permitted  
without written authority from the publisher.

Printed in The Netherlands.

This is volume IV of a four-volume set dealing with the Disc Operating System (non-real time and real time) for the P800M series. It describes the Disc Real Time Monitor.

The other volumes of this set, to be used in conjunction with this one, contain the following:

- Volume I: Disc Operating Monitor
- Volume II: Instruction Set
- Volume III: Software Processors

- Volume VI: Extended Disc File Management

Other books pertaining to the P800M series are:

P852M System Handbook

P856M/P857M System Handbook

P800K Operator's Guide

P800K Interface and Installation Manual

P800K Software Reference Data

Great care has been taken to ensure that the information contained in this manual is accurate and complete. However, should any errors or omissions be discovered, or should any user wish to make a suggestion for improving this manual, he is invited to send his comments, written on the sheet provided at the end of the book, to:

SSS-DOC

at the address on the opposite page.

---

---

## Table of Contents

---

PART 1 MONITOR USE . . . . .	1-1
Introduction . . . . .	1-1
<u>Chapter 1: Principles of Operation</u> . . . . .	1-3
<u>Chapter 2: Memory Organisation</u> . . . . .	1-7
<u>Chapter 3: Interrupt System</u> . . . . .	1-11
Hardware Interrupt Lines . . . . .	1-11
Levels 48 and 49 . . . . .	1-12
Software Priority Levels . . . . .	1-13
Dispatcher . . . . .	1-13
Stack . . . . .	1-13
<u>Chapter 4: Programming</u> . . . . .	1-15
Interrupt Routines . . . . .	1-15
Software Level Programs . . . . .	1-17
Memory Resident Programs . . . . .	1-19
Read Only Programs . . . . .	1-19
Program Save Area . . . . .	1-20
Swappable Programs . . . . .	1-20
Background Programs . . . . .	1-21
Level 63: Idle Task . . . . .	1-22
Scheduled Labels . . . . .	1-23
Reentrant Subroutines . . . . .	1-26
Time Slicing . . . . .	1-28
<u>Chapter 5: Disc Organisation</u> . . . . .	1-31
Sectors . . . . .	1-31
Files . . . . .	1-33
Access Modes . . . . .	1-35
Random . . . . .	1-35
Sequential . . . . .	1-35
Record Format . . . . .	1-36
Special Records . . . . .	1-38

File Types . . . . .	1-39
Load Module Size . . . . .	1-39
D:CI File Size . . . . .	1-40
Disc Structure with DADs . . . . .	1-41
Disc Structure without DADs . . . . .	1-51
DRTM System Disc Structure . . . . .	1-56
Flexible Disc . . . . .	1-57
<u>Chapter 6: Input/Output</u> . . . . .	1-59
File Codes . . . . .	1-60
Access Modes . . . . .	1-60
<u>Chapter 7: Data Management</u> . . . . .	1-61
Sequential Access Method . . . . .	1-63
Direct Access Method . . . . .	1-70
<u>Chapter 8: Operation</u> . . . . .	1-75
Loading Bootstrap and IPL . . . . .	1-75
Initial Program Loader (IPL) . . . . .	1-77
Starting the System . . . . .	1-80
Mounting an new Disc . . . . .	1-80
System Messages . . . . .	1-82
<u>Chapter 9: SCL Control Commands</u> . . . . .	1-85
Table of SCL Commands . . . . .	1-87
<u>Chapter 10: Operator Control Commands</u> . . . . .	1-101
Table of DRTM Operator Control Messages . . . . .	1-101
<u>Chapter 11: Monitor Requests</u> . . . . .	1-105
Table of Monitor Requests . . . . .	1-105

APPENDICES . . . . .	A-1
<u>Appendix A: System Generation</u> . . . . .	A-3
<u>Appendix B: Premark</u> . . . . .	A-39
Disc Premark . . . . .	A-39
<u>Appendix C: Peripheral Input/Output</u> . . . . .	A-41
<u>Appendix D: Control Unit Status Word Configuration</u> . . . . .	A-67
<u>Appendix E: PS52M Bootstrap</u> . . . . .	A-69
<u>Appendix F: Control Commands</u> . . . . .	A-75
<u>Appendix G: Operator Control Messages</u> . . . . .	A-77
<u>Appendix H: Monitor Requests</u> . . . . .	A-79
<u>Appendix I: Device Addresses</u> . . . . .	A-81

PART 2 MONITOR CONFIGURATION . . . . .	2-1
<u>Chapter 1: System Components</u> . . . . .	2-5
Nucleus . . . . .	2-5
System Interrupt Modules . . . . .	2-5
System Programs . . . . .	2-5
<u>Chapter 2: Internal Organization</u> . . . . .	2-9
Dispatcher . . . . .	2-9
Memory Organization . . . . .	2-9
Dynamic Allocation Area . . . . .	2-13
Swap Area . . . . .	2-13
Program Save Area . . . . .	2-13
Read Only Area . . . . .	2-14
Memory Resident Area . . . . .	2-15
Background Area . . . . .	2-15
File Format . . . . .	2-17
Swapping . . . . .	2-17
System Tables . . . . .	2-19
Communication Vector Table (T:CVT) . . . . .	2-20
Program Control Table (T:PCF) . . . . .	2-24
Software Level Table (T:SLT) . . . . .	2-29
Swapping Table (T:SWP) . . . . .	2-31
Device Work Table (DWT) . . . . .	2-31
Disc Control Table (T:DCT) . . . . .	2-35
T:HTP (Pool of HTBs) . . . . .	2-43
HTB . . . . .	2-44
T:DP (Pool of DADCTs) . . . . .	2-45
Logical File Description Table (T:LFT) . . . . .	2-50
File Code Table (T:PCF) . . . . .	2-53
Monitor Request Address Table (T:LRA) . . . . .	2-54
Resident Monitor Request Table (T:RMA) . . . . .	2-55
Timer Management Tables . . . . .	2-58
<u>Chapter 3: Input/Output</u> . . . . .	2-63
Tables . . . . .	2-63
I/O System . . . . .	2-64

IORM . . . . .	2-64
Driver . . . . .	2-65
EMBIO . . . . .	2-66
Service Routines . . . . .	2-67
 <u>Chapter 4: Monitor Modules : Flowcharts and Functional Description</u> . . . . . 2-71	
INIMON . . . . .	2-72
D:LDER . . . . .	2-74
M:DISP . . . . .	2-76
I:PPAR . . . . .	2-90
I:RTC . . . . .	2-94
I:LEM . . . . .	2-98
M:IDRM . . . . .	2-102
M:WAIT . . . . .	2-106
M:EXIT . . . . .	2-108
M:GBUF,M:PBUF . . . . .	2-113
D:CNTR . . . . .	2-116
M:ACT . . . . .	2-122
M:SWTIC . . . . .	2-124
D:ATDT . . . . .	2-126
D:GTIM . . . . .	2-130
M:RSRW . . . . .	2-132
D:CNST . . . . .	2-134
D:DNLV . . . . .	2-136
D:VOT . . . . .	2-138
ASGPRO . . . . .	2-140
DELPROM . . . . .	2-147
KEYPRO . . . . .	2-150
CANKEY . . . . .	2-152
D:SWP . . . . .	2-154
M:DPM . . . . .	2-156
D:CTPN,D:OCOM . . . . .	2-158
M:DMA . . . . .	2-162

MaDML	2-164
MaDCK	2-166
FeBLK	2-172
MaADD	2-174
ItTRAP	2-176

**PART 1**

**MONITOR USE**

---

The Disc Real Time Monitor (DRTM) is a disc-oriented system which is intended to supervise the execution of user application programs in a real time environment. It is assumed that these programs have been developed and prototyped under the DDM.

The system is based on a structure of priorities, incorporating hardware interrupt levels and software priority levels, where up to 48 hardware interrupt signals and 15 software user levels can be handled. This system is enhanced by the so-called scheduled label feature.

The monitor is of modular structure to allow the user to easily adapt it to his particular application.

The 16COM Disc Real Time Monitor has been designed to supervise the execution of pretested programs in a real time environment, on the basis of a priority system consisting of up to 48 hardware interrupt levels and 74 software priority levels. The DRTM is compatible with the Disc Operating Monitor (DOM), i.e. programs can first be developed and partly tested under the DOM, then used under the DRTM.

There is no memory protection, so all programs run in system mode. It is assumed that they have all been debugged and pretested. There is hardly any distinction between system and user programs; not all system programs are confined to any particular level, nor are user-written programs. It is therefore better to speak of standard system programs and application programs. In Part 2 of this manual some guidelines are given as to the interrupt levels to which some standard system programs can best be connected.

The priority system incorporates hardware interrupt lines and software priority levels:

- 0 to 47 are the levels for the hardware interrupt lines
- 48 is the level for interruptable monitor service routines
- 49 - 62 are software priority levels for system and application programs
- 63 is the idle task level.

The highest priority level is 0, so hardware interrupts will always overrule software level programs.

The interrupt routines which service the internal and external hardware interrupts are connected to levels 0 to 47.

Some of the interrupt routines are standard; the user can easily include interrupt routines written by himself, however. Incoming interrupts are handled by hardware and receive control on the basis of the

priority level to which they have been assigned. The dispatcher, a monitor module at level 48 which divides central processor time between competing priorities, also allots processor time to the user programs. The highest level active program always gets control until it is interrupted. Registers are saved by hardware in a system stack (addressed by register A15) and by software in the same stack or in a save area, which the system reserves in front of the program or in a special save area.

There are several types of programs, each related to a particular memory area: memory resident, read only, swappable and background programs.

Read only programs are loaded upon activation, one at a time; when a higher priority read only program is activated, the current one is erased. For this reason these programs must be read only.

Background programs are programs connected to the lowest priority level. Several of them may be in memory at the same time, allowing multiprogramming between them.

Swappable programs are executed on a time slice basis, according to priority. One such program at a time is loaded into a special partition and its execution started. If, at the end of a time slice a program of the same or higher priority has become active, the first program (in fact the whole swap partition) is swapped out, back onto disc, and the new one is loaded into memory. Otherwise the first program continues. Programs must be declared swappable before activation.

All these programs, in particular the read only ones, can make use of the dynamic allocation area to obtain temporary memory space dynamically. Not only programs, but also re-entrant subroutines will utilize this area. The dynamic allocation area is located behind the monitor area in memory. Programs obtain and release memory space in this area by issuing certain monitor requests.

User written programs and certain standard system programs have to be kept on disc and declared as memory resident or read only programs. Disc access can be done via a Data Management package in sequential or in random access mode. For physical I/O operations, drivers are used.

Under DMT the EDPM package can be used for accessing extended disc files. This is described in Programmer's Guide 2, Volume VI: EDPM

Files are stored on discs or DADs (Direct Allocation Devices). A DAD is logically considered as a whole disc, although physically it may be only part of a disc. For each user there is one library, pointed to by the user identification in the disc or DAD catalogue. All his permanent files are stored in this library, which is located on one disc or DAD only, i.e. the one containing the directory for that user. This implies that one user cannot have his files stored on several discs or DADs unless of course, he makes use of several user identifications. However, a user can have access to the system library and other libraries by specifying the user identification of the user of those libraries, but he can only read the data in those files and not write in them.

The files contained in a library may be of several types: source program files, object modules, load files (programs ready for execution) and files not belonging to any of these types, e.g. files created by user programs.

All I/O operations are initialized by monitor requests. Monitor requests can perform various functions. They consist of an LDN instruction followed by a DATA directive with a number as operand, to define the function. Necessary parameters must first be loaded into certain registers.

For I/O operations, several functions can be performed, such as binary, ASCII and object I/O, with conversion and control facilities, random read and write operations, etc. Peripheral devices or disc files are referenced through file codes, logical numbers of 2 hexadecimal digits.

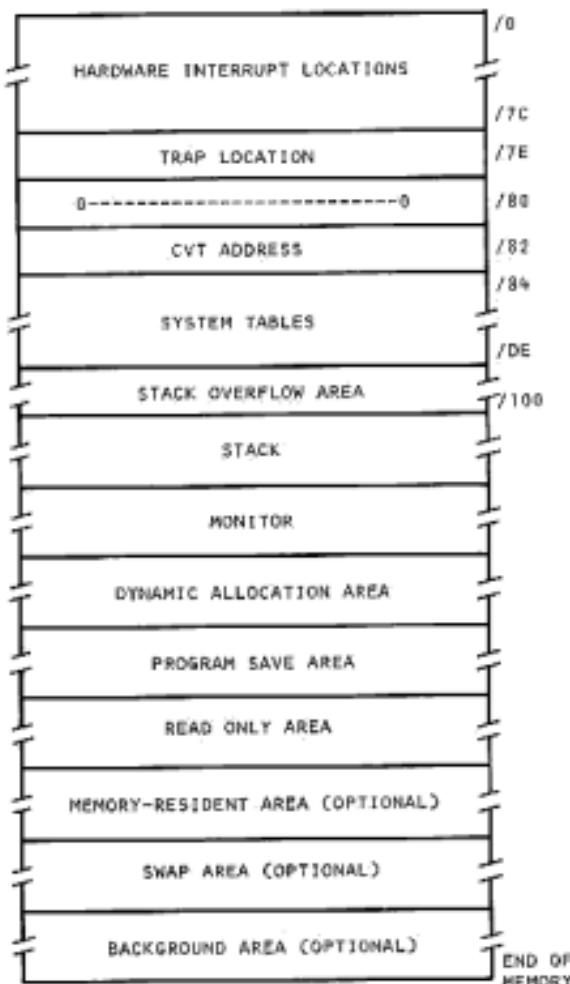
Monitor requests are also used to ask the monitor to perform the following functions:

- program activation
- waiting for the occurrence of an event
- program exit
- obtaining and releasing buffer space in the dynamic allocation area
- connecting programs to a priority level and disconnecting them
- switching from one program to another one at the same level to allow time slicing, i.e. allotting processor time to several programs on one level by turns, on the basis of timer interrupts
- getting the time and date
- informing the monitor of the occurrence of an event
- directing the monitor to wait with activation of a program until a certain, specified time
- reserving a peripheral device for exclusive use by one particular program; detaching that device from the program
- assigning and deleting file codes
- providing for the creating and use of unsolicited operator messages designed by the user himself.

A special feature, enhancing the multiprogramming aspect of the system, is the scheduled label, which is used in conjunction with monitor requests.

A scheduled label is a subroutine which is started when the monitor request function has been completed, although it is specified at the same time as the monitor request, i.e. the main program can continue while the requested function is performed. For example, combined with an I/O request, the main program continues while the requested I/O function is performed and the branch to the scheduled label routine is not made until that operation has been terminated. This can be very useful, for example to analyze the results of a monitor request.

The memory layout is as follows:



- Location /0 to /70 are hardware interrupt locations. They are hard-wired to internal and external interrupt lines. Each location contains the address of the interrupt routine required to service the interrupt connected to that location. The interrupt connected to location /0 has the highest priority (level 0).
- Location /7E contains the address of the trapping routine which handles simulation of certain instructions not included in the hardware (e.g. double add, double subtract, multiply, divide multiple load, multiple store and double shift).
- Location /80 contains 0.
- Location /82 points to the Communication Vector Table. This is a system table which contains information which may be of use to the user program. The layout of this table can be found in Part 2, chapter 2.
- Locations /84 to /DE are used for other system tables.
- The area occupied by the stack is defined at system generation time. When an interrupt occurs, P-register and PSW are stored by hardware and a number of registers by software. The number of registers stored depends on whether the interrupt routine servicing the interrupt runs in inhibit mode (anywhere from 0 to 15 registers) or in enable mode or branches to the dispatcher (always 8 registers). The A15 register always points to the next free location in the stack (where all information is stored towards the lower memory addresses). When A15 reaches the value /100 or becomes lower a stack overflow interrupt is given.
- The area after the monitor area is the user area. From the dynamic allocation area, which is reserved by the monitor at initialization time, blocks of memory space can be requested by the system or by the user. This is mainly intended for use by read only programs, although any user routine may request space in this area. For this purpose the program must give a 'Get Buffer' monitor request. When the use of the requested buffer is no longer required, a 'Release Buffer' monitor request must be given. The size of this area must be large enough to avoid a dead-lock in the system. Programs requesting space in this area might be

put in wait state, if such a request cannot be serviced immediately.

The Read\_Only\_Area is used by the disc resident read only programs. These programs are loaded dynamically into this area, one at a time. When a read only program of higher priority than the one currently in memory is requested, the latter is erased by the former. For this reason these programs are read only and must use the dynamic allocation area for buffers, work areas, etc. This partition must be at least 1/800 words long. It may be used by standard system programs, such as the operator communication package and certain monitor request modules or by application programs. These programs must previously have been declared as read only.

The Program\_Save\_Area is used to save the registers of a program when it is interrupted. Its size can be specified at system generation time and must be such that it can contain all save areas of all programs, including system programs.

This save area must also provide space for saving registers in case one or more scheduled labels are included in the program. The Memory\_Resident\_Area is an optional partition. If used, it contains programs which are urgently required, i.e. interrupt routines or high priority real time programs. These programs are loaded on operator request, remain resident in memory and are scheduled according to their priority levels.

The Swap\_Area is used to load swappable programs into memory, one at a time, on the basis of priority level and time slicing. These programs, when swapped, are written back onto disc, so they may need a work area in the Dynamic Allocation Area.

The Background\_Area is optional as well and is used solely for programs connected to level 62. These are loaded dynamically and several of them may be in this area at the same time, to allow for multiprogramming between them. No program swapping takes place. When execution of a program is completed, its memory space will be released automatically. The size of the monitor area is calculated by the system at loading time and depends on the configuration.

The size of the other memory areas can be defined at system generation time, but they may be modified at system loading time, when the monitor is initialized.

Programs and routines under DDTM run on the basis of an interrupt and priority system consisting of up to 64 levels. These levels are subdivided as follows:

0-47: levels for interrupt routines connected to the hardware interrupt lines	}	hardware interrupt lines
48 : interruptable monitor service routines		software priority levels
49 : high-priority system or user programs	}	software priority levels
50-63: user and system programs:		
50-61: foreground programs		
62 : background programs	}	software priority levels
63 : idle task		

Level 0 has the highest priority, 63 the lowest, so all hardware interrupts always have priority over the software levels.

#### HARDWARE INTERRUPT LINES

The interrupt lines are connected to memory location /0 to /70. These locations contain the addresses of the interrupt routines which service the internal and external interrupts.

For the interrupta the user can define the priority levels at system generation time.

The following priorities are recommended for the various interrupt lines:

/0 : power failure	- (interrupt location /00)
/1 : LEM/stack overflow	- (interrupt location /02)
/2 : real time clock	- (interrupt location /04)
/3 : floating point processor	etc.
/4 : not used	

```
/5 : not used
/6 : operator's console
/7 : control panel
/8 - /E: Data Communication on Programmed Channel
/F : punched tape reader
/10: CDC disc (address /16)
/11: X1215 disc (address /02)
/12: flexible disc (address /05)
/13: magnetic tape
/14: cassette tape
/15: card reader
/16: tape punch
/17: line printer
/18: X1215 disc (address /01)
/19: CDC disc (address /17)
/1A - /1F: Data Communication on I/O Processor
/27: page fault
```

#### Levels 48 and 49

Level 48 is handled by the system as a pseudo-hardware interrupt level, i.e. programs on this level also use the A15 stack and have the same interfaces as interrupt routines. Usually they are intended for processing a higher level interrupt, which does not require servicing at that level. For example, an I/O request is entered via an LKM interrupt. As soon as the request is recognized, the I/O program branches to level 48, so that other interrupts may be serviced, while the I/O request is handled. Most of the I/O interrupts are received at high priority levels, but processed at level 48.

Level 49 is used especially by the system to process a number of monitor requests. However, any user program may use this level. System and user programs are not distinguished at this level, they are dispatched in the same way. This implies, that user programs at level 49 may influence the response time of system programs at level 49 in a real time system.

### Software Priority Levels

The levels 50 to 62 are connected to user programs which may or may not be related to each other. The programs are activated by control command or monitor request, in which case it is possible for one program to activate another one.

The levels 50 to 61 are reserved for foreground user programs, while level 62 is reserved for background user programs, i.e. programs which have the lowest priority in the system. Actually these are programs which have not been connected to a software level. They are automatically loaded as background level programs.

Note: It is possible to connect a foreground program (core-resident, read-only or swappable) to level 62. However, when such a program exits, it is 'erased', as is done with background programs.

By means of the monitor request 'Switch inside a Level' it is possible to divide processor time among several programs on one level, on the basis of real time clock interrupts.

Requests for program activation are handled by the 'Activate' monitor module, then passed on to the dispatcher (see below).

Level 63 is reserved for the idle task, an instruction sequence to use up idle central processor time. See also Chapter 4: Programs.

### Dispatcher

The dispatcher is a monitor module (M:DISP) running on level 40, which divides central processor time by starting programs according to their priority.

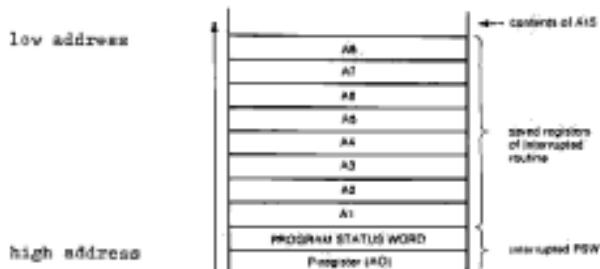
The dispatcher can be entered only from an interrupt routine, i.e. from a level below 40, such as the I/O interrupt and monitor request handlers.

### System Interrupt Stack

When an interrupt occurs, certain information about the interrupt-ed program or routine must be saved before the interrupt can be serviced.

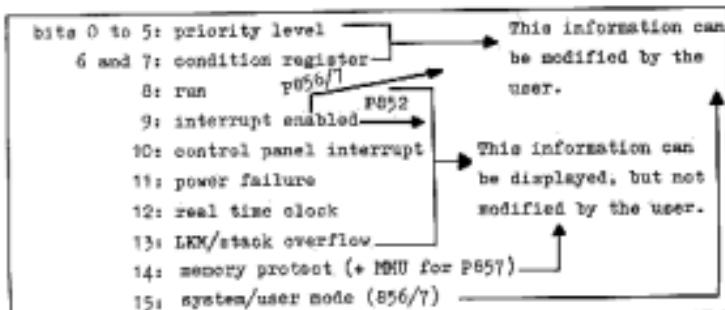
This is done in the system stack, the address of which is held in register A15. The start address of this stack is defined at system generation time and it is built in a downward direction in memory, i.e. towards the lower addresses. The A15 register always points to the first free location in the stack. Upon interrupt, PSW and P-register are always saved in this stack and also a number of registers:

- any number if the interrupt routine runs in inhibit mode and takes care of restoring the registers itself;
- registers A1 to A8 if the interrupt routine runs in enable mode or ends with a branch to the dispatcher, because the dispatcher always handles the stack on this basis;



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

The Program Status Word, stored in the stack upon interrupt, contains the following information:



There are several types of user-written programs:

- interrupt routines, servicing one of the hardware interrupts and connected to any of the hardware levels 0 to 47
- subroutines (reentrant or non-reentrant); they are called through a CP (Call Function) instruction and run on the level of the calling program
- programs, connected to any of the software levels 49 to 63. This category is subdivided as follows:
  - \* foreground programs are connected to any of the levels 49 to 61, programs on level 62 operate in the background area of memory, level 63 is reserved for the idle task.
  - \* foreground programs can be either core-resident, i.e. always present in memory, or read only or swappable, i.e. loaded on request. This implies that for each of these program types a different memory partition must be used.

#### INTERRUPT ROUTINES

User interrupt routines must have been connected to one of the interrupt locations in memory (locations /0 to /70). That is, the address of the interrupt routine must be placed in one of these locations, which at the same time determines the priority level of the interrupt routine. i.e. the interrupt routine whose address is loaded in location /0 has the highest priority (0).

The interrupt routine address may be loaded into this location in several ways. One of them is to link-edit and include the routine with the rest of the system modules at system generation time. It can also be done via the LB command (see chapter 3).

When an interrupt occurs, the P-register and PSW of the interrupted program are stored by hardware in the system stack pointed to by the A15 register (see Ch.3) and the system is put in inhibit mode.

Then the interrupt routine receives control and from within the routine the user may store any other registers by software, if he wishes. The interrupt routine may now continue in inhibit mode, or if the user decides that other interrupts must be able to overrule the current one, he may set the system to enable mode by giving an ENB instruction. (Note: If an INE instruction immediately follows the ENB instruction, a dummy instruction must be inserted, because external interrupts are scanned every two instructions. This dummy instruction may, for example, be another ENB, so the correct sequence becomes: ENB-ENN-INE). This, however, entails a substantial difference in the handling of the system stack. If the routine runs in inhibit mode from beginning to end, any of the registers A1 and A14 can be used, provided the user first takes care of storing old contents in the A15 stack and restoring them at the end of the routine. This may, for example, be done as follows:

STR	A1,A15	(On P056/7 and when the simulation routine for multiple load/store has been selected at sysgen for P052, the sequence is:
STR	A2,A15	
STR	A8,A15	
		MSR 8,A15
coding		coding
		MLR 8,A15
LDR	A8,A15	RTW A15 )
LDR	A7,A15	
LDR	A1,A15	
RTW	A15	

This is the case of an interrupt routine in inhibit mode with a normal return via the A15 stack. The RTW via A15 results in an automatic enable.

However, if other interrupts are to be enabled during a routine or the user makes an absolute branch from the interrupt routine to the dispatcher (ABL M:DISP; for dispatcher address: see CPT), he must

take care that before the EXB or ABL instruction is given, the A15 stack contains only P, PSW and registers A1 to AB inclusive, because on this basis the A15 stack is handled.

#### Conventions

- Interrupt routines must start by saving the old contents of the registers to be used in the routine.
- Before returning via A15, the old register contents must be restored.

If a branch is made to the dispatcher, the stack must contain P, PSW and register A1 to AB, so any other registers used, must have been restored before making the branch.

- In case of an interrupt routine for internal interrupts (LDM/ stack overflow, real time clock, power failure, control panel), an RIT instruction must be given at the beginning of the routine, to reset the interrupt. See Volume II.

#### SOFTWARE LEVEL PROGRAMS

Software level programs under the DRTM are those application or standard system programs which are connected to any one of the levels 49 to 63.

The programs required for a real time process may be used by one or more other programs. They are loaded into memory as separate programs and the connections between the programs which are related to each other are supplied by monitor requests in the individual programs.

A program is connected to a level either at loading time (CM Control Command) or by a monitor request (Connect a Program to a Level - LKM20). The level is registered in the PSW (bits 10-15). Program activation is also possible in two ways:

- by Control Command, at initialization time.
- by an 'Activate' monitor request (LKM12) from another program.

After activation, the dispatcher will start the programs according to their priority.

Besides being connected to a level, programs can also be connected to a timer, so that they may be activated at a certain time or after regular intervals, according to parameters which the user specifies in a monitor request 'Connect a Program to a Timer' (LKM10).

It is also possible to share central processor time between various levels and between memory-resident programs of the same level (including background programs, once they have been loaded into memory) by time slicing, i.e. the monitor request 'Switch Inside a Software Level' (LKM15), see below. The 'scheduled label' is another feature which extends the possibilities for real time programming, see below.

After a program has been loaded, all relevant information about it is stored in a Program Control Table (see Part 2, Chapter 2) in the PCT Pool inside the monitor area in memory.

The program remains under monitor control until it is disconnected from its level, during which time it passes through various states, as recorded in the PCT:

- inactive: the program has been connected to a level, but it has not been called yet;
- active: the program has been called, and is not yet terminated;
- wait for execution: the program is ready to use central processor time when it has the current highest priority;
- wait for an event: the program has given up control voluntarily with a 'Wait for an Event' monitor request (LKM2) to wait for the occurrence of a particular event.

#### Memory resident Programs

Aside from interrupt routines, some high priority foreground programs may also require to be memory resident. They should be so, when their activation, in response to a request, cannot be delayed by the time required for loading such a program. These programs are loaded at initialization time by control command LD, which implicitly defines them as memory resident. They occupy a special memory partition (see Chapter 2).

A 17-word save area is allocated in the program save area (see below). When scheduled labels are used,  $17 + 1 + 16 + n$  words are saved, where  $n$  is the maximum number of scheduled labels allowed in queue.

#### Read only Programs

Programs, for which the response time is not so critical, can be declared read only by the control command RO, given at initialization time. They are loaded later on, on request, into a special partition (see Chapter 2), which must be at least 1880 words long. Only one program at a time occupies this partition. When a higher priority read only program is activated, the current read only program is erased by loading the higher priority one. After this one has terminated and made its exit, the former read only program is resumed.

CPU time is therefore not shared between programs in this partition. Since a read only program can be interrupted and erased at any time, certain provisions must be made.

These programs are read only, so they may not contain any locations that may be modified, any work areas, I/O buffers. For these, the dynamic allocation area must be used, so that this information is always safe in case a read only program is erased by another one. Space in the dynamic allocation area may be obtained and released by means of the monitor requests Get Buffer (LEM4) and Release Buffer (LEM5).

Moreover, the system provides for a save area partition.

In this area the contents of the erased read only program's registers and PSW are saved. The size of this area must be specified

at system generation time and must be large enough to contain the save areas of all user and system programs (see Part 2).

This save area is managed by the system.

Finally, Read only programs should not give a Wait request for an event which must be set by another read only program of lower priority, for any Wait request will lock the read only area to other read only programs of lower priority.

#### Program Save Area

The program save area is reserved to store the context (P-register, PSW, registers A1 to A14) of programs running at one of the levels 49 to 63, in case of a priority change. This may be the case when control is given to a higher priority program or, if the current program is waiting for an event, control may also be given to a lower priority program.

(If a program is interrupted, its context is saved in the system stack addressed by register A15).

The save area consists of 17 words for programs not using any scheduled labels or  $17+1+16+2n$  words for programs using scheduled labels, where  $n$  is the maximum number of labels which may be scheduled at the same time (see also Part 2).

The save area is allocated by the system command language SCL, each time a foreground program is declared (or loaded) or a background program is activated.

#### Swappable Programs

A swappable program is executed one at a time, in a special partition in memory, from which it may be swapped back onto disc to make room for another swappable program. Program swapping may occur in this area when a swappable program of a higher or equal priority level is activated and, at the same time, a time slice has elapsed.

A time slice is a predefined period of time during which a swappable program is allowed to run. The swappable program which is in

memory is swapped at the end of a time slice for another one with the same or a higher priority level, unless it is the only program on the current level and this is also the highest active level, in which case the program continues for another time slice.

There is a difference with read only programs in that swappable programs are not erased but written back onto disc integrally. However, swappable programs may have their buffers and ECB's allocated in the dynamic allocation area, e.g. for wait with swap-out, non-disc I/O. The response time is much the same as with read only programs, however, because the swap operation will take a certain amount of time, depending on the speed of the disc and on the size of the swap program.

Any foreground program from level 49 to 61 may be declared swappable. This is done when it is loaded, by means of the control command SW. The length of the time slice can be defined at System Generation, in the SW command or by means of a separate command TS.

#### Background Programs

These are programs of the lowest possible priority level (62), i.e. time consuming programs of which it does not matter if their execution is delayed.

Background programs are loaded automatically when a request for their activation is made. They need not be connected to a level before activation, for they will automatically be loaded as level 62 programs. It is possible to have more than one program in the background area at the same time, so multiprogramming is possible between background programs. When execution of a background program is terminated, the memory space it occupied will be released automatically.

More detailed information about the internal management of the memory partitions is given in Part 2.

### Level 63: Idle Task

Level 63 is reserved for the idle task, an instruction sequence to use up idle CPU time. It is memory resident and consists of 2 instructions:

```
RF  m+2
RR  m-2
```

Under DRTM this sequence can be replaced by another one, for example measuring the system's performance and computing the CPU idle time. It must have the same interface as other user programs, with the following exceptions:

- No monitor requests must be used which will put the program in wait state, explicitly (Wait for an Event) or implicitly. Therefore, I/O to output calculation results should preferably take place in a program at another level, e.g. 62.
- The idle task may not use the Exit request, since it must consume all idle CPU time and must loop inside the program.
- The entry point must be IDTASK and it must be processed with the user Library before the system Library at SYSGEN.

If no monitor requests are used, the response time of the system will not be affected by the idle task.

It will be clear from the above subdivision that, although a program may be declared in any class, the main reasons for deciding on program declaration will be response time and calling frequency.

#### SCHEDULED LABELS

The scheduled label is a feature which allows the user to do a sort of multi-tasking by attaching a routine to a monitor request. To this end, the user specifies the monitor request as having the two's complement of the DATA number indicating the monitor function which is to be executed, followed by the label of the routine which must be executed upon completion of the request. For example:

Normal I/O request:	with scheduled label:
LDX A7,CODE	LDX A7,CODE
LDKL AB,SCDLAB	LDKL AB,SCDLAB
LEM	LEM
DATA 1	DATA -1, SCDLAB

In this case, the routine SCDLAB is to obtain control on completion of the I/O monitor request specified. When a scheduled label is attached to an I/O request, one should not set the wait bit, so that the program can continue concurrently with the I/O operation requested. When that operation is finished, control is passed to the SCDLAB routine, and the P-register, PSW and registers A1 to A14 are stored in the 16-word save area (SAVADR) in front of the user program. (When entering a scheduled label routine, only the contents of A8 is significant: ECBADR) When the routine is finished and exits, control is returned to the monitor, which passes control to a possible following scheduled label routine, or back to the main program by restoring the registers and PSW from the save area. This can be illustrated as follows:



Any number of scheduled labels may be given in a program. However, it is possible that one scheduled label is blocking execution of another one because it is active, i.e. using central processor time. In such cases the address(es) of the queued scheduled label routines are temporarily stored in a table (FILLAB). The maximum number of scheduled label addresses which may be stored in this table at any one time is the number defined at system generation time. This is the only restriction. Queued scheduled labels are treated on a first-in-first-out basis.

Note:

Although it is possible to give a Wait monitor request within a scheduled label routine, this is not normally recommended, for it blocks execution of the whole program.

Example:

This example illustrates how scheduled labels are queued in the FILLAB table, when their execution is blocked in case of an I/O operation on a slow device.

In a program there are three monitor requests for output onto tape punch, typewriter and line printer. To each of these requests a scheduled label is attached. Each of these scheduled labels requests output on the typewriter. In such a case, it will be evident that the line printer output will be finished first and thus that the scheduled label attached to that request will receive control first. Now, when the other two output operations in the main program are finished, the scheduled labels attached to them will be queued in FILLAB, for they are also requesting output on the typewriter which is still busy with the last scheduled label. When that one is finished control is passed on the last one entered in the FILLAB table.

Note, that in this case the third scheduled label is the first to receive control, because the I/O to which it was attached was for line printer and terminated before the other two.

```

IDENT SEAB
HUE1 1DATA "1234567890"      SPECIFYING THE CHARACTERS TO BE
HUE2 1DATA "ABCDEFGHIJ"  OUTPUT
HUE3 1DATA "ZYXWVUTSRQ"
HUE4 1DATA "IJKLMNOP"
HUE5 1DATA "BDFHJL"
HUE6 1DATA "BDFG"
HUE81 1DATA 5,BUF1,5,0,0,0  DECLARING THE EVENT CONTROL
HUE82 1DATA 5,BUF2,40,0,0,0  BLOCKS FOR THE OUTPUT OPERATIONS.
HUE83 1DATA 3,BUF3,5,0,0,0  5 IS THE FILE CODE FOR TYPEWRITER.
HUE84 1DATA 2,BUF4,12,0,0,0  3 FOR PUNCH, 2 LINE PRINTER.
HUE85 1DATA 5,BUF5,6,0,0,0
HUE86 1DATA 5,BUF6,6,0,0,0

START LDK A7B
LDKL A7C,B1
LKM
DATA 1,SLAB1
LDK A7B
LDKL A7C,B2
LKM
DATA 1,SLAB2
LDK A7B
LDKL A7C,B4
LKM
DATA 1,SLAB3
LKM
DATA 3      MAIN PROGRAM STARS AND REQUESTS
              STANDARD OUTPUT OF BUF1 ON THE
              TYPEWRITER. SCHEDULED LABEL SLAB1
              ATTACHED THIS REQUEST.
LDK A7B
LDKL A7C,B3
LKM
DATA 1,SLAB2
LDK A7B
LDKL A7C,B4
LKM
DATA 1,SLAB3
LKM
DATA 3      MAIN PROGRAM REQUESTS STANDARD
              OUTPUT OF BUF3 ON THE TAPE PUNCH.
              SCHEDULED LABEL SLAB2 ATTACHED
              TO THIS REQUEST.
LDK A7B
LDKL A7C,B5
LKM
DATA 1,SLAB4
LDK A7B
LDKL A7C,B6
LKM
DATA 1,SLAB5
LKM
DATA 3      MAIN PROGRAM REQUESTS STANDARD
              OUTPUT OF BUF4 ON THE LINE PRINTER.
              SCHEDULED LABEL SLAB4 SLAB5 ATTACHED
              TO THIS REQUEST.

SEAB1 LDK A7B
LDKL A7C,B2
LKM
DATA 1
LKM
DATA 3      SLAB1 REQUESTS STANDARD OUTPUT
              OF BUF2 ON THE TYPEWRITER
              AND EXITS
SEAB2 LDK A7B
LDKL A7C,B4
LKM
DATA 1
LKM
DATA 3      SLAB2 REQUESTS STANDARD OUTPUT
              OF BUF4 ON THE TYPEWRITER
              AND EXITS
SEAB3 LDK A7B
LDKL A7C,B6
LKM
DATA 1
LKM
DATA 3      SLAB3 REQUESTS STANDARD OUTPUT
              OF BUF6 ON THE TYPEWRITER
              AND EXITS

END START

```

### Re-entrant Subroutines

Re-entrant subroutines are subroutines which can be used by more than one program. They can be interrupted and restarted for a higher priority program at any time.

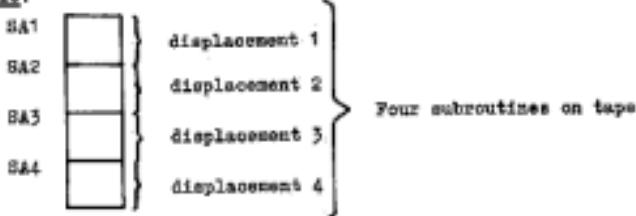
These subroutines must be loaded into memory in such a way, that the programs which will use them, can find their start addresses. To this purpose they can be link-edited with each other (a dummy operation). At the end of the process the Linkage Editor will print a MAP. This map will enable the user to find the displacement of each of the subroutines.

Now the user can write a dummy program with the start addresses of the subroutines used by his program.

This dummy program is loaded and then link-edited with all the programs which use the loaded subroutines, so that these programs know the subroutine start addresses.

Note: Direct addressing is not possible, it is recommended to use the registers for this purpose.

#### Example:



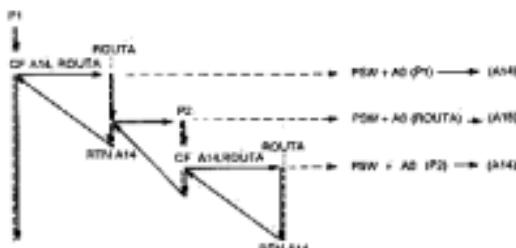
#### Dummy program:

```
    INENT    DUPR
    ENTRY    SA1
    ENTRY    SA2
    ENTRY    SA3
    ENTRY    SA2
    SA1 EQU    /2500
    SA2 EQU    SA1 + 'displacement 1'
    SA3 EQU    SA2 + 'displacement 2'
    SA4 EQU    SA3 + 'displacement 3'
    END
```

This dummy program must be linked with each program using any of the four subroutines. The programs, for example, are as follows:

```
IDENT      PROG
|
EXTEN      SAI
EXTEN      SAA4
|
CF          A14,SAA4
|
CF          A14,SA1
|
END
```

If a re-entrant subroutine wishes to call another re-entrant subroutine, it must reserve two words in the dynamic allocation area through the 'Get Buffer' monitor request. The subroutine obtains any other memory space it requires, in the same manner. Re-entrant subroutines cannot use scheduled label routines. If a subroutine is interrupted for another program of a higher level, requiring the same subroutine, we get the following situations:



- Program P1 starts running.
- CF is given, PSW + P of P1 are stored in stack, of which the address is given in A14 and subroutine ROUT is started.

- Interrupt for program P2, PSW and P of ROUT are stored in the stack specified in A15 and P2 starts running.
- CF is given, PSW + P of P2 are added to the stack specified in A14 and subroutine ROUT is started and allowed to terminate with RTN A14.
- P2 is resumed and allowed to terminate normally.
- Via A15 subroutine ROUT is resumed and allowed to terminate with RTN A14.
- P1 is resumed.

Note: the stack indicated by A14 has to be reserved by the user; the stack indicated by A15 is reserved and handled by the monitor.

#### Time slicing

For some software level programs time slicing is possible, i.e. allotting processor time to several programs connected to one level, under control of a timer and by turns. This is allowed only for memory-resident programs and background programs once they have been loaded and functions as follows:

Let us assume that level 52 is connected to a timer, e.g. the real time clock, and that a program on level 52 contains the monitor request which activates the switching procedure:

Real Time Clock  $\rightarrow$  52  $\rightarrow$  Switching program:

LDK A7, 55

LEM

DATA 13

This calling sequence requests switching on level 55.

LEM

DATA 3 'Exit' monitor request to go to level 55.

Level 55 has three programs connected to it: A, B and C. At an interrupt of the real time clock (every 20 milliseconds), the program connected to level 52 performs the switching request and exits. The dispatcher then activates one of the programs A, B or C on level 55, to which ever one it was pointing. This program then starts running:

- until an interrupt starts an interrupt routine (level 0 to 47) or until it is required to start level 53 or 54, e.g. because an I/O operation for which one of them was waiting is ready.
- until 20 milliseconds have passed. Then the real time clock gives another interrupt and control is returned to the program on level 52. Another request is performed, level 52 exits and, barring higher priority interrupts, the next program on level 55 is started, to run until another 20 msec. have passed or a higher priority interrupt occurs.

This sequence continues until the programs A, B and C have had enough processor time to be executed.

If, for any reason it is desired to limit the interrupts during switching as much as possible, the switching program can be connected to level 50 and the programs to be switched to level 51. In this case, only hardware interrupts can interfere.

Note: This request is also used by some of the monitor modules, i.e. those which handle the monitor requests Activate, Exit, Detach Device, Release Buffer and Input/Output.

In a swappable program the time slice counter can be forced to zero by some monitor requests:

- Wait for an Event/Wait for a Given Time: if bit 15 of EOB/Param Block is set to one, the time slice is set to zero.
- I/O with Wait: if bit 7 of the order is set to one, the time slice counter is set to zero.

To understand this chapter it is necessary to define some concepts first.

The disc organization is based on sectors, which are sections of a track with a length of 205 words, of which 200 are usable.

On the basis of this sector concept, the following definitions must be kept in mind.

- Disc Sector Physical Address (DSPA):

the physical address of a sector on the disc, i.e. the address of a sector in a consecutive arrangement.

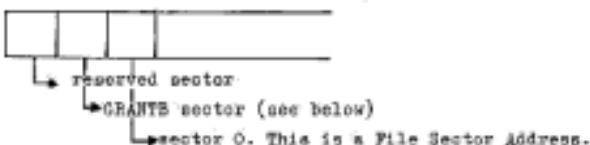
- Disc Sector Logical Address (DSL):

the address of a sector on the disc as calculated by interleaving, which is ordering the sectors in such a way as to make access as efficient as possible (see below).

- File Sector Address (PSA):

the address of a sector inside a logical file as managed by Disc File Management (DFM).

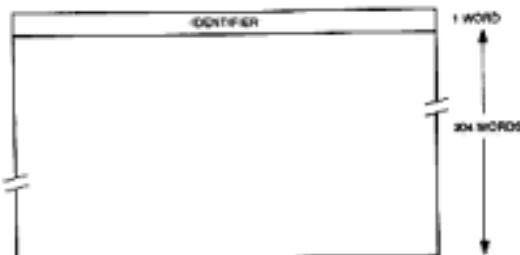
Inside a file, the sector numbering starts at the third sector:



The types of disc which can be used in a P800 configuration are X1215 and X1216 discs, C60 discs (40M and 80M) and flexible disc. The flexible disc is considered as only a physical device, so Data Management (see Ch. 7) and the DAD concept (see Ch. 1 and below) do not apply to it.

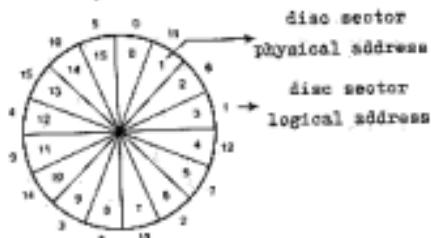
SECTORS

The basic unit in the organization of all types of disc used is a sector. All access operations take place at sector level. The sectors are numbered from 0 to n, consecutively. They can be accessed directly from any program through the I/O driver. A sector has the following format:



The identifier is written in the first word of every sector only on X1215 or X1216 discs by the Disc MARK program. On CDC discs no identifier is written. For moving head discs it contains the number of the cylinder in which this sector is located. For fixed head discs its value is not significant. The identifier is used by the system to check if a seek operation has been successful or not. The first word is set by the driver when a sector is written onto the disc. This word must not be modified during the I/O transfer, or a 'seek error' status may be returned later, when the same sector or another one in the same cylinder is accessed. Although physically the sectors on a disc are numbered consecutively (Disc Sector Physical Addresses), for logical handling they are numbered in a different manner (disc sector logical addresses). This is done to give a requesting program enough time to process the current sector before requesting the next one in a sequential process.

For these logical sector addresses the sectors are interleaved, e.g. on a factor-5 basis for discs with 16 sectors per track (X1215/16).



16 sectors

The disc sectors are always, except in one case, handled according to their logical addresses, e.g. all Data Management operations take place on this basis and when a disc dump is made, the sectors are dumped in their logical order. Only with disc error messages is the sector address indicated the physical address.

#### FILES

We have seen that the term file sector address takes into account that the first two sectors of a file are reserved: one for file header and one for a table called GRANTB, so that sector addressing starts with the third sector, which gets file sector address 0. Space allocation on the disc for a file is done on the basis of granules, where each granule is an area of 8 consecutive disc sector logical addresses, i.e. 8 logically consecutive sectors. A file is always stored on an integer number of granules, so one granule cannot be shared by two or more files. The system allocates as many granules as necessary to a file which is being written. These granules are chained and attached to the file code assigned to that file.

The addresses of the granules allocated to a file are kept in the table GRANTB in the second sector of the first granule of that file.

The sector GRANTB is initialized when a file code is assigned.

GRANTB is filled with the addresses of the allocated granules, any remaining words being set to zero.

For sequential access, when an attempt is made to write onto a granule which has not yet been allocated, a new one is allocated and GRANTB is updated. The system manages a table called BITAB which contains one bit for each granule on the disc or BAD. This table is copied into memory when a disc pack is loaded and updated in core and written back onto disc when a Keep File command is given. A bit is set to 1 when the corresponding granule has been allocated and it is 0 when the granule is still free. Allocation takes place via this table, after which the granule address is stored in GRANTB.

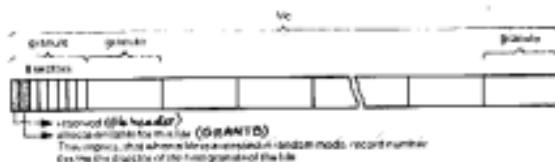
GRANITE	0	IDENTIFIER
	2	LENGTH
	4	ADDRESS OF GRANULE 0 OF THE FILE
	6	ADDRESS OF GRANULE 1 OF THE FILE

In this table, location  $2(i+2)$  is the address of the  $i$ th granule. If the granule has not yet been allocated, the location contains the value zero.

The granule address is a binary value from 0 to  $n$ , representing the relative sector address, from the beginning of the disc, of the first sector of the granule. GRANITE is 200 words long.

**Note:** Random access files of more than 200 granules are possible, but when such a file is deleted under the Disc Operating Monitor (DOM), only the first 200 granules are deleted.

Finally, a file is organized as follows:



Inside the file, the useful area in a sector is 200 words (see Record Format).

The first two granules on a disc or DAD are always reserved and used by the system for catalogues and libraries.

### ACCESS MODES

Files can be accessed in two ways: random and sequential, each access mode requiring a different organization of the file.

#### Random

This type of organization has the following characteristics:

- records are of fixed length: one logical record-one physical record-one disc sector.
- the records in the file may be organized in any manner. Access takes place according to the file sector address, i.e. by the relative position of the record in the file (not counting the first two sectors, which are the file header and GRANTS). The logical sequence of the records is not identical to the physical sequence. Such a file is a keyed file, the relative number of the record (sector) being the key.
- when a random file is created under DFTM, the requested number of granules is allocated and the file cannot be extended, unless it is completely rewritten.
- Random access files should be Assigned, Kept and Deleted under DFTM, not under DQM. See Note on previous page.
- records are accessed directly by specifying the file sector address.
- records may be retrieved, updated and restored individually.

#### Sequential

This type of organization has the following characteristics:

- records may be of any length, up to 16k words. The sector format of such records is described below.
- the only relation between the records in a sequential file is their sequence. The records of such a file must be presented in the order in which they must be written onto the disc, i.e. the

logical sequence of the records is identical to the physical sequence.

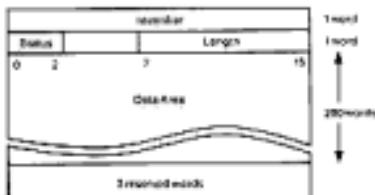
- to retrieve a record, the whole file must be scanned up to the desired record.
- to update a record, the file must be processed as a whole.

Additional details on Access Mode are given in chapter 7, Data Management.

#### Record Formats

The format of physical record (sector) has already been described (see SECTORS). This is the record format for random files.

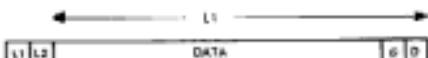
In sequential files a sector may contain a logical record or it may be part of a logical record. Sectors have the following format in these files:



- **status** consists of 3 bits:
  - bit 0: if 1, this sector has been deleted
  - bit 1: if 1, an EOS (End-of-Segment) record has been written in this sector. This record is the last one in the sector, for the first record following an EOS always starts at a new sector.
  - bit 2: if 1, EOF (End-of-File) record has been written in this sector. An EOF record requires a full sector without any other records in it. So when an EOF is written for a file, first the current sector buffer is written, if necessary, and then an additional sector for the EOF record.
- **Length**: specifies the length, in characters, of the used area of this sector.

- the data area contains data written in either sequential or random access mode.
- the last 3 words are not used.

The logical records in sequential access files are always compressed and blocked by the system to save disc space (trailing blanks are removed). The format of these logical records is as follows:



L1 is the length of the record, including the 2 words S and D, but excluding L1 and L2:



V = 1: this record has been deleted from the file

Q = 1: this record is an EOF record

P = 1: this record is an EOP record

L2 is the initial record length, in characters. This is the requested length recorded in the EOB when the I/O request for writing this record was made.

S is the file sector address within the file containing the first word of the record.

D is the displacement in characters in the sector S, of the first word belonging to the record.

Data is up to 1600 words long (one granule), trailing blanks removed.

### Special Records

There are some special records in sequential files, which have the following format (cf. Record Format) (values in hexadecimal):

:BOS:  $\underbrace{4004}_{L1} - 0 - S - D$

:EOF:  $\underbrace{2004}_{L1} - 0 - S - \underbrace{4}_{D}$

Blank card:  $\underbrace{0004}_{L1} - \underbrace{0050}_{L2} - S - D$   
(hexadecimal 50=decimal 80=card length)

Note: Records on disc always consist of an even number of characters. So, whatever the value of L2 given by the user at creation time, L1 always represents an even number of characters, because when the requested length is an odd value, a dummy character is added to the record. An BOS is always stored in one sector and must be the last record in the sector.

An EOF is always written in a separate sector.

## FILE TYPES

Under the DDTM, the following file types can be distinguished:

- all programs must be kept under load module format and they are identified as file type LM. Such files must be created either under DDM or at system generation.
- other kinds of files are considered as data files and are identified as file type UF. These files can be created and kept under the DDTM itself.

## Load Module Size

All system or user programs must be kept on disc in the load format generated by the Linkage Editor or at system generation.

The number of granules required for keeping a program is  $((S-1)/188+2)/8+1$

S being the program size in words. Thus, the following table gives the maximum program size for a number of granules:

<u>Number of Granules</u>	<u>Maximum Program Size (Words)</u>
1	1128
2	2632
3	4136
4	5640
5	7144
6	8648
7	10152
8	11656
9	13160
10	14664
11	16168
12	17672

- All system read only programs are smaller than 1960 words, so each one requires one granule.
- The monitor will generally require from 5 to 12 granules.

D:CI File Size

This file is used by the system to keep all read only and swapp-  
able programs in core image format.

The size of this file is declared during the BEMGEN phase at  
System Generation time.

Each sector contains 200 program words, so for each system or user  
read only or swappable program, the required number of sectors in  
the D:CI file is  $(SRO - 1)/200+1$

SRO being the program size in words.

- As all system read only programs are smaller than 1880 words,  
each one requires 6 sectors on the D:CI file, i.e. 8 granules  
for the 10 system programs.

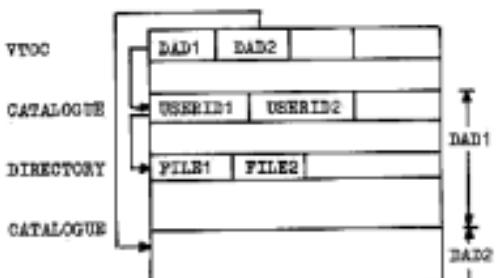
### DISC STRUCTURE with DADS

In such a structure there are three levels of disc access:

- to the physical disc, performed by the driver.
- to a DAD (Direct Access Device), part of the physical disc, but logically considered as a complete disc. The logical structure may vary from one DAD to another. It may consist of up to 32768 physical consecutive records of 410 characters each. All DADS on one physical disc are catalogued in the VTOC (Volume Table Of Contents) at the beginning of the disc. They are defined and assigned a file code in the range CF to CO at sysgen. These are internal file codes. CF is the system DAD. The user can assign any user file code to the sysgen-defined DADS.
- to files, belonging to a DAD. All files of one user are kept in a User Library. Files can be accessed randomly or sequentially at physical or logical record level.

At the beginning of a DAD a catalogue of user directories is created, containing entries to identify users (Userids) and the addresses of user file directories. For each Userid 1 granule is reserved to contain file entries.

The physical disc structure is therefore as follows:



### Physical Structure

The first DAD on the disc starts at cylinder 0.

The first granule contains the disc organisation characteristics:

- VOLAB (volume label):
  - sector 0 on X1215/X1216 discs
  - sector i on CDC discs, where i is the number of interlaces on the first DAD.
- IPL (Initial Program Loader):
  - sector i on X1215/X1216 discs, where i is the number of interlaces on the first DAD.
  - sector 0 on CDC discs.

The logical sector addresses (relative address in the first DAD) of these sectors are:

X1215/X1216:

Physical Address	Logical Address	Contents
0	0	VOLAB
i (interlacing)	1	IPL
	2 to 5	Catalogue of first DAD
61	6	Bad track list
71	7	VTOC

CDC:

Physical Address	Logical Address	Contents
i (interlacing)	0	VOLAB
0	1	IPL
	2 to 5	Catalogue of first DAD
	6	Bad track list
	7	VTOC

Note:- For both CDC and X1215/X1216, logical sector 0 contains the VOLAB and logical sector 1 contains the IPL.  
 - Physical access is possible only on a DAD.

Although physical sector 0 is used differently for the various disc types, 5 words in this sector must always be set as follows, to be able to initialize a disc pack when it is mounted.

Decimal Char. Address	Hex. Char. Address	Contents
74	/4A	Reserved
76	/4C	Number of sectors per track in the first DAD
78	/4E	Number of interlaces in the first DAD
80	/50	Sector size (characters) in the first DAD
82	/52	Physical sector number of the VTOC (logical address 7)

The VOLAB contains in the first 86 characters of its sector:

- volume label
- presmark date
- volume number
- characteristics of the first DAD

laid out as follows:

/0	ident. (used only for X1215/X1216)	
2	Not used	
4	~	~
6	~	~
8	L	A
A	B	E
C	L	~

16	16 characters of volume label
20	
22	D
24	T
26	
28	
2A	Day in ASCII
2C	
2E	Month in ASCII
30	
32	Year in ASCII
34	
36	
38	
3A	P
3C	C
3E	L
40	B
42	
44	
46	4 hexa ASCII characters, containing Vol. Hbr.
48	
4A	Reserved
4C	Number of sectors per track
4E	Number of interlaces in first DAB
50	Sector Size (characters) in first DAB

52	Physical Sector Nbr. of the VTOC (-log. sect. nbr.)	↑
54	Length of BITAB in char., excluding this word	
56	BITAB	

#### VTOC (Volume Table of Contents)

The VTOC contains the coordinates and characteristics of the DADs on a disc.

Each entry consists of 8 words. The entries are written in the order in which the DADs are defined on the disc. I.e. when the address of the first cylinder of a DAD is not equal to the address of the last cylinder + 1 of the previous DAD, a number of free cylinders exists between these DADs. If possible, they will be used for a following DAD allocation before

The layout of the VTOC is as follows:

0	ident. (used only for X1215/X1216)	First entry (dummy)
2	Nbr. of char. used (this and last word included)	
4	V	
6	0	
8	←	
A	0	
C	Nbr. of tracks per cylinder	
E	Nbr. of cylinders on the disc	
10	Address of first free cylinder	
12	Reserved	

14	DAD Name (6 char.)		Current entry
15	Interlace factor	Nbr. of sectors/track	
16	Sector size in characters (410)		
18	Nbr. of consecutive cylinders of the DAD		
20	Address of the first cylinder of the DAD		
22		Nbr. of sectors/granule (8)	
24			

where:

- word A is set to zero.
- word C is set at premark.
- word 10 points to the first free cylinder following the last DAD in the VTOC.
- Note: Cylinder addresses give the relative position of the cylinder from cylinder 0 of the disc.
- The interlace, word 1A, is used to compute the physical sector number as follows:
  - i = logical sector number in the track
  - physical sector number =  $i \times \text{interlace}$ , modulo number of sectors per track.

Entries in the VTOC have the same order as the related DADs on the disc.

The last VTOC entry is followed by a word containing FFFF when the sector is not full.

### First DAD Granule

The first cylinder of the first DAD is always cylinder 0 of the disc. As the first granule contains disc structure characteristics as well as DAD information, the user must re-Premark the disc when he removes the first DAD.

### Sector 0

This sector contains the BITAB of the DAD. The BITAB gives the status of all the granules in the DAD. It starts at location /54:

/0	ident. (X1215/X1216 only)
/2	Not used
/4	Not used with the DAD (Except in first DAD to contain the disc VOLAB)
/52	
/54	Number of characters of BITAB, this word excluded
/56	First word of BITAB

Each bit in the BITAB gives the status of the associated granule, 0 indicating an allocated or non-existing granule, 1 indicating a free granule.

BITAB length depends on DADsize and number of sectors per track in the DAD.

If any sector words remain unused, they are reset to zero, BITAB is created when the DAD is initialized.

### DAD Catalogue (sectors 2-5)

The catalogue of the users of the DAD is contained in sectors 2 through 5.

Each entry consists of 3 words:

0	ident. (X1215/X1216)
2	Number of used char. in sector

4	C	A
6	T	A
8	L	O
A	G	U
C	0	
E	0	
10	0	
12	0	
14	User ident. (up to 8 char., left-justified, space-padded)	
	Reserved	
	Reserved	
	Address of first record of directory	
	Reserved	

Current entry

where:

- in a deleted entry the first word of the user identification is set to zero.
- the last entry of the catalogue is followed by a word containing /FFFF.
- When a new entry is declared, the system will first check if a deleted entry can be used, before taking a free entry at the end of the catalogue.
- the number of used characters (word 2) includes the first two words of the directory and word /FFFF.
- sectors 6 and 7 are not used.
- if sector 5 is completely full, /FFFF is not written.

Remaining granules are used for the User Directory or for User Files. The status of the granules is given in the HITAB which is updated each time a file is catalogued or removed from the directory.

### User Directory

A user directory occupies one granule and contains the names and characteristics of all the catalogued files belonging to the same user. One DAD may contain several users, as many as the Catalogue allows.

For each version of a file one entry is created. This is done when the file is catalogued. When the file is deleted, the entry is removed.

The first entry of a directory contains the name of the user. The layout of a user directory is as follows:

0	Ident. (X1215/X1216)	First Entry
2	Number of char. used (incl. words 0,2)	
4		
6		
8	Userid	
10	0	Current Entry
12	0	
14	0	
16	0	
18	File Name (6 Characters)	Current Entry
20	File Type	
22	Reserved	
24	Number of Sectors (first two included)	
26	Address of first sector of file	
28	Reserved	

where:

- the first three words of an entry contain the file name, up to 6 ASCII characters long, left justified and space-padded.

- the fourth word indicates the file type, in 2 ASCII characters.
  - OB: object file
  - SC: source file
  - LM: Load module
  - UF: user data file
  - EF: extended file (DREM only).
- the sixth word is reset to zero for non-consecutive files, or it contains the number of sectors of the file (8 times the number of granules).

The last entry of the directory is followed by a word containing /FFFF, unless it is the last entry of the last sector of the directory.

Of a deleted entry the first word is reset to zero. Deleted entries may be reused for new files.

### DISC STRUCTURE without BADE

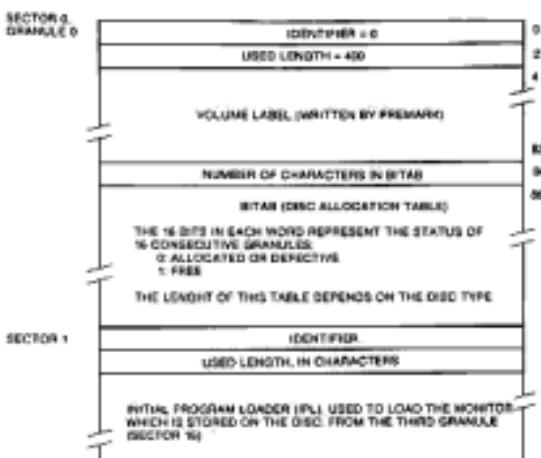
#### Catalogue and Library Structure

On a disc, the Catalogue contains one entry for each user identification declared on that disc. Each of these entries contains a pointer to a library, one for each user in the Catalogue. Each user library consists of a directory and the files in the library.

#### Catalogue Structure

The first granule on a disc contains a catalogue of the users of this disc. Its layout is as follows:

- Sector 0: Volume label and disc allocation table (see Premark: Appendix B).
- Sector 1: IPL (Initial Program Loader).
- Sectors 2 to 7: Catalogue.



The Catalogue consists of entries occupying 8 words each; each entry relates to a user who has been declared for this disc.

An entry has the following format:

USER IDENTIFICATION	RESERVED	POINTER	RESERVED
0	1	2	3

0 1 2 3 4 5 6 7

- words 0 to 3 contains the user identification
- words 4 and 5 are not used
- word 6 contains a pointer to the user directory; it is the disc sector address of the granule containing the user library directory.
- word 7 is reserved.

If the user identification is SYSTEM, the value of the pointer in word 6 is 8, because the system directory always occupies the second granule on the disc.

The Catalogue may contain up to 150 entries (6 sectors, 25 entries each).

When an entry has been deleted, the first word is filled with `0000`.

The last entry in the catalogue is followed by a word containing `FFFF`.



Format of user catalogue (sectors 2 to 7) -

#### Directory Structure

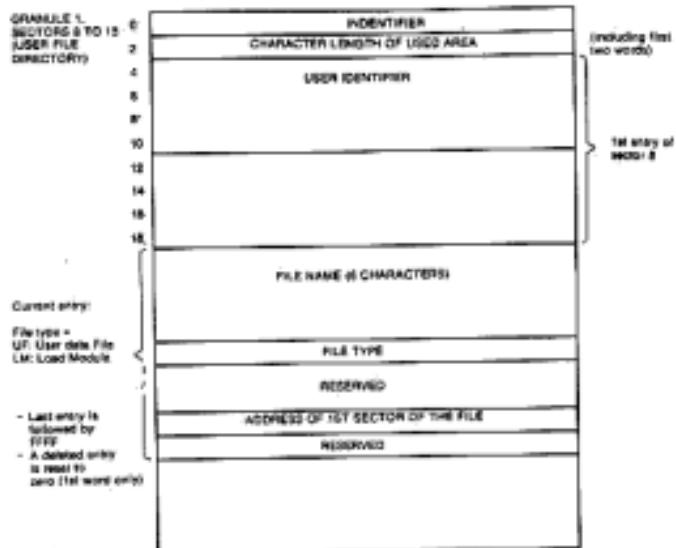
Each user is provided with his own directory and library. The directory occupies one granule and contains the names of and pointers to the user's files. The granule containing the user directory may be located anywhere on the disc, except when the user is SYSTEM. In this case it is the second granule on the disc.

Each entry in a directory consists of 8 words:

RELEASE	TYPE	RESERVED	POINTER	RESERVED
5	1	2	3	4

- words 0, 1 and 2 contain the file name
- word 3 contains the file type, which may be one of the following:

- . for load files: LM
- . for undefined files: UF
- word 6 is the file pointer; it contains the direct sector address of the first granule of the file
- words 4, 5 and 7 are reserved.



#### Format of Directory

A user directory may contain up to 200 entries (8 sectors of 25 entries each).

Each entry points to a granule table (GRANTB), containing the successive addresses of the granules allocated to the file represented by the entry in the directory. GRANTB may contain up to 200 granule addresses.

When an entry has been deleted, the first word is filled with the value /0000.

The last entry in a directory is followed by a word containing the value /FFFF.

The granule for the user directory is allocated at the time when a new user is declared to the system and entered in the Catalogue.

#### Temporary File

These files are always of the type UP. They have to have a file code assigned to them either by monitor request or by Control Command. The access method for these files is defined as follows:

- a) For random files, when a file code is assigned to the file, the required number of granules will be allocated on the disc.
- b) The access flags (in the Logical File Description Table LFT) will be reset to zero.
- c) One of these will be set when the first attempt is made to access the file, depending on the access method used:
  - If sequential access is used, any additional granules which may be required are allocated dynamically.
  - If direct access is used, the file is considered consecutive and no more additional granules will be allocated. Therefore, for this access method all the required granules must be reserved by the user program.

These temporary files may be made permanent by means of a Keep File control command.

#### Permanent Files

The names of these files are stored in a directory, of which there is one on each disc. Permanent files cannot be expanded, even if they are sequentially organized.

When a file code is assigned to a file, the system will check whether its granules are consecutive or not. If they are, random as well as sequential access may be used for this file. If they are not, only sequential access is allowed. The first attempt to access the file will set the access flag and thus define the access method for this file. This cannot be changed unless a new file code is assigned to the file. To remove a permanent file from the directory, the control command Delete File is used.

### BRIM System Disc Structure

The System Disc contains all disc resident components required by the running system. The supervisor part of the monitor is memory resident, so does not have to be stored on the disc. However, if stored on disc, it must be stored on consecutive sectors, starting from sector /12, in load module format.

The System Disc must therefore have the following structure:

- Sector 0: Volume Label
- Sector 1: Initial Program Loader (IPL)
- Sectors 2 to 5: Catalogue (only one entry is used under BRIM)
- Sectors 6 to F: Disc Directory
- Granules 3 to n consecutively contain the supervisor if it is stored on the disc.

The structure of the System Disc is therefore compatible with that of the Disc Operating System (DOS).

The following components must be resident on the System Disc:  
(BRIM Supervisor)

- D:USV1 (user monitor requests)
- D:USV2 (user monitor requests)
- D:USV3 (user monitor requests)
- D:OCOM (operator communication)
- D:DUMP (system debugging facilities: EM, WM, DD )
- D:ROOT (system command language)
- D:SEG1 (system command language)
- D:SEG2 (system command language)
- D:SEG3 (system command language)
- D:SEG4 (system command language)
- D:CI (core image file, used to contain core images of all swappable programs)

All these files, except D:CI must be catalogued before loading the monitor. The D:CI file, used to contain the core images of the swappable and read only programs, may be declared at system generation time on the system disc or on any other disc in the configuration. It will be catalogued automatically when the system is loaded for the first time, so it is not necessary for the user to catalogue it with a Keep File command.

#### FLXIBLE DISC

The flexible disc can be accessed as a physical device with the following characteristics:

- random access at sector level
- the length of a physical sector is 128 characters, all of which are available for data
- the length of a logical record may be from 1 to 512 characters. If a record comprises more than one physical sector, it is defined by the physical address of the first sector in the record. The number of sectors (1, 2, 3 or 4) is determined by the driver on analysis of the record size.
- One track contains 26 sectors of 128 characters. These sectors are not interleaved.
- 1 sector is defined by a sector number from 0 to 2001. This is the sector number given in EOB Word 5 for I/O operations.
- Bad tracks are handled automatically by the driver.

Physical and operational details about flexible disc and flexible disc drive unit are described in the P900M Operator's Guide.

All I/O operations are initiated by an I/O monitor request. At system generation time, the necessary tables for fulfilling this request must have been filled and the necessary modules loaded. When the request is given, with an LRM instruction, register A7 must have been loaded with parameters about the particular type of I/O function, while register A8 must contain the address of an Event Control Block which holds the necessary information about the data to be transferred.

There are several types of I/O request (as specified in A7):

- Random I/O requests: for random access I/O operations on disc devices.
- Basic I/O requests: for these requests the monitor will not do any character checking or data conversion, as they are used in case of binary I/O. The monitor handles only the control command initialisation and signals the end of the I/O operation.
- Standard ASCII I/O requests: these requests provide more monitor facilities, such as error control characters, data conversion from external code to internal ASCII code and vice versa, character checking for end of data. Characters are stored 8 by 8 bits, two to a word.

Moreover, a number of control functions can be performed through a monitor request, such as writing BOS or EOP records, skipping forward or backwards, rewinding, etc.

In the Event Control Block (ECB), pointed to by register A8, the user specifies the file code (see below) of the device or file concerned with the I/O operation, and additional parameters such as buffer address and buffer length. At the end of the I/O operation, the monitor places information about the result of the I/O operation in this ECB, so that it may be verified by the user program.

For non-disk devices, I/O operations are done at record level, by I/O drivers running at level 48. No blocking-deblocking is performed. For disk devices, the user can use an I/O driver or he can access the disk through the Data Management package. If he uses the driver, he must specify an absolute sector address for the I/O operation. With Data Management, he specifies the relative sector address for direct access and Data Management will find the correct sector. Moreover, Data Management automatically provides additional functions, such as blocking-deblocking.

In a following chapter detailed information will be given about the I/O monitor requests.

#### FILE CODES

Assignment is done by monitor request or control command. Some file codes are generated at system generation time, particularly those used by the monitor. The others may be assigned dynamically.

The file codes used by the DRIM system are:

02: used by the Dump routine, to output the disk or memory contents.

CF to Cx: DAD file codes used by system and defined at sysgen from CF down towards C0 for as many DADs as defined.

E0: used to read the SCL control commands from.

EP: used for operator communication.

FO to FF: used for the physical disk units.

For Device Addresses, see Appendix I.

#### ACCESS MODES

Two access modes are allowed for disk files: random and sequential.

- Random access is possible only for fixed length records of 200 words (one sector).

A record is accessed by means of the record number (file sector address).

- Sequential access is possible for records of variable length of up to 3200 characters (1600 words = 8 sectors = 1 granule).

The interface for sequential access in read or write mode is the same as for punched tape, cassette tape or magnetic tape, the blocking-deblocking function and blocking buffer allocation being handled by the system.

A file written in sequential mode can be accessed in random mode. For further details see Data Management in Chapter 7.

Data Management consists of a set of routines to help the user transfer his records between the memory and the peripheral devices, to help him create files of a particular type and retrieve records from these files. The routines are selected and included in the monitor at system generation time.

Data Management is memory resident and runs at level 49. This implies, that a request coming from a program at level 49 can be processed only when this program, or any others connected to level 49, have given a Wait monitor request.

All operations on the peripheral devices take place through file codes, so the user need not know the type and physical address of each device. The system will find this out by translating these file codes with the aid of monitor tables.

There are two types of Data Management, i.e. two ways of writing or reading files:

Sequential Access Method and Direct Access Method.

The user creates a file by assigning a file code to a temporary disc file (AS control command) and writing onto it by running a program.

All Data Management operations, i.e. reading and writing a record, writing EOS or EOP, etc., are done by I/O monitor requests in the program for each record, in which the user can specify the access method and the data management function. It is not necessary to call special routines. The mode of access is determined by the first request for a whole run.

At system generation time the user has to define the number of buffers and their lengths for use by the Data Management package. In general 2, or at most 3, buffers of one sector length will suffice. These will then be included in the system to be allocated automatically.

The first sector on each disc contains a disc allocation table (BITAB) in which the status of each granule, free or allocated, is recorded. See page 1-33.

The second sector of each file contains a granule table (GRANTS) with the addresses of all the granules of this file. See page 1-33. Two types of file exist:

- load module (LM): only direct access possible
- undefined (UF): sequential and direct access possible.

#### SEQUENTIAL ACCESS METHOD

A file is sequential when the only relation between the different records is their sequence. When such a file is created, the records must be presented in the same order in which they must be written onto the disc. To access the file, it must be scanned sequentially until the desired record is found.

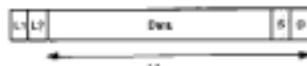
The logical sequence in a sequential file is identical to the physical sequence of the records in the file.

#### Record Structure

User records may contain up to 5200 characters. This implies that a record may be part of a sector or that it may occupy a number of sectors. When these records are written onto disc they are first blocked into a buffer.

#### Logical Record Format

In order to save disc space, the system compresses and blocks the logical records used in sequential access; trailing blanks are removed. The format of a record is as follows then:



where:

L1 is the record length, including the words S and D, but not including L1 and L2:



V = not used

S = 1, if the current record is a segment mark (EOS)

P = 1, if the current record is a file mark (EOF)

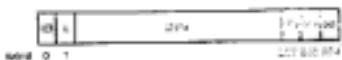
L2 = the initial record length in characters, as specified in the user's EOB (word 2) in Write mode

S = file sector address of the first word of the record.

D = the displacement in sector S of the first word belonging to the record (number of characters).

### Sector Format

The format of a sector is as follows:



where:

ID is the cylinder identification: a number from 0 to 202, used to check seek operations on moving head discs.

L is the length of area used for data in the sector (0 to 400 characters)



where:

D = 1, if the sector has been deleted from the file.

S = 1, if the current sector contains a segment mark (EOS).

F = 1, if the current sector contains a file mark (EOF).

### Special records

Some special records have the following format (compare with - Logical Record Format):

:EOS:  $\underbrace{4004}_{L1} - 0 - S - D$

:EOF:  $\underbrace{2004}_{L1} - 0 - S - 4$   
+  
D

: Blank card:  $\underbrace{0004}_{L1} - \underbrace{0050}_{L2} - S - D$   
(hexadecimal 50 = decimal 80 = card length).

### Note:

Records on disc always consist of an even number of characters. So, whatever the value of L2 given by the user at creation time, L1 always represents an even number of characters, because when the requested length is an odd value, a dummy character is added to the record.

An EOS is always stored in one sector and must be the last record in the sector.

An EOF is always written in a separate sector.

#### File Creation and Processing

A sequential disc file is created by the program delivering the logical records with the aid of the Data Management Package. Each record is written by an I/O request up to 'Write EOF'. When this request is encountered the contents of the last blocking buffer are output to the disc and an EOF record is written in the next sector.

When a request is given to write an EOS, the current sector is terminated with an EOS record and the following record will be written at the beginning of the next sector. Before creating a sequential file the user must assign a file code to a temporary disc file. If the user wants to have the file catalogued, he must give a KF control command. Before  
reading such a catalogued file or writing onto it, an assign command has to be given for it.

#### Updating

If a user wants to update a sequential file he must first read it, then update it and finally write the updated file on another temporary disc file. Such an updated file can be made permanent in the same way as the original file and under the same name, by means of the KF control command. If the  
user gives a new name to the updated file, the original file is not destroyed, which enables him to have several versions of one program belonging to the same library.

#### Read a Record

To get a logical record from an input sequential disc file, the user may give a Read monitor request (LKM 1), as for any other device. The system automatically provides the disc buffer, fills it, deblocks the records and recovers any errors. Only the sign-

ificant part of the record will be stored in the record area specified by the user in his ECB, i.e. control words will be removed by the system.

When an EOS or EOP mark is encountered, this is indicated to the user in the Status word of the ECB (word 4). An attempt to read records beyond an EOF mark will cause an EOF status to be returned to the user.

#### Write a Record

To put a record on an output file may be done by means of a Write monitor request (IMN 1), as for any other device. When the record is moved to the disc buffer it will be formatted with control words, as it consists only of data words in the user area. The system will also take care of buffer allocation, record blocking and error recovery. For temporary sequential files, records can be written onto a file until the maximum number of granules has been allocated (200). After this, an 'End of Medium' Status will be returned in the user's ECB if an attempt is made to write over the number of granules already allocated.

#### Opening a File

Opening a file need not be requested by the user as this is implicit in the first read or write request.

When the AS control command is given an entry is made in a file code table and a Logical File Table is built by the system to record information about the file used. This, however, does not imply that the file has been opened yet.

#### Closing a File

Closing a file is done in write mode, after the last record of a file has been written onto the disc, by giving a Write EOF monitor request. If the user wants to write the contents of the last buffer onto a disc without closing the file, he should give a Write EOC request. This is the case with the common object file created by the language processors, which do not have to close the /O file

as this is done by the system when the Linkage Editor is called:

#### Positioning a File

It is possible for the user only to position the file at the first logical record. This is done by giving an I/O request with the order to rewind the file.

#### Data Management Requests

The Data Management package is activated by an I/O monitor request (LKM 1). The function which has to be performed is loaded into the A7 register, while the A8 register must be loaded with the address of an Event Control Block, containing the necessary parameters. The calling sequence is as follows:

LINK	A7,00DB
LDKL	A8,ECB
LKM	
DATA 1	

where:

the word ECB is made up as follows:

		1	0	ORDER
0		1	0	10

bit 0 = 1: wait for completion of the event is implicit in the request.  
= 0: control will be returned to the calling program after initialization of the operation. To check for completion the program must give a Wait monitor request (LKM 2).

bit 9 is not used, and must be reset to zero. (If it is 1, the status /0010 will be returned).

ORDER contains the function codes:

- /01,/02: Read a Record (Basic, Standard)
- /05: Write a Record (Basic)

```

/06: Write a Record (Standard)
/07: Write a Record (Object, 4+4+4 tape format)
/08: Write a Record (Object, 8+8 tape format)
/22: Write EOF mark (Close a file)
/26: Write EOB mark
/30: Get information about a file code
/31: Rewind the file.

```

Notes:

When the order /30 is given, the user must specify the file in ECB word 0; the ASCII characters DK (disc physical files P0 to P9) or DL (disc logical files) will be returned in ECB word 1 and the word LPTMOD1 (see Logical File Table, part 2) in ECB word 4; the other words will be reset to zero, because disc file codes are handled by the system.

Basic orders (01, 05) and Object Write orders (07, 08) are converted to Standard Read/Write for disc files.

The standard ECB, to which register A8 points, has the following layout:

	0	1	2	3
ECB0	EventCharacter	FileCode		Y/X
ECB1	RecordAreaAddress		X	
ECB2	RecordAreaLength		X	
ECB3	EOFRecordLength		Y	
ECB4	Status		Y	
ECB5	NotUsed		X	

The words marked X must be filled by the user.

Those marked Y must be reserved by the user, but will be filled by the system.

ECB 0: Event Character: Bit 0 is set to 1 on completion of the I/O operation. The other bits are not used and reset to zero.

File Code: Defines the logical reference to the file.

ECB 1: Specifies the beginning address of the area where the record is stored in memory.

- ECB 2: Length in characters of the record area. (Words for basic read on cards).
- ECB 3: Number of characters which has actually been moved from or to the record area. (Words for basic read on cards).
- RCB 4: This word contains the status returned to the user program by Data Management. See page 1-110. In addition, status /IO is returned when an attempt is made to write beyond the last allocated granule of a file (End of Medium) and for temporary files, when over 200 granules are written.
- ECB 5: is not used with sequential access method.

### DIRECT ACCESS METHOD

When the direct access method is chosen, the records within a file may be organized in any manner and accessing a record may be done at random, by specifying a file sector address from 0 to 1597. This is possible because with this method a logical record is equivalent to a physical record on the disc: one sector. When a direct access file is created, the records may be delivered in any order. The system will create a granule table and allocate granules to the records (sectors) that are delivered. Each granule address is noted in the table, and when the user wants to read a particular record, he specifies the file sector address, upon which the system will be able to find it with the aid of this granule table. Thus, such a direct access file may be considered a keyed file, where the relative number of the record (=sector) is the key. Although the great advantage of a direct access file is that the user can read, write or update individual records without having to scan or copy a whole file sequentially, he may nonetheless want to be able to access such a file sequentially. If this is the case, the individual records must be formatted in the same way as for a sequential file, i.e. the sector format must be the same and the records must be written sequentially and terminated with a 'WRITE EOF' request.

### Record Structure

With direct access, a logical record is the same as a physical record: a record is equal to a sector.

The first word contains the cylinder identification (used by the disc driver to check the position of the head after a seek operation). The remaining 204 words may be used for data storage.

### File Creation and Processing

A file is created when an Assign command or monitor request is given. This reserves the required number of granules on the disc where the records can then be written. Such a file can be made

permanent by means of a Keep File control command. When a request is made, each record is transferred directly from the user area to the disc.

For DFTM, the requested number of granules is allocated straight away and extension of this number during file creation is not possible.

Random access files do not have to be closed by the user.

#### File Retrieval

Here lies the main advantage of a direct access file, because once the file has been assigned, any sector record can be retrieved, erased or updated and rewritten individually.

The size of a file is fixed at creation time, when a granule table is also written with an entry for each granule of the file. When the file is catalogued by means of a XF command, creation is terminated and no more granules can be added. Therefore the user must know the maximum size of the file at creation time (no more than 1598 sectors).

#### Read a Sector

This is done in the same way as for sequential access, the only difference being that the user must specify in ECB5 the relative number of the sector within the file (i.e. the file sector address, a number from 0 to 1597), and specify / A for the read order in register A7. Moreover, he must supply the system with a 205-word buffer in which the physical record will be stored. As mentioned above, the first word contains the cylinder identification.

#### Write a Sector

This is done in the same way as for sequential files, the only difference being that the user must specify in ECB5 the relative number of the sector to be written into the file (a number from 0 to 1597) and specify / B for the write order in register A7.

Moreover, he must supply the disc buffer in which the information is stored: a 205-word buffer of which the first word will be replaced by the cylinder identification by the system (required by the physical disc I/O driver).

#### Data Management Requests

The Data Management package is activated by an I/O monitor request (LKM 1). The function which has to be performed is loaded into the A7 register, while the A8 register must be loaded with the address of an Event Control Block, containing the necessary parameters. The calling sequence is as follows:

LKD	A7, CODE
LDEKL	A8, ECB
LBN	
DATA 1	

where:

the word CODE is made up as follows:



bit 8 = 1: wait for completion of the I/O operation is implicit in the request.

- 0: control will be returned to the calling program after initialization of the I/O operation. To check for completion, the program must give a Wait monitor request (LMR 2).

bit 9 is not used and must be reset to zero. (If it is 1, the status /0010 will be returned).

ORDER contains the function codes:

/0A: Read (Random)

/0B: Write (Random)

/30: Get information about a file code.

Note: When the order /30 is given, the user must specify the file code in ECB word 0; the ASCII characters 'BB' will be returned in ECB word 1 and the word LPNMD01 (see Part 2, Logical File Table) in ECB word 4. The other words will be reset to zero, because disc file codes are handled by the system.

The standard ECB, pointed to by register A8, has the following layout:

	0	1	2	3	4	5
ECB0	Event Character	File Code			X	
ECB1	Disc Buffer Address			X		
ECB2	Requested Length			X		
ECB3	Effective Length				X	
ECB4	Status				X	
ECB5	Relative Sector Number				X	

The words marked X must be filled by the user.

Those marked Y must be reserved by the user, but will be filled by the system.

ECB0: Event Character: Bit 0 is set to 1 on completion of the I/O operation. The other bits are not used and reset to zero.  
File Code: Defines the logical reference to the file.  
ECB1: Specifies the beginning address of the 205-word disc buffer.  
ECB2: Whatever the value of the requested length, 205 words will be written on the disc or read into memory.  
ECB3: Remains unaffected.  
ECB4: This word contains the status returned to the user program by Data Management when the sector transfer is terminated. See page 1-110. In addition, status /10 is returned when an attempt is made to write beyond the last allocated granule of a catalogued file and, for catalogued files, when over 200 granules are written.  
ECB5: Specifies the relative position of the sector within the file, i.e. the file sector address.

#### Physical Disc Access

This is a direct access on a physical sector level and can only be used on DADs. The Data Management module is not used for this type of access, but special orders are used in the LDM 1 monitor request:

/11=physical read  
/15=physical write

Moreover, the file code in the ECB must be one of the disc unit file codes /70 to /FF.

Access is done at sector level, where the sectors are numbered consecutively from 0 to n. This implies that word 5 in the ECB must contain the disc sector logical address of the sector which is to be accessed. The disc which is accessed must not be shared by the user and Data Management, but it may be shared between a number of other programs doing direct physical access. An example of this type of access is 'Dump Disk'.

The initial loading procedure is very simple: The bootstrap is loaded, either through the toggle switches or by pushing the IPL button on the control panel, then the Initial Program Loader (IPL) is loaded into memory, followed by the monitor.

#### LOADING BOOTSTRAP AND IPL

The bootstrap can be loaded in one of two ways, depending on whether the optional ROM bootstrap is included in the system or not:

If not, the procedure is as follows:

- switch on the CPU
- load the bootstrap into the first 64 memory locations manually, by means of the toggle switches and the Load Memory button on the control panel. The 64 bootstrap values can be found in Appendix E. Then check by reading these locations out.
- set up the device parameters on the toggle switches, as shown below, and load this value into the A15 register.
- put the disc containing IPL and monitor into the disc drive, push the START button on the drive and wait till the READY button lights
- push the MC button
- load 0 into the A0 register
- push the RUN button on the CPU control panel
- the IPL is now loaded into memory and it loads, in turn, the monitor, after which the monitor initialization phase is started with the typing out of the monitor identification.

If the ROM bootstrap is included in the CPU, which is highly recommended, the procedure is much simpler:

- switch on the CPU
- put the disc containing IPL and monitor into the disc drive, push the START button on the disc unit and wait for the READY button to light

- set up the device parameters on the toggle switches on the CPU, as shown below, and load this value into the A15 register
- push the IPL button on the control panel. This loads the bootstrap into memory, which immediately loads the IPL from disc. The IPL then loads the monitor and the monitor initialization phase is started with the typing out of the monitor identification.

The device parameters on the data switches must be set as follows:

0	1	2	3	4	7	8	9	10	15
---	---	---	---	---	---	---	---	----	----

where:

bit 0 = 0: character exchange on Programmed Channel  
 - 1: word exchange on Programmed Channel

bit 1 = 0: IPL not loaded from disc or loaded from CDC disc  
 - 1: IPL loaded from disc

bit 2 is used only if bit 1 = 1:

- 0: fixed head disc or flexible disc  
 - 1: moving head disc (X1215 or X1216)

bit 3 = 0: IPL input device connected to I/O Processor

- 1: IPL input device connected to Programmed Channel

bits 4 to 7 contain control information for the control unit:

TY = 0001 HE = 0010 PL = 0000  
 XK = 0111 PR = 0000 CDC = 0001 X1215/16 = 0011

bit 8 = 0: a single device control unit is involved

- 1: a multiple device control unit is involved

bit 9 = 1: X1215 or X1216 used for IPL

- 0: other device or disc type used

bits 10 to 15 contain the device address (See appendix on device addresses)

### Initial Program Loader (IPL)

The disc IPL program is written onto the disc when the disc is pre-marked. It is written in absolute binary, so, to enable it to run anywhere in memory it does not contain any memory direct reference. When loaded, the IPL reads and loads into memory from the disc from which it has itself been loaded, starting from sector number /12 (the first two sectors of a file are reserved for the system). The first four words of sector /12 contain:

- start address of the load module
- number of sectors used by this module  
(this is the standard load format on disc; these words are generated by the disc linkage editor).

#### Note:

As long as it has been stored on the disc according to the IPL requirements, any stand-alone program, even if it does not use the disc, can be loaded into memory by the disc IPL.

#### Programs to be loaded by disc IPL

- must be in disc system load format (106 code words plus relocation per sector)
- must be catalogued on disc as a file starting at disc sector logical address /10
- must be built of consecutive granules.

## STARTING THE SYSTEM

After the monitor has been loaded from an input device, the system will output a question on the typewriter:

PARTITIONING?

The answer to this question must be Y or N.

If it is N, the sizes of the different memory partitions will remain as defined at system generation time.

If it is Y, the user is given the option of modifying these sizes. The following messages are output for this purpose:

### DYN AREA LENGTH:

The user may type in 4 hexadecimal characters to define the length, in characters, of the Dynamic Allocation Area.

### READ ONLY LENGTH:

The user may type in 4 hexadecimal characters to define the length of the Read Only Area. This must be at least /680 words.

### PROGRAM SAVE AREA:

The user may type in 4 hexadecimal characters to define the length of the Program Save Area.

### CORE RESIDENT AREA LENGTH:

The user may type in 4 hexadecimal characters to define the length of the Memory Resident Area.

### SWAP AREA LENGTH:

The user may type in 4 hexadecimal characters to define the length of the swap area.

Any remaining memory space will be used for background programs.

Bits 10 to 15 of the device parameter word, initially set up on the control panel data switches (see page 1-64), give the address of the system disc. If this is different from the one defined at system generation, file code /F0 will be assigned to this disc.

For example, at system generation time file code F0 was assigned to disc 02, and file code F1 to disc 12. If the contents of bits 10 - 15 = 02, this will remain so; if they are 12, file code F0 will be assigned to disc 12 and F1 to disc 02.

When the monitor has been loaded control is given to the INIMON module which takes the following actions:

- Checks the unit from which the monitor was loaded and possibly exchanges file code /F0 with that of the disc from which the monitor was loaded.
- Checks the operability of the disc units declared at SYSGEN. At this point the messages

DISK or  
DISC UNIT <dev. addr.> UNKNOWN

may be output in which case these units are flagged to be inoperable so that only a ready interrupt can allow their use. Fixed head discs must be ready before loading the system, for they do not generate ready interrupts.

- Reinitializes the memory layout parameters from the keyboard, if Y was answered to PARTITIONING? . determines the memory size and formats the memory. At this point the message

TRY MORE CORE

may be output to indicate that the memory is too small for the specified parameters.

- Determines the background area size.
- Sets the PCT pointers of the read only programs on their disc addresses (via the directory), reserves the save areas for the read only programs and activates the D:USV3 system program which creates the D:CI file and generates all system read only programs in core image format.

At this point one of the following messages may be output:

D:CI TOO BIG

(not enough consecutive grammes can be found on disc for the D:CI file)

#### D:CI TOO SMALL

(not enough room in this file for the system read only programs, or Read Only Area too small for the system read only programs, or read only program missing on disc).

At the end of this initialization by INIMON the message

DRTM <release number>

is output and a branch is made to the dispatcher.

Note: When a new supervisor is implemented (with an RRU command) with a D:CI file size declared during sysgen different from the size of the old one, it is strongly recommended to delete the old D:CI file before the loading phase, because the system will use this file instead of allocating granules to the new D:CI file.

#### System Initialization

When the monitor has been loaded by the Initial Program Loader (IPL), the user must load all memory resident programs, through LD control commands, and declare all Read Only programs, through RO or RW control commands. Then these programs must be connected to a hardware or software level and the system is ready to run. By means of external interrupts or by an ST control command various activities may be started.

#### Mounting a New Disc

When a system is running, the user may mount a new disc. A 'Ready' interrupt is sent to the system when the disc becomes operational, with the following distinctions for the different disc types:

- a) If the interrupt comes from the system disc, nothing happens.
- b) If it comes from the fixed disc in an X1215, there are two possibilities:
  - In case the disc has already been used, i.e. at least one file code has been assigned to a catalogued or temporary file on this unit, nothing happens.
  - In case the disc has not been used, i.e. no file code has yet been assigned to it, the system will read the volume label of the disc, print it on the typewriter and initialize the disc allocation table.

- c) If the interrupt comes from a moving head disc or from the removable disc of the X1215, all file codes assigned to a disc on it are deleted, the volume label is typed out and the disc re-allocation table reinitialized.
- d) In case the disc volume label must be printed and there is no memory space available for the disc initialization program, the label will not be printed. The disc must then be mounted again.

## SYSTEM MESSAGES

The following messages may be output by the monitor:

- during the initialization phase (see 1-66):

PARTITIONING?

BYN AREA LENGTH:

READ ONLY LENGTH:

PROGRAM SAVE AREA:

CODE RESIDENT AREA LENGTH:

DISK, if a disc cannot be initialised (see below).

DISC <address> UNKNOWN, if the disc specified by <address> is unknown by the CPU (wiring-error or non-existent).

When the system is ready to be started, the following message will be printed on the typewriter:

\* \* DRTM \* \* X X \* \*

Where X X indicates the release number.

The operator may now press the control panel interrupt button and start the system.

- while the system is running:

ER if an operator message is not recognized by the system, or cannot be executed.

PU,<device name> <device address>,<status>,RY

An error or abnormal condition has occurred on the device with address specified. The hardware status is printed.

If RY is typed as well, the operator may attempt to correct the error and retry the last I/O operation by typing the operator message RY.

DEER-<disc address>,<sector number>,<status>

This message indicates that an irrecoverable error has occurred on a disc:

<disc address> is the physical address of the disc unit on which the error has occurred.

<sector number> is the address of the sector on which the current operation should be performed. It is specified as a 16-bit value:

- for fixed head disc, it gives the absolute sector address;

- for the X1275 discs:

- bits 1 to 10 give the cylinder number

- bit 11 the head number

- bits 12 to 15 the sector number where the error occurred.

#### DISK INIT ERR

When a disc unit is switched on and the monitor cannot read its volume label, it will print this message on the typewriter to tell the operator that this disc cannot be used. He will have to mount another disc or try and fix the current one.

#### Fatal Errors

There will be no message output in case of a fatal error, but the system will execute a loop instruction (EB M) and put a code in the A1 register to indicate the cause of the fatal error:

- 0: unknown interrupt
- 1: power failure (if no routine is available)
  
- 3: stack overflow
- 4: non-wired instruction
- 5: scheduled label for a level equal to or below 48
- 6: overflow on T:EVT tables (event count; see Part 2)
- 7: overflow on the scheduled label table
- 8: overflow on the dynamic allocation area
- 9: request to M:INML (dynamic allocation) to release an unknown block
- A: exit from a level equal to or below 48
- B: overflow on the program save area.

Note: The address of the EB M instruction can be found in the SYBLINK MAP output (see System Generation): it is the address of location B:EREG+2, inside the module named HALT.

SYSTEM COMMAND LANGUAGE (SCL)

The DRTM has not been designed to handle batch processing. Therefore, the System Command Language must not be considered as a job control language, but more as a system initialization program. It allows the operator to load or declare various programs written independently from the monitor.

The System Command Language module is activated when the operator types in CC after having pressed the INT button on the control panel. It consists of read only programs usually connected to level 61 and can therefore interrupt the execution of lower priority user programs. This is not much of a problem, because:

- it performs many I/O operations, often on typewriter (a slow device) on a conversational basis, so that it does not consume a great deal of processor time;
- it is used mainly to initialize the system, not to supervise processing.

Each command consists of a mnemonic of two characters, followed by a space and parameters, if any.

Parameters are positional and separated by commas.

The syntax description for the commands where DAD or disc specification is required, is as follows:

```
<daa> ::= <dfc>[<din>]<dadname>
<dfc> ::= <DAD file code>(user assigned)
<din> ::= <disc file code>(/r0 - /rr)
```

DADname may be omitted, in which case the first DAD on the disc will be taken.

When input via the typewriter, a command must be terminated by CR LF. System commands are always read from a device with system file code /E0, which is normally assigned to the typewriter. It can easily be assigned to another device.

Numerical characters used in the command parameters can be specified in hexadecimal format (preceded by a slash) or simply in decimal.

When an error occurs, the command cannot be processed and an error code will be output on the typewriter through file code EF:

ER 00: command unknown  
ER 01: syntax error  
ER 02: disc not operational  
ER 03: file code unknown or: not compatible with the requested operation  
ER 11: I/O error  
ER 15: parameter error

Other error codes are specific to each command and are therefore described with the commands in the following paragraphs.

TABLE OF SCL CONTROL COMMANDS

Command	Meaning	Page
AS	Assign a file code	1-96
BF	Space file backwards	1-99
BR	Space record backwards	1-99
CN	Connect a program to a software level	1-91
CT	Connect a program to a timer	1-92
DF	Delete a file	1-98
DL	Delete a file code	1-97
DN	Disconnect a program from a level	1-93
DT	Disconnect a program from a timer	1-93
EN	End of commands	1-99
FF	Space file forward	1-98
FR	Space record forward	1-98
FP	Keep file	1-95
LD	Load a memory resident program	1-88
Magnetic Tape Control Commands		1-98
RO	Declare a read only program	1-89
RW	Rewind tape	1-98
SC	Set clock	1-94
SD	Set date	1-94
ST	Start a program	1-94
SW	Declare a swappable program	1-90
TS	Define time slice	1-91
UN	Unload tape	1-98
WF	Write End-Of-File mark	1-99
WS	Write End-Of-Segment mark	1-99
WT	Write End-Of-Volume mark	1-99

## LOAD A MEMORY RESIDENT PROGRAM

### Syntax

LD,<name>,<das>[<level>][SL,<number>]

### Use

This command is used to load a program into memory from disc.

<name> is the name of the program which must be loaded. It consists of a character string of up to 6 characters.

<das> : see page 1-85

<level> must be specified if the program is an interrupt routine. If <level> is not specified, the program to be loaded is not an interrupt routine. In this case a 16-word save area will be reserved in front of the program.

SL,<number> indicates that the program uses scheduled labels; <number> is the maximum number of labels to be scheduled at the same time. A scheduled label save area is also reserved in front of the program.

Note: The granules of a load module need not be consecutive on the disc. This command must be used at initialization time, before starting the memory resident program.

### Error Codes

One of the following error messages may be output for this command:

ER 04: no Program Control Table available

ER 06: memory resident area overflow

ER 07: level error

ER 08: level already connected

ER 09: program unknown

ER 10: too many scheduled labels

ER 12: program already declared

ER14 : DADname unknown

ER 21: DAD not found

LDER <program name>: I/O error during loading operation.

## DECLARE A READ ONLY PROGRAM

### Syntax

ROw <name>,<das>[,SL,<number>]

### Use

Before starting any program activity, the user must declare all the programs which are used in the system at execution time, by means of this command. The monitor is then able to build the Program Control Table used by this program.

<name> : is the name of the read only program (up to 6 ASCII characters).

<das> : see page 1-85.

SL,<number> indicates that the program uses scheduled labels and specifies the number of scheduled labels for which a save area must be reserved.

Note: All read only programs must be declared before use, otherwise they will be considered as background programs.

The system will make a core image copy of the program and store it in the B:CI file. This file does not have to be catalogued by the user, for it will be created implicitly by the system when the monitor is loaded for the first time.

### Error Codes

One of the following error messages may be output for this command:

ER 04: no Program Control Table available

ER 05: read only save area overflow

ER 09: program unknown

ER 10: too many scheduled labels

ER 12: program already declared previously

ER 13: program too long (overflow into read only save area).

ER 14: DADname unknown

ER 21: DAD not found

ER 26: B:CI file overflow.

ER 27: A read-only program cannot be segmented.

## DECLARE A SWAPPABLE PROGRAM

### Syntax

SL, <name>,<das>[<time slice>] S,[I|E] /,SL,<number>]

### Use

The user must declare all swappable programs before they are activated, otherwise they will be considered as background programs. By means of this command a program is declared as swappable, causing the monitor to build an entry for it in the Program Control Table, allocate its save area and produce the core image of the swappable program in the DCI file.

<name> is the name of the program.

<das> : see page 1-85.

SL,<number> defines the number of scheduled labels for which space must be reserved in this program's save area.

<time slice> is the value of the time slice which must be used for this program. It must be a multiple of 100 milliseconds.

If S is specified instead, the default value for the time slice, as defined at System Generation time, must be used.

If I is specified, the program can be swapped immediately when its time slice has elapsed, regardless of the number of I/O operations taking place on non-disc devices.

If E is specified, the program will be swapped only after all current I/O operations have been terminated.

If I is specified, the buffer and the ECB required for the I/O operations for non-disc devices must be outside the swap area, i.e. in the memory-resident area or in the dynamic buffer area. When E is specified, there is no such restriction.

### Error Codes

ER01: syntax error

ER04: no free Program Control Table available

ER05: Program Save Area overflow

ER11: I/O error

ER12: the program has already been declared

ER15: parameter error

ER 14: DADname unknown  
ER 24: DAD not found  
ER26: overflow on the D:CI file.  
ER27: a swappable program cannot be segmented

#### DEFINE TIME SLICE

##### Syntax

TS<number>

##### Use

This command is used to define the time slice used for the swappable program, if it has not already been specified in the SW command. If the time slice has already been defined at System Generation time, it may be redefined by means of this command. *<number>* is a value, specifying the length of the time slice in tenths of seconds, with a maximum of 256.

#### CONNECT A PROGRAM TO A SOFTWARE LEVEL

##### Syntax

CK,<name>,<level>

##### Use

By means of this command a program, which has already been declared to the monitor as either a memory resident or read only program, can be connected to a software level. Interrupt routines cannot be connected to a priority level with this command, as no Program Control Table is allocated for them.

It is also possible to connect a program to a software level later on, by means of a monitor request.

*<name>* is the name of the program (up to 6 ASCII characters).

*<level>* is the software level to which the program must be connected.

**Note:** background programs need not be connected explicitly when they are activated, because they don't have to be declared to the system at initialization time.

However, at syntax generation time, the user must reserve enough memory space to handle the Program Control Tables of all background programs used. These tables are allocated and released dynamically for the background programs.

#### Error Codes

ER07: <level> error  
ER08: program already connected  
ER09: program unknown.

#### CONNECT A PROGRAM TO THE CLOCK OR A TIMER

##### Syntax

CTw<name>,<NTIM>,<PR>,[<HC>][<HH>,<MM>,<SS>]

##### Use

When a program has been connected to a software level, it may, through this command, also be connected to the real time clock or to a timer. This may also be done by monitor request.

For timer numbering, see part 2, page 2-51.

<name> is the name of the program which must be connected (up to 6 ASCII characters).

<NTIM> is the timer number. If the program must be connected to the real time clock, 0 must be specified. For timer numbering, see under monitor request LRM10.

<PR> gives the pulse rate, a number from 0 to 127. When 0 is specified, only one program activation is made, after which it is automatically disconnected from the clock or timer.

<HC> indicates the number of timer cycles before the first program activation (a hexadecimal value from 0 to 1FFF).

<HH>,<MM>,<SS> gives the time in hours, minutes and seconds when the program must be activated.

Error Codes

ER16: the specified timer has not been defined.  
ER14: the program has not been connected to a level.

DISCONNECT A PROGRAM FROM A LEVEL

Syntax

DE,<name>,<level>

Use

This message is used to disconnect a program from a software level.

Error Code

ER01: syntax error  
ER07: level error.  
ER20: disconnection impossible: program busy

DISCONNECT A PROGRAM FROM A TIMER

Syntax

DE,<name>,<NTIM>

Use

This command is used to disconnect the program specified with its <name> from the timer specified to which it has been connected.

Error Code

ER16: the program had not been connected to a timer or wrong timer number.

### SET DATE

#### Syntax

SD,<DD>,<MM>,<YY>

#### Use

This command is used to set the current date in the machine. <DD>,<MM>,<YY> denote day, month and year, 2 characters each. They will be stored in memory in ASCII format and returned to user programs issuing the monitor request Get Time. The system does not check the validity of the date.

### SET CLOCK

#### Syntax

SC,<HH>[,<MM>[,<SS>]]]

#### Use

This message is used to initialize the system clock. The current time is entered in a specific internal system table and the real time clock is started. <HH>,<MM>,<SS> specify the time in hours, minutes and seconds, two characters for each. Default value is 0.

### START A PROGRAM

#### Syntax

ST,<name>[,<data>]

#### Use

This command can be used for starting memory resident, read only, swappable or background programs.

`<dns>` : see page 1-85.

`When <name>` identifies a memory resident, swappable or read only program declared previously, the program must have been connected to a software level.

Note: A segmented program may run as a core resident or as a background program.

#### Error Code

ER 04: POF pool overflow  
ER 05: Save Area pool overflow  
ER 14: program has not been connected  
ER 18: program does not exist on disc  
ER 19: overflow on dynamic area (no possibility to build an activate block to activate background)  
ER 20: program too large for background area.  
ER 21: B&D not found  
ER 51: disc I/O error.

#### KREP FILE

##### Syntax

`KREP<file code>,< file name>`

##### Dec

This command is used to make a temporary file permanent by storing it in the library of the disc on which it is located.

`<file code>` identifies the file which must be catalogued.

`<file name>` is a string of up to 6 characters to identify the file which must be catalogued. The type of the file is implicitly UP (user data file). If `<file name>` already exists in the directory, this one replaces the old one, but the granules are not recovered.

##### Error Codes

ER22: non-disc file

ER23: file has already been catalogued

ER24: no entry available in the library directory.

## ASSIGN A FILE CODE

### Syntax

```
AS<fc1>[<fc2>|<dnna>| DK<das>,<n>| DK<das>,<file name>
| EP<das>,<file name>|<das>]
```

### Use

This message is used to assign or re-assign a file code either to a physical device or to a disc file.

<file code 1>: file code which must be assigned.

<file code 2>: file code to which <file code 1> must be assigned.

DN[DA] is the device name (2 characters) and device address (2 hexadecimal digits) of the device to which <file code 1> must be assigned.

<fc1> is assigned to:

- another file: <fc2>
- a peripheral device: <dnna>
- a temporary file on disc: DK<das>,<n>
- a permanent file on disc: DK<das>,<file name>
- an extended file on a disc: EP<das>,<file name>
- a disc or DAD: <das>

When <n> is specified, the system will allocate a number of consecutive granules to the file. Such a file is temporary and must be used with random access.

<das> : see page 1-85.

Note: the system file codes must all be assigned at system time, because when an AS command is given the previous assignment is deleted before the new one is assigned. If an error occurs at that point, a system deadlock may occur.

Note: When this command is used for re-assigning a file code, the monitor will delete the old file codes, before assigning the new one. Therefore, if an error occurs, the file code remains unassigned. For some system file codes this may cause system deadlock, so the user must not re-assign system file codes, but declare them all at system generation time.

Also, as it is useful to change the assignment of /30, SCL assigns this file code to /3F when it finds it unassigned. The following device names are supported by the system:

- MT: magnetic tape
- TK: cassette tape
- CR: card reader
- PR: punched tape reader
- PP: tape punch
- TY: operator's typewriter
- TR: ASR punched tape reader
- TP: ASR tape punch
- LP: line printer
- NO: no device.

#### Error Codes

- ER 14: DABname unknown
- ER51: I/O error on disc
- ER52: no spare entry available in file code table
- ER53: no disc file description table free
- ER54: device unknown or disc file code unknown
- ER55: disc overflow or too many granules requested
- ER56: file unknown
- ER57: <file code 2> unknown
- ER58: more than 7 file codes assigned to the same disc file.

#### DELETE A FILE CODE

##### Syntax

DL<file code>

##### Use

This message is used to delete a previously assigned file code. <file code> is the file code which must be deleted. If it was a file code assigned to a temporary disc area, all granules allocated to this file are released for use by other files.

Note: In case of disc files, the file description table is released. If the file had not been closed, however, the current contents of the buffer are not written on the disc, but they are lost and the space occupied by the blocking buffer is not released.

#### DELETE A FILE

##### Syntax

DEL <das>,<file name>

##### Use

This command is used to delete a catalogued file on a disc. <file name> is the name (up to 6 characters) of the file which must be deleted.

<das> : see page 1-85.

The granules occupied by this file are released, but they are not used before the disc allocation table has been loaded again, i.e. at system reloading time for the System Disc and when the disc unit is restarted for user discs. This is done because there may be several file codes assigned to the deleted file.

##### Error Code

ER21: unknown <file name>

#### MAGNETIC AND CASSETTE TAPE CONTROL FUNCTIONS

A number of commands can be used only for magnetic or cassette tape, to perform a number of control functions:

- RW<file code> is used to rewind the tape of which the <file code> is specified.
- UN<file code> is used to unload the tape specified.
- FF<file code>[<number>]ALL is used to space the specified tape forward over a <number> of files or to skip all

files (ALL), in which case the tape will stop as soon as two consecutive tape marks are encountered.

- **RE<sub>n</sub>****<file code>[<number>]** is used to space the specified tape backwards one file or, if specified, a <number> of files. The file is positioned after the tape mark.
- **FR<sub>n</sub>****<file code>[<number>]** is used to skip forward in the defined file over 1 or, if specified, a <number> of records.
- **RE<sub>n</sub>****<file code>[<number>]** is used to skip backwards in the defined file over 1 or, if specified, a <number> of records.
- **WE<sub>n</sub>****<file code>** is used to write an end-of-file mark after the specified file.
- **WE<sub>n</sub>****<file code>** is used to write an end-of-segment mark.
- **WE<sub>n</sub>****<file code>** is used to write an end-of-volume mark.

Error Codes

ER01: syntax error  
ER11: I/O error.

END OF COMMANDS

Syntax

EN

Use

This command is used to indicate the end of a series of system commands. The SQL package will perform an exit and thus release the dynamic area.

By means of the operator commands the user may have limited control over the running system. This is initiated by pressing the interrupt button (INT) on the control panel, to start the operator control package. As soon as this module becomes the highest priority active program in the system, it will output the message

M:

on the operator's typewriter to request input from the operator. Any of the commands described below may then be typed in.

All commands are terminated by LF CR.

The characters ↑ and ← may be used to correct the typed-in message, if necessary.

Message	Meaning	Page
OC	Request SCL	1-102
CR	Correction	1-102
DD	Dump Disc	1-102
DM	Dump Memory	1-102
HD	Halt Dump	1-103
ST	Stop CPU	1-103
RD	Release Device	1-103
RY	Retry I/O Operation	1-103
WM	Write Memory	1-104

Request SCL:

CC

This command is given to activate the SCL (System Command Language) program, so that commands may be given to initialize the system or give control to the running programs. Upon this command, SCL will type out C: and wait for the operator to type in an SCL command.

Correction:

CH<file code>

By means of this command correction records can be entered. <file code> is the file code of the device from which the corrections are input.

The correction records must have the following format (up to 72 characters each):

<address>,<value 1>[,<value 2>.....,<value n>]

where <address> is the address of the first (or only) location to be modified and <value 1> specify the values which must be stored in <address> and the locations following it.

Dump Disc:

DD<disc>,<sector1>[,<sector2>]

As a result of this command, a hexadecimal dump will be made of the contents of the disc specified by <disc> (see page 1-85).

The dump will be made of <sector 1> or, if specified, from <sector 1> to <sector 2> inclusive. The dump will be output on the device with file code 02.

Dump Memory:

DM<address 1>[,<address 2>]

A hexadecimal memory dump will be made of <address 1> or of the memory locations from <address 1> to <address 2> inclusive, if specified. The dump will be output on the device with file code 02.

Halt Dump:

ED

This command is used to stop a current dump operation, which was initiated by a DD or DM command.

Stop CPU

ET

This command is used to stop all CPU activity. However, all current physical I/O operations are terminated (without checking). Then the CPU will enter the idle loop:

INR  
RR  $\times$ 2

after which the operator can switch off the system.

This command is not normally used, but in some situations it might be useful: if, for example, a program continues writing onto a magnetic device for some reason, and the operator wants to stop the machine, he may do so with the ET command and still allow the current write operation to be terminated.

Retry/Release and I/O Operation:

[RT|RD]<device address>

When an I/O error occurs which requires operator intervention, the system prints out the message:

PU<device name><device address>,<status>, RT

where RT denotes that the operator may retry the I/O operation. If the error can be corrected, the operator must give the RT command to allow the program to continue. Otherwise he must type in the command RD to release the I/O operation from the device specified. In both cases, the system will give a C10 Start Instruction to retry the last erroneous operation.

Write Memory:

W\$<address>,<value 1>[,<value 2>,...,<value n>]

This command is used to modify one or more memory locations.   
<address> is the first location to be modified (an even address, of up to 4 hexadecimal characters).   
<value 1> to <value n> are the values in hexadecimal to be entered in the memory locations, starting from <address>, i.e.

<value 1> is stored in location <address>

<value 2> is stored in location <address>+2, etc.

The user program can request the monitor to perform certain functions. A request takes the form of an LHM (Link to Monitor) instruction followed by a DATA directive.

The directive has a number as operand, which specifies the function to be executed. If this number is negative, the user is scheduling a label on completion of the request.

Preceding a request, certain parameters may need to be loaded into the A7 and A6 registers.

After the monitor has processed the request it loads a return code in the A7 register. If the requested service module is not available, A7 will always contain the value -1.

Note: It is possible to use scheduled labels in conjunction with a monitor request. See chapter 4.

Under the BRTM, some of the modules handling monitor requests are core resident, to provide fast response. Others are stored on disc just as other user-written read only programs. Still, these user programs can use these monitor requests, because any work areas or parameters are built in the dynamic allocation area, to provide the possibility of communication.

However, the read only monitor requests cannot be used by programs connected to a priority level which is equal to or higher than that of the read only monitor request, i.e. if the level of the request handler (read only) is 49, only those user programs from level 50 to 62 can use it.

Note: No monitor requests except Activate and Set Event may be issued by a program with priority level smaller than 49.

Request	LEM	Page
Input/Output	1	1-107
Wait for an Event	2	1-112
Exit	3	1-114
Get Buffer	4	1-115
Release Buffer	5	1-117
Connect Program to Timer	10	1-118
Disconnect Program from Timer	11	1-120
Activate a Program	12	1-121
Switch Inside a Software Level	13	1-123
Attach Device to Program	14	1-124
Detach Device from Program	15	1-126
Get Time	17	1-127
Set an Event	18	1-129
Connect Program to Software Level	20	1-130
Disconnect Program from Level	21	1-131
Wait for a Given Time	22	1-132
Assign a File Code	23	1-134
Delete a File Code	24	1-137
Read Unsolicited Operator Message	25	1-138
Cancel Unsolicited Message	26	1-140

## I/O REQUEST

### Calling Sequence

LBK	A7,CCOB
LINKL	A8,ECBADR
LEM	
DATA 1	

### Use

The user can ask the system to start a particular I/O operation on a peripheral device.

Processed at level 48 for physical I/O requests, at level 49 for logical I/O requests (Data Management).

Register A7 is loaded with a CCOB specifying the details of the I/O function, as follows:

bit	7	8	9	10	15	
	S	W	R		ORDER	A7

S, W and R specify the mode of operation:

S = 1: (W must also be 1): used by a swappable program; this program can then be swapped out immediately when its swap event count value becomes zero.

W = 1: the requesting program wants to wait for the completion of the requested I/O operation. Only after completion of the requested function, will the return to the calling program take place.

W = 0: a return to the calling program will be made as soon as the transfer has been initiated. The program will give a Wait request later on for synchronisation.

R = 1: the program itself will process any abnormal condition concerning the requested operation (possible only with Basic Read/Write). The system will return the hardware status in ECB word 4. No retry is possible.

R = 0: any abnormal conditions will be processed by the system. The software status is returned in ECB word 4.

ORDER specifies which I/O function is required, by giving one of the following hexadecimal values:

01: Basic Read (These orders are used by Disc File Management and can be used by user programs only if DFM has not been selected at sysgen.)  
05: Basic Write (user programs only if DFM has not been selected at sysgen.)  
For Basic I/O requests the system does not provide for character checking or data conversion, only for control command initialization and end of operation signals.

02: Standard Read  
This order can be used to input standard object code records in 4x4 or 8x8 format, as well as ASCII character strings.

06: Standard Write  
Standard (ASCII) I/O requests provide, by means of standard conversions, for special features such as error control characters, conversion from external code to internal ASCII and vice versa.

07: Object Write (4x4+4x4 tape format i.e. 4 rows in 4 columns)  
08: Object Write (8x8 tape format i.e. 2 rows in 8 columns)  
Object I/O requests provide, by means of standard conversions, for special features such as error control characters, checksum and data conversion from external 4x4+4x4 or 8x8 tape format to internal 16-bit format.

0A: Random Read  
0B: Random Write  
11: Read Sector (flexible disc; DAD)  
14: Skip forward to EOF mark  
15: Write Sector (flexible disc; DAD)  
16: Skip forward to EOF mark  
22: Write EOF mark (tape and disc sequential files)  
24: Write EOF mark (tape and disc sequential files)  
26: Write EOS mark  
2D: Door Unlock (flexible disc)  
2E: Door Lock (flexible disc)  
2F: Write Deleted Data Address Mark (flexible disc)  
30: Get Information about a File Code  
31: Rewind File (magnetic and cassette tape)  
33: Backspace one block (magnetic and cassette tape)  
34: Space one block forward (not allowed for cassette)  
36: Skip backward to EOF mark (magnetic and cassette tape)  
38: Unlock (cassette and magnetic tape)

37: Lock (cassette tape)/Off line (magnetic tape)  
 38: Compound Read (flexible disc)  
 39: Compound Write (flexible disc)  
 30: Search Key with Mask (flexible disc)  
 32: Write Deleted Data Address Mark and Verify (flexible disc)  
 3E: Search Key (flexible disc)  
 3F: Write Sector and Verify (flexible disc)

(Physical disc access on sector level is possible with orders /11 and /15, only for flexible disc and DAD. The Disc File Management module is not used in these cases. EOB5 must contain an absolute sector number.)

For each of these request orders, specific information applies to the various peripheral devices. This information is given in Appendix C at the end of the book.

The Event Control Block, of which the address must have been loaded into the A8 register, has the following format:

	0	7	8	15	
T/X	EVENT CHARACTER		FILE CODE		WORD 0
X		BUFFER ADDRESS			WORD 1
X		REQUIRED LENGTH			WORD 2
Y		EFFECTIVE LENGTH			WORD 3
Y		STATUS WORD			WORD 4
X	TAB. TABLE ADER./REL.SECT.NBR./ABS.SECT.NBR.				WORD 5

X: these words must be filled by the user

T: these words are filled by the monitor

where:

WORD 0: event character:

bit 0 = 1: end of operation has occurred for the EOB.  
 The other bits remain unused.

WORD 1: address of the user buffer

WORD 2: requested length to be read or written, in words (basic read on card reader) or characters (other devices). The first character is always the character given by the buffer address. For standard write on typewriter or line printer, two characters must be added, at the beginning of the buffer.

WORD 3: effective length which has been transmitted, in words (basic read on card reader) or characters (others). Stored here by the monitor upon completion of the I/O operation.

WORD 4: status word, stored here by the monitor upon completion of the requested I/O operation.

- For Basic orders, this word will be filled with the hardware status by the control unit. However, if the monitor detects an error in the calling sequence, bit 0 will be set to 1 and the other bits will contain the software status. (Hardware status: Appendix D).
- For the other orders, the software status will be returned:
  - bit 0 = 0: the operation has been successfully completed.
  - bit 7 = 1: no data (tape cassette)
  - bit 8 = 1: End-Of-Volume } cassette or
  - bit 9 = 1: End-Of-Tape } magnetic tape
  - bit 10 = 1: beginning of tape encountered.
  - bit 11 = 1: end of input medium (disc only).
  - bit 12 = 1: requested length is incorrect.
  - bit 13 = 1: illegal character code.
  - bit 14 = 1: an EOB mark has been read.
  - bit 15 = 1: an EOP mark has been read.

When the operation was not successfully completed, bit 0 is set to 1 and bit 1 is set to 0 (retry also was not possible). In this case bits 2 to 15 give the hardware status.

When the monitor has detected an error in the calling sequence, bits 0 and 1 are both set to 1 and bits 2 to 15 have the following significance:

- bit 2 = 1: power failure
- bit 5 = 1: disc overflow (no more granules available)
- bit 6 = 1: no disc buffer available (dynamic area overflow)

bit 7 = 1: disc queue overflow (queue area reserved at system is too small)  
bit 10 = 1: file is write-protected.  
bit 11 = 1: function is unknown or not compatible with the device or bit 9 in A7 set for a DGN request  
bit 12 = 1: illegal buffer size or address  
bit 13 = 1: illegal ECB address  
bit 14 = 1: device is attached to another program.  
bit 15 = 1: an illegal file code has been used.

WORD 5: this word is used by the user to store:  
- the relative sector number to be exchanged in the case of a random disc file (see Data Management).  
- the absolute sector number in the case of physical disc access (orders 11 and 15, and orders for flexible disc through the user driver for that device).  
- the tabulation table address in the case of a standard read operation on ASR or punched tape equipment. This tabulation table has the following format.

Number of Tackets	First Tacket
Second Tacket	Third Tacket

The tackets indicate an absolute position in the print line. Characters up to the following tacket are filled with blanks.

Example:

3	10
20	30

Input line: LABEL \OPER \OPERAND \COMMENT

Line in buffer:

LABEL.....OPER.....OPERAND.....COMMENT

1           10           20           30

At completion of the input, the buffer is filled with spaces, but the returned length is the length effectively entered and stored, including the spaces replacing the tabulation codes (\).

Note: If word 5 is used for tabulation, the required length in word 2 must contain both the characters and the blanks in the tackets, i.e. for the above example word 2 must be filled with 36.

## WAIT FOR AN EVENT

### Calling Sequence

LEKL	AB,ECBADR
LEM	
DATA	2

where:

ECBADR gives the address of the Event Control Block (see I/O requests). The first character of the ECB is the event character. If the first bit of this character is set to 1, the event has been completed.

### Use

This request causes a program to stop and wait for the completion of an event which has to take place in another program (user or system). If the event has occurred, the dispatcher returns control to the requesting program. If the event has not occurred, the program is put in wait state, to be restarted when the event has occurred.

### Note:

It is recommended not to use a Wait request inside a scheduled label routine, as this causes the whole program to be blocked temporarily.

There are two kinds of events:

- Wait for the exit of a program:

When a user program activates another program (see Activate), the first word pointed to by the A8 register is the address of the ECB which must be used to wait for the exit of the activated program. As the activated program and the calling program run concurrently, this provides a means of communication between the two programs.

- Wait for an I/O operation.

The program waits for the end of an I/O operation it has requested.

Notes:

- In case of explicit or implicit wait (implicit wait inside I/O, Attach Device, etc.) processed by a Read Only program, the Read Only Area will be forbidden during the whole waiting time for all Read Only programs with equal or lower priority, no explicit wait or conditions resulting in implicit wait should be avoided in Read Only programs.
- When this request is used by a swappable program and bit 15 of A8 is set to 1 while the event has not been set (LEM 16), the calling program will be suspended immediately and it may then be swapped out as soon as its swap event count becomes zero.
- The user can create his own set of events. In this case he must reset the event bit in the EOB and inform the system by giving a Set Event monitor request.
- This request is processed at level 48 by a memory resident program.

EXIT

Calling Sequence

LKM
DATA 3

Use

This request is used to specify the end of a program. The program exit is effected after completion of all I/O operations and after all labels, if any, have been scheduled. A scheduled label exit passes control to the next scheduled label, if one is present, otherwise control passes to the main program.

The program becomes inactive, unless there is another activation request waiting for this program.

The program remains connected to its level.

The ECB supplied at the activation of the program must be updated by giving a 'Set Event' monitor request (LKM 18), to set bit 0 of the event word to 1.

This request is processed at level 48 by a memory resident program.

Note: For swappable programs, the contents of the Swap Area will not be swapped out.

### GET BUFFER REQUEST

#### Calling Sequence

LDW	A7, LENGTH
LBN	
DATA 4	

where:

LENGTH is the length, in characters, to be allocated to the buffer area (maximum 32k characters):



If bit 0 = 0: return to user in case of overflow.

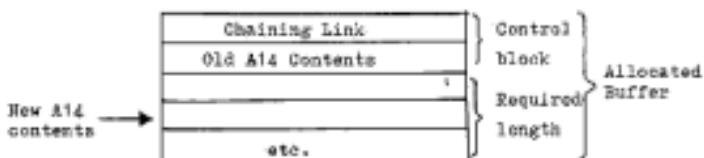
If bit 0 = 1: implicit wait of the calling program in case of overflow. This should be avoided in read only programs.

If 0 is loaded into A7, the monitor will return the highest address of memory in A7, except if the memory size is 32k, in which case 0 will be returned in A7.

#### Use

By means of this request, the user can allocate a memory area for temporary use, in the dynamic memory allocation area.

When the allocation is made, a control block is created by the system at the beginning of the allocated area. This block will contain a chaining link and the old contents of the A14 register:



The user must not destroy this control block

Upon completion of the request, the system responds as follows:

- A7 = 0: the buffer is allocated
- = 1: there is no memory space available (bit 0 in LENGTH = 0)
- A14: contains the address of the fourth word of the allocated buffer, so that, as soon as the buffer is allocated, the user may give a Call Function instruction with the A14 register without having to update the A14 register first. However, the user must then provide for stack handling.

This request is processed at level 48 by a memory resident program.

### RELEASE BUFFER REQUEST

#### Calling Sequence

LDKL	A14, BUPADR
LEM	
DATA 5	

where:

BUPADR points to the second word in the buffer as given in register A14 after the Get Buffer request.

#### Use

To release the memory space previously reserved by a Get Buffer request. The A14 register is reloaded with the value it contained before the Get Buffer request was made.

The system responds as follows:

A7 = 0: the memory space is released.

    = -1: the memory space has already been released.

If the A14 pointer is incorrect, or if the buffer area has been destroyed, the system issues a Halt.

If the dynamic area was in overflow state, this request frees it again, and programs waiting because of buffer overflow are restarted, and their requests reinitialized.

This request is processed at level 48 by a memory resident program.

### CONNECT A PROGRAM TO A TIMER

#### Calling Sequence

LINKL	A8, PARAM
LINKL	A7, PNAME
LEM	
DATA	10

where:

PARAM points to a two-word block, containing the necessary parameters.

Two formats are possible for this block:

- format 1 (bit 0 = 0):

0	1	2	3	4	5	15
0	TIMER NUMBER		PULSE RATE (PR)			
NUMBER OF CYCLES BEFORE FIRST ACTIVATION (NC)						

NTIM: the number of the timer (0-real time clock)

PR: requested pulse rate, i.e. the number of cycles of the timer between two activations: a number from 0 to 2047. If 0 is specified, the program is activated only once and then automatically disconnected.

NC: number of cycles of the timer before the first activation: a number from 0 to 32767.

- format 2 (bit 0 = 1):

0	1	2	3	4	5	15
1	TIMER NUMBER	HH	MM	PULSE RATE		
MM						SS

where NC is replaced by an absolute time: HH (hour) - MM (minute) - SS (second), and PR is number from 0 to 127.

Use:

A program running at a software level can be connected to a timer, according to the parameters given in register A8. The program must have been connected to a software level, otherwise it cannot be started by the dispatcher.

For timer numbering, see part 2, page 2-60.

The system responds as follows:

A7 = 0: the connection has been made.

A7 ≠ 0: connection is impossible, because:

- the specified timer has not been defined, or the program does not exist: A7 = -5;
- the program has already been connected to a timer: A7 = -2;
- dynamic area overflow: A7 = -4.

Note:

The program must have been declared by the system as memory resident, swappable or read only. Background programs cannot be connected to a timer.

- the program processing this request may be memory resident or read only at level 49. If it is read only, it cannot be used by user read only programs connected to level 49.

### DISCONNECT A PROGRAM FROM A TIMER

#### Calling Sequence

LDK	A8,NTIM
LDBL	A7,PRNAME
LEM	
DATA	11

where:

NTIM is a constant specifying the timer number.

PRNAME points to a 3-word block containing the program name.

#### Use

A program can, by means of this request, be disconnected from the timer specified in register A8.

The program remains connected to its software level.

The system responds as follows:

A7 = 0: the program is disconnected from the timer

A7 = -3: it is impossible to disconnect the program, because the program does not exist, or was not connected to this timer.

The program processing this request may be memory resident or read only, at level 49. If it is read only, it cannot be used by user read only programs connected to level 49.

## ACTIVATE A PROGRAM

### Calling Sequence

LDKL	A7, PRNAME
LDKL	A8, BLOCK
LMR	
DATA	12

where:

PRNAME points to a 3-word block containing the name of the program to be activated.

BLOCK points to a 2-word block of the following format:

Event Character	ECB
Parameter Block Address	

The first word of this block may be updated by means of a 'Set Event' monitor request (LMR16) at the exit of the activated program, so that, if the calling program has requested a wait for the activated program, this word may be considered its Event Control Block (see Wait Request). The second word contains the address of a parameter block. The calling program may give a Get Buffer Request to build this block in the Buffer pool, before the Activate request is given. Register A4 will then contain the address of the parameter block when the activated program is started by the dispatcher and register A14 will contain zero. That is, register A4 contains the value in ECB+2 and A8 points to the ECB for a possible Set Event request.

### Note:

When a user program is activated by an interrupt sequence, there is no parameter block and the contents of A4 is not significant. A14 is set to zero.

Use

This request can be made by a program running at any level, to activate a software level program.

Calling program and activated program may be processed concurrently, depending on their priority levels.

If the activated program is busy, the request is recorded in a stack, to be processed later.

The activated program must previously have been connected to a level.

The system responds as follows:

$A7 = 0$ : the request has been processed or recorded in a stack;

$A7 \neq 0$ : the request has not been taken into account, because:

- the program has not been connected to a level:  $A7 = -2$ ;
- the program does not exist:  $A7 = -3$ ;
- dynamic area overflow:  $A7 = -4$ ;
- PCT pool overflow:  $A7 = -5$ ;
- Save Area pool overflow:  $A7 = -6$ ;
- disk I/O error:  $A7 = -7$ .

- the program is too large for the background areas:  $A7 = -8$

The program processing this request is a memory resident program at level 48 for activation of memory resident, read only or swappable programs. Requests for activation of background programs are handled by a program at level 49, which may be either memory resident or read only.

SWITCH INSIDE A SOFTWARE LEVEL

Calling Sequence

LDK	A7, LEVEL
LMN	
DATA 13	

where:

LEVEL is a constant specifying the number of the level in which the switch is to be made. If LEVEL is specified as 0, the level to be switched is equal to the requesting level plus one.

Use

By means of this request it is possible to have timeslicing inside a software level (only for core resident programs or background programs which are already in core) by halting execution of the program running on that level (A7) and giving control to the next program on the same level.

See also page 1-28.

This request is processed at level 48 by a memory resident program.

## ATTACH A DEVICE TO A PROGRAM

### Calling Sequence

LDE	A7, FLAG
LDEBL	A8, DEVBLLK
LBN	
DATA	14

where:

FLAG is either zero or not zero. When it is zero, control is returned to the calling program with -3 in A7, if the device was already attached to another program.

When it is not zero, the calling program is put in wait until the device is detached. This should be avoided for read only programs (see unit request).

DEVBLLK points to a one-word block containing:

	FC
--	----

where:

FC is the file code assigned to the device concerned or a disc logical file code.

### Use

By means of this request, a program running at any software level can obtain exclusive use of the device specified. Other programs will be unable to perform I/O operations on this device.

If the device has already been attached to another program, the requesting program is put in wait state until the device is detached by the other program (see following request).

This is checked via word 34 in the D.W.T. where bit 0 is 1 if the device is not attached. If it is attached, the wait is done on this bit.

If the file code has already been attached to another program and the Wait flag has been set to one, the calling program is put in Wait and the Time Slice is set to zero.

In case several programs have given an attach request and are waiting for a device to be detached, the highest level program will receive control first.

The system responds as follows:

A7 = 0: the device has been attached;

A7 = -2: the device does not exist, or it is impossible to attach it to this program.

A7 = -4: the file code specified is assigned to an extended file.

Notes: - The ASR is considered as 3 devices, so 3 requests must be given to attach the ASR keyboard, tape reader and tape punch.

- When used with a disc file code, only the specified disc file is attached, the rest of the disc remains free for other programs. With DAD file code, only the DAD is attached.
- This request must not be used with file codes F0 to FF.
- User programs must not attach devices which may be used by the system, because in that case the results may be unpredictable.

The program processing this request may be memory resident or read only, at level 49. If it is read only, it cannot be used by user read only programs connected to level 49.

DETACH A DEVICE FROM A PROGRAM

Calling Sequence

LDCL A8, DEVELK
LKM
DATA 15

where:

DEVELK points to a one-word block of the following format:

PC
----

where:

PC is the file code of the device concerned.

Use

By means of this request, a device which has been attached to a program by an Attach request (LKM 14), is detached from that program.

To prevent another program having to wait un-necessarily for this device, the Detach request must be made as soon as the device is no longer required.

The system responds as follows:

A7 = 0: the device has been attached;

A7 = -2: the device does not exist, or it is attached to another program

The program processing this request may be memory resident or read only, at level 49. If it is read only, it cannot be used by user read only programs connected to level 49.

### GET TIME

#### Calling Sequence

LINK A7, FLAG
LINKL A8, TIMBLK
LEM
DATA 17

where:

FLAG is a value of either 0 or 1, to specify whether the time will be output in ASCII (0) or binary (1) format.  
TIMBLK points to a six-word block into which the monitor will load the requested information.

#### Use

Upon receipt of this request, the monitor will return the time to the block specified in register A8, in the format indicated by the flag in register A7.

If A7 = 0, the information will be in the following format:

DAY	word 0
MONTH	1
YEAR	2
HOUR	3
MINUTE	4
SECOND	5

If A7 = 1, the information will be in the following format:

	word 8
HOUR	1
MINUTE	2
SECOND	3
Tenth of Second	4
Hundred of Second	5
C:TIME	

The values are given in binary.

C:TIME is 0, if the standard clock is included.

If a non-standard clock is included, it contains the pulse rate of this clock.

The program processing this request may be memory resident or read only, at level 49. If it is read only, it cannot be used by user read only programs connected to level 49.

SET AN EVENT

Calling Sequence

LEVEL	48, ECBADR
LEM	
DATA	18

where:

The user can create his own set of events. Each time such an event has occurred, the user must inform the system about it by giving this request, so that it can restart any programs which were waiting for this event.

The system sets the event bit to 1.

For example, the system sends this request after monitor requests for I/O (twice: for ECB and for controller status), Release Buffer, Detach Device and Exit.

This request is processed at level 48 by a memory resident program.

Note: This request can be made by an interrupt routine. In that case, the interrupt routine is stopped and a branch made to level 48. After the request has been processed, a return is made to the interrupt routine, which is then restarted at its own hardware level.

## CONNECT A PROGRAM TO A SOFTWARE LEVEL

### Calling Sequence

LINK	A7, NUMBER
LINKL	AS, PRNAME
LEM	
DATA	20

where:

NUMBER is the number of the software level to which the program is to be connected.

PRNAME points to a 3-word block containing the name of the program.

### Use

A program, running at any level can make this request for another program to be connected to a software level.

The system responds as follows:

A7 = 0: the connection has been accomplished.

A7 ≠ 0: it is impossible to make the connection, because:

- the program does not exist: A7 = -3;
- the program is not compatible with the requested level.  
i.e. it is an interrupt level: A7 = -5;
- the program has already been connected to a level:  
A7 = -2.

The program processing this request may be memory resident or read only, at level 49. If it is read only, it cannot be used by user read only programs connected to level 49.

Note: This request must not be used to connect background programs (level 62) or the idle task (63). This is done automatically.

DISCONNECT A PROGRAM FROM A LEVEL

Calling Sequence

LDX	A7, NUMBER
LDXL	AB, PRNAME
LXM	
DATA	21

where:

NUMBER is the number of the software level to which the program has been connected.

PRNAME points to a 3-word block containing the name of the program which is to be disconnected.

Use

The program specified is disconnected from the level given in register A7.

The system responds as follows:

A7 = 0: the program is disconnected.

A7 ≠ 0: it is impossible to disconnect the program, because:

- the program does not exist: A7 = -3;
- the program is busy: A7 = -2;

The program processing this request may be memory resident or read only, at level 49. If it is read only, it cannot be used by user read only programs connected to level 49.

Note: This request must not be used with the levels 62 and 63.

## WAIT FOR A GIVEN TIME

### Calling Sequence

LDKL	A8, PARAM
LEM	
DATA 22	

where:

PARAM is the address of a 3-word block of the following format:

WORD				
0	TIMER NUMBER	W G T	IRRELEVANT	
9 1	2 4 5	15	NC	

where word 0 contains zero, as the EOB on which the calling program will be put in wait state, must be equal to zero.

word 1: Bit 0 = 0

    Bits 1-3: timer number of the timer for which the program waits. See under LEM 10.

    Bit 4: WGT flag: any value; destroyed by the system.

    Bits 5-15: not significant.

word 2: Number of cycles, of the time defined in word 1, to be made before the program is restarted.

Note: When this request is used by a swappable program and bit 15 of A8 is set to 1, the calling program may be swapped out immediately, provided its swap event count is zero.

### Use

This request is given to put a program in wait state, until a certain time has passed, as indicated in word 2 of the PARAM block. The precision of the waiting time depends on the precision of the timer defined in word 1 of this block.

When the number of cycles specified has passed, the requesting program is restarted with the system response in A7: 0 = request performed correctly.

If the response in A7 is -1, the wrong timer has been defined.  
If it is -4, there has been dynamic area overflow.  
The request is processed at level 49, by a memory resident or  
read only program.  
If it is read only, it cannot be used by user read only programs  
connected to level 49.

ASSIGN A FILE CODE

Calling Sequence

LDEBL	A8, ASBLK
LCM	
DATA	23

where:

ASBLK is a parameter block with the following layout:

- for assignment to a peripheral:

00000000	file code
	device name
	device address

- for assignment to a temporary file:

00000001	file code
D	- disc file code
	number of granules
	not used
	UF
	DAD name

- for assignment to a permanent file:

00000010	file code
D	- disc file code
	file name
	UF
	DAD name

- for assignment to an extended file:

00000100	file code
D	- disc file code
	file name
	UF
	DAD name

- for assignment to a DADs

00000011	file code
D	disc file code
	not used
	DAD name

- for assignment to another file code:

00000011	file code
	2nd file code
	not used

The DAD specified in the second word of the block, can be specified in one of three ways:

- by specifying the DAD file code previously assigned by an Assign of type 5).
- by specifying only the disc file code (FO to FF), thus implicitly specifying the first DAD on the disc
- by specifying disc file code and DAD name, in which case bit D must be set to 1, to indicate that the block contains a DAD name.

For assignment to a DAD and another file code, the same assign type must be used (/05). The distinction lies in the second word: If it contains a DAD or disc file code it is a DAD assignment, otherwise it is an assignment to another file code.

#### Use

By means of this monitor request the user may assign a file code to either a physical device or a disc file.

#### Notes:

- A file code can be assigned to only one physical device or file at a time; however, a physical device or file can be used through different file codes.
- As the ZXTM is used as a supervisor essentially, error checking facilities are limited and system file codes are not protected.

Therefore, the user must be very careful not to destroy any system (especially disc) file codes.

- When a new file code is assigned, the old assignment is first deleted. This implies, that when an assign request is refused, the old file code no longer exists and a new request must be given to restore the old assignment, if desired.
- This request is handled by a module connected to level 49, which may be either memory resident or read only.

Upon completion of the request, the system responds as follows:

A7 = 0: assignment performed

/ 0: assignment refused, for one of the following reasons:

- 1: I/O error on disc
- 2: no spare entry in file code table
- 3: no file description table available
- 4: device or disc file code unknown
- 5: disc overflow or too many granules requested
- 6: file unknown
- 7: second file code unknown (assign type/11)
- 8: more than 7 file codes assigned to the same disc file.

### DELETE A FILE CODE

#### Calling Sequence

LDXL	AS,PARAM
LEN	
DATA	24

where:

PARAM points to a parameter word:

0	File Code
---	-----------

File code indicates the file code which must be deleted.

#### Use

By means of this request the user can ask the system to delete a file code which has previously been assigned by monitor request or control command.

If the file code had already been deleted, this request will be ineffective. If the file code does exist, it is removed from the file code table.

If the file code to be deleted is a disc file code, its file description table will be released and, for temporary files, the granules of the file will become available again.

This request is handled by a module on level 49, which may be either memory resident or read only.

Note: The system does not protect any file codes, system or user.

For sequential files, it will be assumed that they have already been closed implicitly by EOS or EOF mark detection. The system responds as follows:

AT = 0: the file code has been deleted

= 1: I/O error occurred while reading the file to release the granules occupied by the temporary file, if any. The file code is still removed from the file code table.

#### READ UNSOLICITED OPERATOR MESSAGE

##### Calling Sequence

LDKL	AS,	PARAM
LEM		
DATA	25	

where:

PARAM points to a parameter block with the following layout:

WORD	0	Event Character	
	1	Buffer Address	
	2	Requested Length	
	3	Reserved	
	4	Reserved	
	5	Special Characters	

The first bit of the event character is used to indicate whether the event has occurred or not.

Word 1 contains the address of the first character of the buffer where the message will be stored with its two leading characters.

Word 2 contains a value equal to the length of the buffer.

Word 5 contains the two leading characters which are used to identify the message.

##### Use

By means of this request the operator is given the possibility of interrupting the system through use of the INT button on the control panel and sending a message to the program in which this request was made. For the system to be able to recognise the message and send it to the right program, the operator must start the message with two special characters, as he has defined them in word 5 of the PARAM block, described above. It is recommended to use not alphanumeric characters, but special characters, so as not to interfere with the regular operator commands.

When the system has found the two characters matching the ones that were typed in, the message is stored in the user buffer and the event is declared as having occurred. The scheduled label facility can be used to process the message.

The message must be sent from the system keyboard (file code /EP) and the maximum message length is 72 characters.

The event count is incremented when the request is given and decremented when the operator types in the message.

If more requests are given, with the same identification characters, they are serviced first-in-first-out. Requests are queued, but removed from the queue after servicing, so if a message is to be given more than once, the request must be reinitialized.

Note: If this monitor request has been given, the operator must type in a message with the two special characters or give a monitor request (LKM26) to cancel this one, otherwise the program will not be able to make its exit (see below).

CANCEL REQUEST FOR AN UNSOLICITED OPERATOR MESSAGE

Calling Sequence

LDEL	A8, PARAM
LEM	
DATA	26

where:

PARAM points to the same block to which is referred under the Request for an Unsolicited Operator Message (LEM25).

Use

When a monitor request is issued in a program to Read an Unsolicited Operator Message (LEM25), the event count is incremented by the monitor and decremented again only when the operator types in the defined message.

For a program to be able to make its exit, the event count must be zero. So, if it has been un-necessary or undesirable to type in the message, this request (LEM26) can be issued to have the event count decremented and the LEM25 request removed from the queue. The effect is that the event is considered as having occurred: the event bit is set and, if a scheduled label is attached to this request, it is started.

## **APPENDICES**

---

To standardize the system generation procedure for all systems, a set of system generation processors has been developed which provides great flexibility, extensive logging of the process and improved efficiency.

The three steps inherent in any system generation process, i.e. monitor tables generation, monitor body generation and system medium generation are handled by a number of generation processors which are loaded and started successively.

For the generation of a Disc Real Time Monitor these are:

- GENMON, a generation monitor used only during the system generation process to run the generation processors.
- DRMGEN, which generates the monitor tables from the answers it receives from the user in a conversational process with a standard list of questions.
- PREMDK, which is used to premark the system disc and write an Initial Program Loader on it, as the first module on the system disc. PREMDK runs under any monitor.
- GENLKE, which runs under GENMON and scans the library of system modules (DRTMLIB), to select the ones requested during the DRMGEN phase and link them with the tables generated during DRMGEN.
- DISLOAD, which, running under GENMON, is used to record the system processors on disc in load module format.

Depending on his configuration, the user may receive his system tools, i.e. the above-mentioned processors, on punched tape or on cassette. In the first case, each processor is contained on a separate punched tape, in the second case all the processors are contained on one cassette.

In the description which follows in the paragraphs below, the cassette case is the basic as this is the assumed standard for this system process. It is very easy for the user, however, to redefine these standards (under GENMON) in case he works with punched tape. Apart from the redefinition of the standards, the main difference in the description is that from cassette the successive processors can be loaded and started without any manual

operation, whereas with punched tape, for each following step a new tape with the following system processor must be put on the tape reader and then loaded and started.

Note: For DataCom SysGen, see P800M Data Communication User Manual. In the following paragraphs, the whole set of operations necessary for the generation of a pgm is described in a number of sections corresponding to the system generation processors listed above. At the end of each section a number of notes and remarks is given, which the user must carefully read before starting the operation.

#### OPERATION

The minimum configuration required for generating a DRTM is:

- CPU with 16k memory
- typewriter with address 10
- paper tape reader and punch, or
- two magnetic tape cassette drives on 1 control unit
- one X1215 disc unit

If the configuration is paper tape-oriented, the user receives 18 tapes, containing:

- IPL + GENMON
- DRMGEN
- PREMDK
- GENLKE
- DRMLIB 1
- DRMLIB 2
- DISLOD
- one tape for each of the monitor segments:  
  D:USV1, D:USV2, D:USV3, D:USV4, D:OCOM, D:DUMP, D:ROOT,  
  D:SEG1, D:SEG2, D:SEG3, D:SEG4.

If the configuration is cassette tape-oriented, the user receives two so called generation cassettes (G1 and G2), containing:

```

- cassette G1: side A: IPL
    GENMON
    DRMCEN
    FREMBK
    GENLKE
    side B: DRTMLIB 1
- cassette G2: side A: DRTMLIB 2
    DISLGD
    side B: monitor segments:
    D:USV1
    D:USV2
    D:USV3
    D:USV4
    D:OCOM
    D:DUMP
    D:ROOT
    D:SEG1
    D:SEG2
    D:SEG3
    D:SEG4

```

The user needs two cassettes or paper tapes of his own, to be used for intermediate storage of sysgen output.

Note:

Throughout this chapter, user replies typed in response to questions output by one of the sysgen processors, are underlined.

To start the process:

- switch on the CPU
- for cassette 1
  - load generation cassette 1 (hereafter called cassette G1) in cassette drive TK05 with side A up
  - set the data switches on the CPU control panel to allow the bootstrap to load from TK05:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	1	1	1	0	0	0	0	1	0	1

(hexa 0785)

(for the significance of the bits, see page 1-64; suffice it to mention here that bit 3 is 0 if the cassette drives are connected to the I/O processor and 1 if they are connected to the programmed channel, and bits 10 to 15 contain the device address.)

- for paper tape:

- put the tape containing IPL +GENMON on the tape reader and make it operable
- switch on the paper tape punch and feed tape
- set the data switches on the CPU control panel to allow the bootstrap to load from the tape reader:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0

(for the significance of the bits, see page 1-64; suffice it to mention here that bit 3 is 1 because the reader is connected to Programmed channel and that bits 10 to 15 contain the device address which is here assumed to be 20).

Then:

- press the MC button
- press the IPL button

Now the bootstrap is loaded which loads the first system program from the cassette in TK05 into memory:

GENMON

GENMON is a special monitor used only during the system process. To be able to do this, it must know the system configuration, the device addresses, interrupt levels and file codes. It is here that this system generation procedure shows its flexibility, for the great number of definition possibilities which the user has at this point. GENMON outputs two questions, allowing the user to give his definitions and assignments. However, the system generation will handle a set of

built-in standards if these are acceptable to the user. In this case he does not have to define anything, but if one of the user's assignments or definitions is different from the standard ones as listed below, he must redefine under GENMON.

When GENMON is loaded its identification is output on the typewriter:

GENMON

When loading is terminated,

:EOS

:EOF

is output on the typewriter, followed by the question

STANDARD CONFIGURATION?

The reply to this question can be Y or N.

If the user replies Y or YES followed by LP CR, GENMON assumes the following standard configuration definition:

- typewriter : TY10 at level /6
- tape reader : PR20 at level /4
- tape punch : PP30 at level /3
- line printer : LP07 at level /17
- card reader : CR06 at level /15
- cassette tape : TK05 at level /14  
                  TK15 at level /14  
                  TK25 at level /14
- magnetic tape : MT04 at level /13  
                  MT14 at level /13  
                  MT24 at level /13
- XII215 disc : BM02 at level /10 (removable cartridge).

(It is not necessary for the user to have all these devices in his configuration to be able to answer YES; but the ones he does have must then correspond to these standards.)

If the user replies N or NO, i.e. if one or more of the device addresses or levels is different from the standards above, GENMON outputs the following list on the typewriter, thereby allowing the user to define the configuration himself:

TY:  
PP:  
PR:  
LP:  
TK:  
MT:  
CR:  
DK:  
DISK TYPE: (type is B)  
LKM LEVEL: (standard = 1)  
RTC LEVEL: (standard = 2)  
PANEL INTERRUPT LEVEL: (standard = 7)

For each of the devices listed, the user can reply as follows:  
- (CR) if he wants the standard address and level (see above);  
- <address>,<level> if one of these is different from the standards, followed by (CR);  
- N or NO followed by (CR) if he wants the device excluded from the system.

When the user has terminated his reply to the first question,  
GENMON types out:  
STANDARD FILE CODE ASSIGNMENT?

The procedure here is the same as for the first question:  
the reply may be either YES or NO.

If the user replies Y or YES followed by (LF) (CR) GENMON assumes the following standard file code assignments:

- file code 1 : TY10 (system keyboard)
- file code 2 : LP07 (listing output)
- file code 3 : TK15 (object output)
- file code 4 : TK05 (object input)
- file code 5 : TY10 (system keyboard)
- file code 6 : TK05 (object input)
- file code 7 : TK25 (object input)
- file code 8 : PR20 (object input)
- file code A : TY10 (sysgen source input)

- file code B : TK15 (sysgen object output)
- file code E0: TY10 (system keyboard)
- file code E1: TK05 (Disload object input)
- file code E7: TY10 (system keyboard)
- file code F0: BM02 (Disload disc output; i.e. X1215)

If the user replies N or NO to this second GENMON question, i.e. if one or more of his file code assignments is going to be different from the standard list above GENMON outputs the following list on the typewriter, thereby allowing the user to give his own file code assignments (since the GENMON program is the same for all monitors, some of the file codes given in this list may be irrelevant to the generation of a DRTM and the user must type in NO after those):

LOAD INPUT DEV. AND MAIN LKE INPUT DEV. F.C./4: (standard = TK05)	
SYSGEN INPUT DEV.	F.C./A: (standard = TY10)
SYSGEN OUTPUT DEV.	F.C./B: (standard = TK15)
AUX. LKE INPUT DEV 1	F.C./6: (standard = TK05)
AUX. LKE INPUT DEV 2	F.C./7: (standard = TK25)
AUX. LKE INPUT DEV 3	F.C./8: (standard = PR20)
AUX. LKE INPUT DEV 4	F.C./9: (no standard)
IPLGEN/LKE/CASLOAD OUTPUT DEV.	F.C./3: (standard = TK15)
LISTING OUTPUT DEV	F.C./2: (standard = LP07)
CASLOAD INPUT DEV.	F.C./C: (type in <u>NO</u> )
DISLOAD INPUT DEV.	F.C./E2: (standard = TK05)

For each of the file codes listed, the user can reply as follows:

- CR if he wants the standard assignment (see above)
- <dev.name><dev.address> if one of these is different from the standards, followed by LP CR
- N or NO followed by CR if he does not want this file code taken into account
- if there is only one device of its kind, e.g. one PP, or in case a device which must be taken is the first of a series encountered in the standard list above, e.g. TK05, it suffices to specify only the device name, i.e. PP or TK.

When the user has terminated his reply to this question,  
GENMON types out:  
END OF GENMON INITIALIZATION  
READY TO LOAD PROGRAMS

Now the user can proceed to the next phase: DRNGEN.

Notes on GENMON:

For the question STANDARD CONFIGURATION:

- From the time the MC button has been pushed up to the end of GENMON initialization, no ready interrupts should occur.
- If the user has answered N or NO to this question, in the list typed out by GENMON, specification of a level is mandatory for LKM LEVEL. For RTC LEVEL it is mandatory if the CPU key is in the RTC/ON or LOCK position. For PANEL LEVEL it is also mandatory.
- The standards imply that the system disc will be the removable cartridge of the X1215 disc unit; therefore, if the user wants his system on the fixed cartridge, he must redefine DK under this question, e.g. as BM22.

For the question STANDARD FILE CODE ASSIGNMENT:

- File code /4: all programs will be loaded from this file code. During the syslink phase it is used as GENLKE object input, so it must be cassette or punched tape.
- File code /A: from this file code the parameters for table generation will be read. This may be done in interactive mode (e.g. TY) or not (e.g. PTK, cassette or magtape or card reader).
- File code /B: this may be cassette, punched tape or magnetic tape.
- AUX, INPUT DEV,: these may be assigned in advance, especially for syslink if libraries are to be scanned on various devices.
- File code /3: this is the main output device (sequential), i.e. cassette tape, punched tape or magnetic tape.
- File code /2: for logging of the sysgen operation LP.
- File codes /4, /A, /B and /3 are mandatory;  
file codes /6 up to /9 and /2 are optional.

When answers to GENMON questions are given on an ASR type-

writer, they must not be typed in before the bell signal because of the low speed of the GENMON I/O module. This is not necessary for devices on V-24 interface such as the matrix typewriter PB42, because they work in echo mode.

#### DRMGEN

When GENMON initialisation is terminated and the message READY TO LOAD PROGRAMS has been output the user can start the second phase, DRMGEN, the building of the BRTM tables. This is done by typing replies to questions output on the typewriter by DRMGEN. From these replies DRMGEN builds the tables and records them on the medium with file code /B, i.e. in standard cases the cassette in drive TK15. When the questions and answers are handled via the typewriter, this phase is done in conversational mode. It is also possible however, to do it in non-conversational mode, for example by having the questions and answers pre-recorded on punched tape. In such a case, file code /A must have been assigned to tape reader during the GENMON phase under the question STANDARD FILE CODE ASSIGNMENT

The following actions must now be taken:

- when working with paper tape, take the IPL+GENMON tape from the reader, put the DRMGEN tape on it and make it operable; prepare the tape punch for output by switching it on and feeding tape
- when working with cassettes, put the first working cassette (hereafter called cassette M1) in cassette drive TK15; this cassette will receive the DRMGEN output.
- load a disc cartridge in the X1215 disc drive (this is the disc which will later on receive the generated system, depending on the disc defined under DK in the question STANDARD CONFIGURATION during GENMON)
- push the INT button on the CPU control panel
- on output of M1 type in  
LD
- now the following system processor, i.e. DRMGEN is loaded from the cassette or paper tape and its identification is output:  
IDEST DRMGEN

- when loading is terminated,

:EOS

:EOF are output on the typewriter

(At this point, if file codes /A and/or /B have been redefined under GENMON, i.e. if they are not assigned to TY10 or to a cassette respectively, prepare the relevant device. Normally, the cassette in drive TK15 should now be ready to receive the tables output by DRMGEN according to the replies given on TY10).

- push the INT button

- on output of M: type in

ST

- now DRMGEN is started and outputs the following message on the typewriter:

TABGEN INITIALIZATION

IDENT SYSTEM DEFINITION

- then, if the user works in conversational mode, a series of questions is output, to which the user must type in the replies:

#### IDENT

Reply:

Specify a character string of up to 6 alphanumeric characters, to be punched at the beginning of the module. This may be followed by a characters string of up to 73 characters, containing comments. The comment field must be separated from the ident field by a blank.

Error Message: TG03: the first character is not alphabetic.

#### STACK SIZE

Reply:

Specify 4 hexadecimal characters, giving the size, in words, of the stack which is used by the system to save registers when an interrupt occurs.

Note that 10 decimal words are required in the stack for each accepted interrupt.

Error Message: TG03: the reply is not a hexadecimal number.

## DYN AREA SIZE

### Reply:

Specify up to 4 hexadecimal characters, giving the size, in characters, of the dynamic allocation area in memory, shared by the system and user programs.

The minimum size to be declared is 7580.

Note that, in addition to user requests for memory space, the system also requires memory space to process some of the user requests and external events, i.e.

- to activate a background program the system requires 205<sub>10</sub> words during program loading;
- to assign a disc logical file, the system requires 205<sub>10</sub> words during the assign process;
- when a disc sequential file is used the system requires 205<sub>10</sub> words as long as that file remains opened;
- for the requests Connect a Level to a Clock or to a Timer and Wait for a given Time the system requires 4<sub>10</sub> words from the time the request is issued until the program is disconnected from the time or until the given time has elapsed;
- for an activate request (from user as well as from system) 6<sub>10</sub> words are required if the program to be activated is busy, until its activation becomes effective;
- for a request to Read an Unsolicited Key-In, 7<sub>10</sub> words are required until the character string is input or until the request is cancelled.

Note: In addition to the size requested, the system reserves one extra word for block management in this partition.

### Error Message:

TC03: the reply is not a hexadecimal number.

#### READ ONLY LENGTH

Reply: Specify up to 4 hexadecimal characters giving the length, in characters, of the Read Only Area. This length must be equal to the size of the largest Read Only program used. It may not be smaller than /980, i.e. the size of the system read only program.

Error Message: TGO3: the reply is not a hexadecimal number.

#### SWAP AREA LENGTH

Reply: Specify up to 4 hexadecimal characters giving the length, in characters, of the Swap Area. (Minimum: /196 characters). This area is optional, so the reply may be 0.

Error Message: TGO3: the reply is not a hexadecimal number.

#### SWAPPING FILE CODE

Reply: Specify the file code of the disc (from /P0 to /77) on which the core image file D:CI will be stored when swapping occurs. This file is used for read only as well as for swappable programs.

Error Message: TGO3: invalid reply.

#### SWAPPING NUMBER OF GRANULES

Reply: Specify the size of the D:CI file, in granules. The minimum size must be 8 granules, due to the size of the system read only programs. For user programs, the size must be defined as follows:

- Read Only Programs:  
if  $L = \text{program length}$ , then the number of

sectors required for this program is:

$$\frac{L - 1}{200} + 1$$

10

- Swappable Programs:

if  $S$  = length of the swap area, the number of sectors required for each swappable program is:

$$\frac{S - 1}{200} + 1$$

10

After the requirement for each program has been calculated, add all the results, divide by 8 and add 9 (upper value rounded + 8 granules for the system read only programs) to obtain the definite reply, i.e. the number of swapping granules required.

Error Message: TG03: invalid answer.

#### SWAPPING TIME SLICE

Reply: Specify the standard time slice for swapping, in tenths of a second.

Because the System Loader cannot be interrupted by other software level programs (its standard level is 49), the time slice must be defined according to the type of disc on which the D:CI file is stored and so the length of the Swap Area.

Error Message: TG03: invalid answer.

#### CORE RESIDENT AREA LENGTH

Reply: Specify up to 4 hexadecimal characters giving the length, in characters, of the Core Resident Area. The size must be equal to the total length of the user programs declared as core resident at initialization time. If the area is not used, 0 must be specified.

Error Message: TG03: invalid reply.

#### PROG SAVE AREA LENGTH

Reply: Specify up to 4 hexadecimal characters giving the length, in characters, of the Program Save Area.

Note:

- Since the 10 system read only programs also use this area the minimum size declared must be /154.
- For user programs it must be extended as follows:
  - for each core resident, read only or swappable user program, add:  
 $17_{10}$  words if no scheduled label is used,  
 $34_{10} + 2n$  words if n scheduled labels will be declared in the program.
  - for a background program, a save area is allocated only when the program is in memory:  
 $34_{10} + 2n$  words, where n is the same value which is given in reply to the question MAX NUMBER OF SCHEDULED LABELS

Error Message: TGO3: invalid reply.

#### USER INTERRUPT ROUTINES

Reply: Specify all user interrupt routines (usually drivers for special devices) which must be linked with the monitor for inclusion in the system, as follows:

L,<ENTRYI>

END

L, ENTRYI specifies the name (up to 6 characters) of a routine connected to the single level L.

END indicates that all routines have been declared or is used if none are entered.

Error Messages: T003: syntax error  
T006: the first character of the name (ENTER1/j)  
is not alphabetic.  
T009: error in the level declaration.

#### POWER FAILURE

Reply: Specify the level of power failure option as follows:  
(L) giving the level number 1 or 2 hexadecimal digits.  
  
Specify N if this feature is not available in the system.  
  
Note: The standard system power failure routine performs only a HALT.  
Error Message: T003: parameter error.

#### HALT WHEN POWER ON

Reply: Type in Y or N.  
If the reply is Y, the system will, when power is on again after a failure, execute a Halt instruction to give the operator some time to attend to any devices that might need his intervention before the system is started again.  
To select the user routine processing power failure and automatic restart, the user must include it during GENLKE, before entering the standard library. If he does not, the standard routine will be selected.

#### REAL TIME CLOCK LEVEL

Reply: (L)  
where L is the level to which the clock is connected

#### ABS TIME MANAGEMENT NEEDED

Reply: Specify Y if connection to absolute time is wanted, or N if it is not wanted.  
This question will appear only if the answer to the question REAL TIME CLOCK was neither N[0] nor N[1]. If the reply is N, the connection to a timer remains available, as well as the Get Date and Get Time features (monitor requests).

#### PANEL LEVEL

Reply: When operator communication (OCOM) modules are selected, the user must specify the level to which the control panel interrupt is connected.  
The level is specified as:  
<L> a level number of 1 or 2 hexadecimal digits.

Specify N if no operator communication feature is used.

Error Message: TOO3

#### LEM LEVEL

Reply: First specify the level to which the LEM interrupt is connected in the same manner as for the previous question. Then, define the optional monitor requests derived for your system (other

monitor requests provided in the software are standard). DRMGEM will print the names of the optional ones, after which the user may type Y if he wants it to be included or N if he does not.

DRTM lists only two optional functions:  
DFM (Disc File Management) (required for SCL)

Note: The user must not issue a request in his programs for a function he has not included. For example, if he has typed N after DTM at DRMGEM, an I/O monitor request (LKN1) on a logical disc file may cause a system hang-up. Therefore users should define exactly the required functions.

Error Message: T906: the reply is neither N nor Y. Try again.

#### RESIDENT MACRO

Reply: Some of the monitor requests which are standard to the DRTM may be core resident or disc resident. DRGEM prints all the names of these requests one by one and the user replies by typing behind each one a Y if he wants it to be core resident or an N if he wants it to be disc resident.

TARGEM prints the following list:

CTNM (Connect a Timer)  
DTNM (Disconnect a Timer)  
ATDV (Attach a device/file)  
DTDV (Detach a device/file)  
GTIM (Get Timer)  
CNLV (Connect A Level)  
DNLV (Disconnect a Level)  
WFGT (Wait for a Given Time)  
ASSG (Assign a File Code)  
DEFG (Delete a File Code)  
KEYI (Read Unsolicited Key-in)  
CNSL (Cancel Read Unsolicited Key-in)

Error Message: T906: the reply was neither N or Y. Try again.

## USER LKM

Reply: The user may specify his own set of monitor requests for inclusion in the monitor, as follows:

```
<NI>,<ENTRYI>  
<Ni>,<ENTRYI>  
<Nm>,<ENTRYn>
```

END

where NI consists of two hexadecimal digits defining the DATA number which follows the LKM instruction, and ENTRYI is the symbolic entry point of the routine processing the LKM DATA NI function. The system will extend the monitor request table in which word i contains the address of ENTRYI. Therefore, during SYSLINK the user must provide the Linkage Editor with the module containing all entry points ENTRYI specified here. END indicates that all user LKM functions have been declared or that none are wanted. All user LKMs are processed by core resident programs working at level 48.

Note: Ni must not be equal to any of the standard LKM DATA numbers.

Error Message: TGD3: the first parameter is not hexadecimal  
TGD7: the first character of the second parameter is not alphabetic  
TGD8: syntax error

## POT STANDARD LEVELS

Reply: DRMGEN will print a list of system programs, behind each of which the user may type Y or N to indicate whether he wants the standard level and assignments for that program or not. If he types N, he must also type the level and

assignments he wants, as follows:

<level>, [DR | CR]

where level must be a software level between /31  
and /3F exclusive.

DR must be specified if the program should not be  
core resident.

CR must be specified if the program is to be core  
resident.

Note:

The program names printed are:

TIMER (clock management) : M:DCK2 module

LOADER: D:LDER module

PWRLMK (possibly non-resident monitor requests):

D:USV1, D:USV2, D:USV3 and D:RMAG modules.

GR ALLOC (granule allocation): D:ALGR and D:CTFW  
modules

OCOM (control panel interrupt): D:OCOM module

SWAP: D:SWP module

DUMP: D:DUMP module

SCL (system command language segments): D:ROOT,  
D:SEG1, D:SEG2, D:SEG3, D:SEG4

If Y is specified as the reply, DMGEN prints and  
implements the following list:

TIMER : 31 CR

LOADER : 31 CR

PWRLMK : 31 DR

GR ALLOC : 31 CR

OCOM : 31 CR

SWAP : 31 CR

DUMP : 32 DR

SCL : 3D DR

The Idle Task (IDLTSK module) will always be level  
/3F and core resident.

NB OF PROG CONTROL TABLES

Reply:

Specify the number of program control tables which  
must be reserved for your system. For each user

program one entry in the program control table is allocated. Each entry occupies 10 words. The user need not reserve entries for the system programs.

Error Message: T003: invalid reply.

#### DEVICES ON PROGRAMMED CHANNEL

(Note: devices on channels must be declared according to Release Bulletin)

Reply: Specify, which devices are connected to the programmed channel, as follows:  
(DHDA),(L)

  |  
  END

Where:

DH is the device name, in two ASCII characters.  
DA is the device address, in two hexadecimal digits.  
<L> specifies the level to which the device is connected.

END indicates that all devices have been declared.

Note: When the configuration contains only 2 cassette drives (TX), the physical addresses (DA) of these drives must be /0X and /1X, X being the control unit address.  
Note: For the operator's typewriter, three devices(TY, TP, TR) may be declared with the same address, if they are all used.  
No check is made on device declaration.

Error Messages: T001: the device specified is not supported by the system.  
T002: device address error  
T009: level error  
T010: device not declared.

#### Device names used:

TR : ASR tape reader  
TP : ASR tape punch  
PR : high-speed tape reader  
PP : high-speed tape punch

TY : operator's typewriter  
CR : card reader  
LP : line printer  
TK : cassette tape unit  
MT : magnetic tape unit

#### DEVICES ON MULTIPLEX

Reply: Specify, which devices are connected to the multiplex in the same manner as for the devices on the programmed channel.

Note: For line printer there

are additional parameters:

- Line printer:

DEPA,L,LG, number

where LG-[S|L] : S = 80-column printer  
L = 132-column printer.

<number> : a hexadecimal number  
specifying the number  
of lines per page want-  
ed.

No check is made on the device declaration.

#### DISK UNITS

Reply: Specify all disc units connected to the I/O pre-  
cessor as follows:

DNDA,L,Q

RND

where:

DN is the device name, giving the type of disc on  
2 ASCII characters ast

X1: X1215      X2: X1216

D1: CDC 40M    D2: CDC 80M

Note: For the X1215, each disc pack, not  
disc unit must be declared, for there  
are 2 packs to one unit.

DA is the device address, 2 hexa digits (see App. ).

I specifies the interrupt level to which the disc is connected (1 or 2 hexadecimal digits)  
g defines, on 2 hexadecimal digits, the length of the disc queue table, in number of entries

Note:

- The disc queue table is used only if DPM has been selected. If it has not, this parameter must be 0. The number of entries in the disc queue table must be equal, for each disc unit, to the maximum number of disc logical I/O which may be processed simultaneously on the same unit. The answer must therefore be equal to the one given under DISC LOGICAL FILES. given

NBR OF DAD CONTROL TABLES

Reply: A 2-digit hexadecimal number, indicating the number of DAD file codes (from 0F downwards as far as 00) which has to be assigned internally.

TOTAL LENGTH OF BITABS

Reply: A 2-digit hexadecimal number, giving the total length of the required BITAB tables (see Disc Organisation).

SPECIFY FILE CODES

Reply: Specify all file codes used by the programs. If system processors are used, their standard file codes must be declared.

Declaration is done as follows:

(PC);(BMDAD)  
(PC);(BMDAD)

END

where FC is a file code (2 hexadecimal digits) assigned to the device indicated by DN with address DA.

Note:

File codes /EF and 01 are used by the system to read/write to/from the operator's typewriter. It is therefore necessary to declare at least these file codes, if the typewriter is going to be used. Other file codes may be assigned by using the command AS.

Some file codes, i.e. /02, /E0, /F0 to /FF are used by the system, but they have to be assigned, too.

FILE CODED ABC TO USER DEVICES

Reply: The user may declare all file codes assigned to his non-standard devices. The related I/O drivers (including Device Work Tables) and the interrupt routines for these devices must be written by the user and be incorporated in the system during the GESTKE phase. (The entry points for the interrupt routines must be declared under USER INTERRUPT ROUTINES). These file codes must be declared here, as they cannot be assigned by AS operator message or ABC control command. The reply must be as follows:

```
<FC>,<DWTI>
|
END
```

where:

FC is a two-digit hexadecimal file code which will be generated in the File Code Table.

DWTI is the entry point (a name of up to six characters) of the Device Work Table (DWT) associated with this device.

END indicates that all file codes have been declared.

Any number of file codes can be assigned to the same DWTI. The system checks only if the file code is a two-digit hexadecimal number, but not if it has already been used or is one of the standard file codes used by the system processors.

Note

The user devices can be assigned by means of the SCL command AS only with the following syntax:

AS,<file code1>,<file code2>

It is therefore necessary to define <file code2> during DRMCEN for each user device and it must not be reassigned under the DWTI.

Error Message:

- TG03: syntax error, e.g. the file code has not been specified as hexadecimal.
- TG07: the first character of the DWTI is not alphanumeric.

SPARE ENTRIES IN PCT

Reply:

Specify the number of spare entries reserved in the File Code Table (PCT), on one or two hexadecimal digits.

These entries are required for assignment of user file codes at execution time or for assignment of temporary system file codes.

Each entry takes up two words.

DISC LOGICAL FILES

Reply:

This message is printed only in case the user has selected the Disc File Management (DFM) option (see under LEM LEVEL), to find out the number of disc logical files used in system.

Specify two hexadecimal digits, giving the maximum number of logical files which can be assigned (and not yet opened) at the same time.

Each entry takes up 28<sub>10</sub> words, as the system will

reserve a logical File Table for each disc logical file, to store all the information about it.

#### SIMULATED INSTRUCTIONS

Reply: Y or N, depending on whether the simulation package must be included or not. If the reply was Y, the following list is output:

MULTIPLY:  
DIVIDE:  
D ADD:  
D SUB:  
D SHIFT:  
MLR-MSR:

After each item the user must type in Y or N to indicate whether he wants that instruction simulation routine included or not. For P856/857 the answer should be N, for the instructions are included in the instruction set.

#### SIMULATED ROUTINES SAVING AREAS NB

Reply: This question is output only if the reply to the previous question was Y.  
The user must type in the number of save areas required by the simulation package.  
For DRTM, it is the maximum number of programs, main sequences or scheduled labels which might be simulated at the same time.

#### MAX NUMBER OF SCHEDULED LABELS

Reply: Type in a two-digit hexadecimal number, specifying the maximum number of scheduled labels which may be in queue at the same time. This will be the length of the FILTAB table described in the paragraph on Scheduled Labels in Chapter 4.

Note: This is not the maximum number of scheduled labels used in the program, which may be a higher number.

TABGEN ENDED

## PREMK

This is a program which checks the disc on which the generated system must be written for defective tracks, writes sector identifiers on it and an Initial Program Loader (IPL) in the second sector.

PREMK outputs a number of questions on file code /E0 (i.e. TY10) to which the user must type in the replies. Output is done onto file code /F0, i.e. the system disc.

Operation is as follows:

- with paper tape, put the PREMK tape on the reader; make it operable
- push the INT button on the CPU control panel
- on output of M: type in
  - LB
- now PREMK is loaded as the next processor from the cassette GI in drive TK05 or the tape reader and its identification is output:  
IDENT PREMK
- when loading is terminated,
  - :EOS
  - :EOFis output on the typewriter.
- push the INT button
- on output of M: type in
  - ST
- now PREMK is started and the following messages and questions are output on the typewriter:

INITIALISATION OF PREMK

COMPLETE PREMK:      Reply: YY is typed to have PREMK write only the VTOC and label.  
YY if the user wants PREMK to only write the sector identifiers.

CB if the user wants PREMK to write and verify the sector identifiers.

DISK TYPE:      Reply: Enter 2 characters identifying the type of disc to be premarked: X1 for X1215, X2 for X1216, D1 for CDC 40M, D2 for CDC 80M.

LABEL:      Reply: Type in no more than 16 characters for the volume label. If it is less, it will be left-justified and padded with blanks.

DATE: Reply: Type in 6 decimal characters, giving day, month and year.

PACK NBR: Reply: Type in no more than 4 characters for the pack number. If it is less, it will be right-justified and zero-padded.

DISC UNIT PHYSICAL ADDRESS: Reply: Type in 2 hexa characters giving the device address.

DAD NAME: Reply: Type in no more than 6 characters, specifying the name of the first DAD. If it is less, it will be left-justified and padded with blanks.

NBR OF CYL OF ddnname: Reply: Specify the number of cylinders of this DAD. (Max. per disc type: X1215: 200; X1216: 400; CDC40M/80M: 103).

NBR OF INT OF ddnname: (CDC discs only) Reply: Specify the interlace factor of this DAD (not the number 15, no multiples of 5).

SYST. USERID: Reply: Type in the name of the userid; no more than 8 characters. If less, it will be left-justified and filled with blanks.

ANOTHER DAD: Reply: OK if more DADs are to be specified for this disc. The sequence will then restart at DAD NAME.  
NO if this is the only or last DAD.

END OF PREMARK

Error Messages:

- BAD GRANULE 0: the disc pack is not usable
- NO SYSTEM USER POSSIBLE: granule 1 is bad and therefore no system can be stored on this disc pack.

Now the disc is premarked and ready to receive the generated system. But first we have the GENLKE phase, during which the tables generated under DRMGEN are linked with the modules required from the DRM Library and, possibly, user and/or extension libraries to generate the user's monitor.

Note: If sysgen is to be done under control of the Disc Operating Monitor, the user must reload the system by means of IPL procedure after Premark.

### GENLKE

During this phase the final user monitor is obtained by linking the tables generated under DRMCEN with the monitor modules required from the DRM library and/or any user library or extension libraries (see note at the end of this section).

The input to GENLKE is done from file code /4, i.e. cassette or paper tape. The output is done onto file code /3, in standard cases working cassette W1 in drive TK15, but it may also be punched tape or magnetic tape.

In any case, the GENLKE processor must now be loaded:

- when using paper tape, take the PREMDK tape from the reader, put on the GENLKE tape and make the reader operable
- push the INT button on the CPU control panel
- on output of M: type in LD
- now GENLKE is loaded from the cassette tape or the tape reader and its identification is output:  
100111 GENLKE
- when loading is terminated,  
:E08  
:EOF
- is output on the typewriter.

### With\_cassettes

On the basis of the availability of two cassette drives, the cassettes are now handled as follows (with three drives, see below):

- take cassette G1 from drive TK05 after GENLKE has been loaded
- take cassette W1 (containing the tables generated under PRIMER) from drive TK15 and put it into drive TK05 and wait for it to be rewound
- put the second working cassette W2 into drive TK15
  
- now start GENLKE as follows:
- push the INT button
- on output of M: type in  
ST
- GENLKE outputs  
L:
- and the user must type in the link-edit command as follows:

**E[<decimal number>],<module name>[.1[3|4]]**

where <decimal number> consists of three digits:

- the first is the file code for the object output device
- the second is the file code for the listing device
- the third is the file code for the object input device.

<module name> is the name of the user's monitor

3 or 4 is used if the monitor is punched on paper tape to indicate whether it must be punched in 4-track or 8-track format.

If the standard file codes are used (see under GENMON, i.e. /3, /2 and /4 respectively), they need not be specified and the command can be given as:

**E,<module name>**

- then GENLKE outputs

**L:**

to which the user must reply with

**R**

The tables generated during DRMGEN are now recorded from the cassette W1 in drive TK05 onto cassette W2 in drive TK15.

- when this is finished take cassette W1 from drive TK05
- put generation cassette G1 into drive TK05 with side B up (DRMLIB!) and wait for it to be rewound, so that it is positioned correctly for the scanning of the first part of the DRTM Library.

- in response to the

**L:**

output by GENLKE, now type in

**L**

upon which GENLKE will start scanning part 1 of the DRTM Library, select the required modules and record them onto the cassette W2 in drive TK15. The names of the selected modules are output on the listing device, together with their base addresses and any comments included in the identifiers.

- when this is finished, GENLKE again types out

**L:**

The user must now type in

**U**

to check if there are any unsatisfied references. This will appear to be so, for at least IXIMON must remain as an unsolved reference, so:

- GENLKE now lists the references remaining to be solved in the second part of the DRTM Library
- take cassette G1 from drive TK05
- put cassette G1 in drive TK05 with side A up and wait for it to be read
- after the L:  
typed out by GENLKE, the user must now type L upon which GENLKE will start scanning the second part of the DRTM Library, selecting the required modules and recording them on cassette W2 in drive TK15. The names of the selected modules are output on the listing device, together with their base addresses and any comments included in the identifiers.
- when this is finished, GENLKE again types out L:  
The user again types in U to check for unsatisfied references. The last one to be solved must be INIMON, so if GENLKE types INIMON (possibly followed by ASEX, a module referenced by INIMON) after the user has typed in U, it is correct.
- GENLKE then types out L:  
and the user must type L to solve this last unsatisfied reference.  
(If there were more unsatisfied references, the user must repeat this L:L process until INIMON is the last unsolved reference and then give his last L command)
- now, after all modules have been included, GENLKE again types: L:
- the user once more types U to make sure that all references have been solved. Then, on the next L:  
the user types I to indicate the end of the GENLKE phase
- GENLKE then outputs the symbol table of the generated DRTM

on the listing device and on the typewriter it outputs mono-  
to length ( $L = XXXX$ ), monitor start address ( $S = XXXX$ ) and the  
first free location after the monitor in memory ( $E = XXXX$ ).

Note: If the user has three cassette drives and wants to use  
them all during this phase, the procedure is as follows:

- after loading GENLKE, take cassette W1 from drive TK05 and put  
it back in with side B up. Wait for it to be rewound.
- put cassette W1 (tables output by DRNGEN) into drive TK25  
and wait for it to be rewound
- put the second working cassette W2 in drive TK13
- start GENLKE by pushing the INT button and, on output of  
M: typing in ST
- on output of L: type in the option message as follows:  
E:327,<module name>8  
where 3 and 2 are the normal object output and listing file  
codes and /7 is now assigned to the input file code, so not  
/4 = TK05, but /7 = TK25. This is because TK25 contains the  
cassette W1 with the tables generated under DRNGEN. See also  
under GENMON for the file code assignments.
- when this is accepted, GENLKE outputs L: and the user types  
E  
Upon this command, GENLKE scans the input file (i.e. the cas-  
sette W1 with the tables) up to EOF, then goes into Pause state.
- now GENLKE is restarted with an RS command with a new file code,  
switching L: back from /7 to /4, the normal input file code  
and the one assigned to TK05, which contains the DRTM Library  
part 1 which must now be scanned. So:
- push the INT button and on output of M: type  
RS 04
- GENLKE now starts scanning and selecting the required modules  
from the DRTM Library 1 and the rest of the procedure is the same  
as described above, starting after the user's first L command.

With paper\_tape:

- in this case the procedure is basically the same as with  
cassettes (see description above), but input is done from  
the tape reader and output onto tape punch. On the tape  
punch an IPT has already been generated and the paper tape  
should be left as it is.

- first the GENLKE tape must be put on the reader and GENLKE is loaded into memory by LD command. Then the tape containing the tables generated under DRMLIB is put on the reader and GENLKEB is started with an ST command. Having entered the E: option command, the user types P and the tables are processed.
- then the DRMLIB 1 tape is put on the reader and the user types L after which this library is scanned and the selected modules are output on the punched tape.
- if the user types in U, the unsolved references are listed. They should all be contained on the DRMLIB 2 tape. This tape is now put on the reader and also processed with an L command.
- having then checked if INIMON (+ ASEK) is the last unsolved reference with a U command and having typed L to solve it, the user then types I to terminate the process.
- on the tape punch a DRTM has now been generated.

Note:

If modules from other (paper or cassette) tapes beside the DRMLIB tape must be link-edited during this phase, the tapes must be scanned in a defined order, which is:

- User Library tape(s), if any
- Extension tape(s)
- DRMLIB tape

When more references than INIMON remain unsatisfied, rescanning must start at the first step in this sequence.

## DISL0D

DISL0D is the last system generation processor. It is used to record the generated monitor and monitor modules onto the system disc specified under GENMON, in load module format accepted by the DRIM.

The standard object input file code for DISL0D is /E2, (TK05), the output disc file code is F0 (BM02), the listing file code is /2 (LP07) and all conversational processes are done through file code /EF (TY10).

- when working with paper tape, put the DISL0D tape on the tape reader and make it operable
- with cassettes, the cassette tape in file code /E2 is positioned after the DRMLIB 2, i.e. before DISL0D)
- push the INT button on the CPU control panel
- on output of M1 type in LD
- now DISL0D is loaded from the tape reader or the cassette and its identification is output:  
IDENT DISL0D
- when loading is terminated,  
:EOS
- :EOF is output on the typewriter.
- with paper tape, DISL0D tape must now be taken from the reader and the newly generated DRIM tape must be placed on it, ready to be read
- with cassettes, take cassette G2 from file code /E2 (normally TK05) and replace it by cassette W2 (containing the new DRIM) which must first be taken from file code /3 (normally TK15).
- push the INT button
- on output of M2 type  
ST  
to start the DISL0D processor.
- DISL0D then outputs the message  
SYSLOAD XE P852  
on the typewriter, followed by the question  
NEXT ACTION:
- at this point, the user has three possibilities for action:
  - if he types **LP** **CR** the next program or module on the Input file (/E2) will be recorded onto the system disc. DISL0D will output the program name onto the typewriter (PRGM.NAM) = XXXXXX and list name, length in sectors,

start address and program length on the listing device (/2).

After this, DISLOG will again output the message NEXT ACTION:

- if he types PE, DISLOG goes into Pause state, enabling the user to modify an assignment, operate on the cassettes by manual control, etc. To restart DISLOG, the message RS must be typed in (after having pushed the INT button), which may be followed by a parameter containing a new object input file code.
- if he types HT, DISLOG performs an Exit and the process is stopped.
- so, after the first NEXT ACTION: message output by DISLOG, the user types LP CR and his newly generated monitor is recorded from cassette W2 or paper tape onto the system disc specified by the user under GENMON and premarked under PREMDK.
- when this is finished, DISLOG again outputs NEXT ACTION:
- with paper tape, now put the first of the monitor segment and system processor tapes on the reader, i.e. the DISV1 tape (see list of system tapes at beginning of chapter)
- with cassettes, now take cassette W2 out of the drive and replace it by the second generation cassette (G2), with side A up, containing the disc-resident parts of the monitor
  
- now, after each NEXT ACTION: message, the user must type LP CR to include successively the disc-resident monitor parts and the system processors. With cassettes, this requires no further manual operations, but with paper tape, the user must put the following tape in the reader each time before typing in LP CR
- when the last one (IPLGEN) has been recorded onto the system disc and DISLOG again types NEXT ACTION:  
the user must type in  
HT  
to terminate the DISLOG operation.

The user now has a Disc Real Time Monitor on disc.

DISC PREMARK

PREMREK is a stand-alone program to be used for formatting a disc pack before it will actually be used. It divides the disc into sectors, writes identifiers in them and checks for bad tracks.

PREMREK is in absolute format, so it is loaded by IPL.

After it has been loaded, it starts to type out the following questions on the operator's console and the user can type in his answers:

## INITIALISATION OF PREMREK

COMPLETE PREMARK:      Reply: **Y** is typed to have PREMREK write only the VPOG and label.  
**W** if the user wants PREMREK to only write the sector identifiers.  
**Q** if the user wants PREMREK to write and verify the sector identifiers.

DISK TYPE:      Reply: Enter 2 characters identifying the type of disc to be premarked: **X1** for X1215, **X2** for X1216, **B1** for CDC 40M, **B2** for CDC 80M.

LABEL:      Reply: Type in no more than 16 characters for the volume label. If it is less, it will be left-justified and padded with blanks.

DATE:      Reply: Type in 6 decimal characters, giving day, month and year.

PACK NBR:      Reply: Type in no more than 4 characters for the pack number. If it is less, it will be right-justified and zero-padded.

DISC UNIT PHYSICAL ADDRESS:      Reply: Type in 2 hexa characters, giving the device address.

DAD NAME:      Reply: Type in no more than 6 characters,

specifying the name of the first DAD.  
If it is less, it will be left-  
justified and padded with blanks.

NBR OF CYL OF dadname: Reply: Specify the number of cylinders of  
this DAD. (Max. per disc type: X1215:  
200; X1216: 400; CDC40/80M: 103)

NBR OF INT OF dadname: (CDC disc only) Reply: Specify the interlace  
factor of this DAD (not the number 13,  
no multiples of 3).

SYST. USERID: Reply: Type in the name of the userid; no more  
than 8 characters. If less, it will be  
left-justified and filled with blanks.

ANOTHER DAD: Reply: OK if more DADs are to be specified  
for this disc. The sequence will then  
restart at DAD NAME.  
NO if this is the only or last DAD for  
this disc.

END OF PREPARE

Below, device driver information is given for each I/O order, except for the order for the driver for the video display terminal, which are grouped together at the end of this Appendix.

This information applies to the order, which the user must specify in an I/O monitor request.

### **BASIC READ (/01)**

#### **Operator's Typewriter**

All characters are entered on 8 bits until the requested length is reached.

#### **ASR Tape Reader**

All characters are entered on 8 bits. The reader stops one character after an Xoff code has been read.

#### **High-speed Tape Reader**

All characters are entered on 8 bits, without checking or special features, until the requested length is reached.

**Basic Read does not ensure a stop on character.**

#### **Card Reader**

All the words are entered and stored in Hollerith code on 12 bits (4 to 15). In each word the column image is right-justified. The words are stored until the requested length is reached. The length is given in words.

#### **Disc**

With the aid of Data Management all the sector words are entered in the memory buffer.

#### **Magnetic Tape Cassette:**

All Read/Write operations (Basic, Standard, Object) are the same, with the following characteristics:

- maximum record length: 256 characters
- required length: block length.
- effective length: block length (without control character).
- all read/write operations are done on the requested length
- incorrect length after read operation: no error, if requested length is greater than block length and the returned status is correct.
- throughput error or data fault: retry is made automatically, up to five times:
  - after read: backspace - read
  - after write: backspace - erase - write.

#### **Magnetic Tape:**

Same as for cassette tape, with the following differences:

- maximum record length: 4095 characters; minimum 12 characters.
- required length: block length without control word.
- physical block length: required length + 2.
- effective length: block length.
- 12 characters are always transferred, in any case.
- incorrect length: see cassette tape above.

## BASIC WRITE (/05)

### Operator's Typewriter

All characters are output without checking or special features. This order can be used to print something and have the answer on the same line.

### ASR Tape Punch

All characters are output without checking or special features.

### Line Printer

All characters are output without checking. There is no control character.

### High-speed Tape Punch

All characters are output without checking or special features.

### Disc

With the aid of Data Management all the sector words are output onto the disc.

### Cassette and Magnetic Tape

See under Basic Read (/01).

## STANDARD READ (/02)

### Operator's Typewriter

ASCII characters are entered on 8 bits, with the following special features:

- the special characters, coded from /0 to /1F, are ignored.
- code /7F (Rub-out or Delete character) is ignored.
- code /5F (←) can be used to delete the preceding character. If several ← are used consecutively, an equal number of preceding characters will be deleted.
- code /5E (↑) is used to delete the line preceding it, up to the next carriage-return.
- code /0D (carriage return) indicates end of block. It is the last character to be entered. It is not transmitted to the user's buffer.
- code /0A means "line feed".
- code /5C (\) is used as a tabulation symbol (see ECB word 5). If the address of the tabulation table is zero, or if the number of tackets is zero, or if the storage address is greater than the last tacket, the code /5C is stored in the buffer. In other cases, /5C is not stored and replaced by spaces, as indicated by the tackets in the tabulation table.

### ASR Tape Reader

For ASCII characters, the same features apply as for the keyboard: the code for carriage return must be preceded by the code for Xoff.

For object code in 4+4+4+4 tape format, the first character identifies the object format. It must be in the range from /18 to /1F and is converted to a number from /0 to /7 and stored on one character. The second character contains the word-count of the input block, excluding the first word and the checksum. Each punched row (8 bits) entered after this identifier is stored on one half-character up to the checksum. When the checksum has been read, input is stopped. The 8+8 tape format cannot be read on the ASR tape reader. To start the reader, an Xon code is sent by the system before entering the characters.

### High-speed Tape Reader

Same as for the ASR tape reader. In addition: for object code in 8+8 format, the first character, identifying the object code format, must have one of the following values: /10, /11 to /4 or /15 to /17. It is converted to a number from /0 to /7. Each punched row (8 bits) entered after this identifier is stored on one character up to the checksum. The second character is the length of the block, in words, excluding the first word and the checksum.

### Card Reader

All words are read in Hollerith code, on 12 bits, converted and stored in ASCII code, on 8 bits, until the requested length is reached. Words which are not in Hollerith are converted into the ASCII code for /20 and a "data fault" status is returned in the software, status (ECB word 4: bit 13 is 1). There is no special code. However, EOS and EOF marks are detected (bits 14 and 15 in the software status).

### Cassette and Magnetic Tape

See under Basic Read (/01).

## STANDARD WRITE (/06)

### Operator's Typewriter

All characters, except /0 to /1F (special code characters) are output without checking. At the end of a line, a carriage return and line feed are output. The first word in the buffer contains a right-justified control character, as for the line printer (see below). If it equals /30 or /31, it is output as line feed; if it is different, it is not output.

### ASR Tape Punch

Same features as for the keyboard. At the end of a line, the following character sequence is output: LF - Xoff - CR - Rubout.

### High-speed Tape Punch

Same as for ASR tape punch.

### Line Printer

All characters are output without checking, except for the control code. The first word in the buffer contains a right-justified control char. This control code may have one of the following three values:

+ (/2B): print the line without advancing the paper (superposition).

0 (/30): advance two lines before printing.

1 (/31): skip to top of page before printing.

All other control codes are used as normally: advance one line and print. At the end of the buffer, after the requested length, one character must follow to be used by the system for a print code.

If the requested length is more than one line, the system puts a print code after the maximum length and the buffer will be printed on two or more lines.

### Cassette and Magnetic Tape

See under Basic Read (/01).

## OBJECT WRITE 4+4+4 TAPE FORMAT (/07)

### ASR Tape Punch

The first character is output on one row, converted from /0 - /7 to /18 - /1F. Each following character is output on two rows; to avoid special (ASCII) code each row is converted. The second character contains the length of the block in characters, excluding the first character. At the end an 8-bit checksum is performed and punched, followed by an Xoff code.

### High-speed Tape Punch

Same as for ASR tape punch, except that the second character contains the length in words.

## OBJECT WRITE 8+8 TAPE FORMAT (/08)

### High-speed Tape Punch

The standard object code is output in 8+8 format, where the first character is a format character and is output on one row, converted as follows:

/0 → /10

/1 to /4 → /01 to /04

/5 to /7 → /15 to /17

The second character contains the length in words, excluding the first word. An 8-bit checksum is performed and punched.

### Cassette and Magnetic Tape

See under Basic Read (/01).

### READ SECTOR (/11)

#### Flexible Disc

Any sector on a flexible disc can be read at random with this order.

The requested length, as expressed in characters in ECB word 2, determines the record length (1, 2, 3 or 4 sectors). The length must be an even number with a maximum of 512 characters. If an odd number is given ( $2n+1$ ), the driver transmits an even number ( $2n$ ). ECB5 must contain the absolute sector number (0 - 2001) of the sector to be read. If more than one sector must be read (128 requested length 512), only the number of the first sector must be specified.

### WRITE SECTOR (/15)

#### Flexible Disc

With this order any sector on a flexible disc can be written at random.

The requested length, as expressed in characters in ECB word 2, determines the record length (1, 2, 3 or 4 sectors). The length must be an even number with a maximum of 512 characters. If an odd number is given ( $2n+1$ ), the driver transmits an even number ( $2n$ ). ECB5 must contain the absolute sector number (0 - 2001) of the sector to be written. If more than one sector must be written (128 requested length 512), only the number of the first sector must be specified.

### READ UP TO END-OF-SEGMENT (/14)

#### High-speed Tape Reader

The tape is read until an :EOS statement has been read.

#### Card Reader

The cards are read until an :EOS statement has been read.

### READ UP TO END-OF-FILE (/16)

#### High-speed Tape Reader

The tape is read until an :EOF statement has been read.

#### Card Reader

The cards are read until an :EOF statement has been read.

#### **WRITE EOF MARK (/22)**

##### **Operator's Typewriter**

An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

##### **ASR Tape Punch**

An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

##### **High-speed Tape Punch**

An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

##### **Line Printer**

An end-of-file mark is output as follows: :EOF

#### **WRITE END-OF-VOLUME (/24)**

End-of-tape management for Magnetic and Cassette tapes is a user program responsibility.

When the physical end of a tape is encountered during a write operation, a status is returned in ECB word 4 with the EOT bit set. The user may then issue a Write EOF request (/24; Write End-Of-Volume; see under I/O monitor requests), before requesting the operator to mount a new tape. When a new tape is mounted, for magnetic tape the unit must first be switched off by pressing the OFF LINE button, while for cassette tape a Manual Control (MC) operator command 'Unlock' must be given to enable the operator to remove the cassette.

Then the operator can mount a new tape reel or cassette and restart the program. To ensure that all records will be retrieved when the file is read, the EOT (end-of-tape) status also returned in the status word of the ECB should be ignored and only the EOF status must be taken into account.

*Note:*

In case the EOT is detected while reading an EOF, only the EOF status is returned.

#### **WRITE EOS MARK (/26)**

##### **Operator's Typewriter**

An end-of-segment mark is output as follows: :EOS LF Xoff CR Rub-out

##### **ASR Tape Punch**

An end-of-segment mark is output as follows: :EOS LF Xoff CR Rub-out

##### **High-speed Tape Punch**

An end-of-segment mark is output as follows: :EOS LF Xoff CR Rub-out

##### **Line Printer**

An end-of-segment mark is output as follows: :EOS

##### **Magnetic Tape Cassette**

An end-of-segment mark is written as /6F6F.

##### **Magnetic Tape**

An end-of-segment mark is written as :EOS + 8 blank characters.

DOOR UNLOCK (/2D)

Flexible Disc

With this order the door of the flexible disc drive unit identified in ECB word 0 is unlocked. The other words in the ECB must be filled with dummy parameters:

ECB0: file code

ECB1: dummy value

ECB2: dummy value

ECB3: updated by the flexible disc driver

ECB4: updated by the flexible disc driver

ECB5: dummy value

DOOR LOCK (/2E)

Flexible Disc

With this order the door of the flexible disc drive unit identified in ECB word 0 is locked. The other words in the ECB must be filled with dummy values:

ECB0: file code

ECB1: dummy value

ECB2: dummy value

ECB3: updated by the flexible disc driver

ECB4: updated by the flexible disc driver

ECB5: dummy value

WRITE DELETED DATA ADDRESS MARK (/2F)

Flexible Disc

With this order a special mark, providing IBM compatibility, is written on the sector. The length of the record on which the deleted data address mark is written is that specified as the requested length in the ECB, which must be expressed in characters.

The buffer of the calling program must contain the information to be written on the sector(s) with a deleted data address mark. This information may be anything the user chooses to write.

ECB5 must contain the number of the sector in which the deleted data address mark is to be written.

If the deleted data address mark is to be written in more than one sector, only the number of the first sector is to be given. See also order /3D.

#### RETURN INFORMATION ABOUT A FILE CODE (/38)

By means of this order it is possible to find out the assignment of a file code. The information will be returned in the Event Control Block:

ECB - word 0: File Code

ECB - word 1: Device Name (2 ASCII characters):

TY = operator's typewriter (listing)  
TR = ASR tape reader  
TP = ASR tape punch  
PR = tape reader  
PP = tape punch  
LP = line printer  
CR = card reader  
MT = magnetic tape  
DK = disc  
TK = cassette tape

If NO device is assigned, the ECB contents are set to zero.

ECB - word 2: maximum record size.

ECB - word 3: left character unused.

ECB - word 3: right character device address.

ECB - word 4: status = 0. For line printer, it contains the number of lines specified for this printer at system generation time.

If this order is used for a logical disc file code, the system will return the device name DL in ECB word 1, and word LFTMD1 from the Logical File Description Table (LFT) in ECB word 4 and fill the other words of the ECB with zeroes. LFTMD1 contains the following information:

A P S O U C T R Se Fm E W

--	--	--	--	--	--	--	--	--	--	--	--	--

A=0: LFT busy	Fm=1: file closed (file mark written)
1: LFT not active	E=1: current entry already in queue
P=1: file write-protected	W=0: write request
S=1: source file	1: read request
O=1: object file	
U=1: undefined file	
C=1: load module	
T=1: temporary file	
R=1: random access	
Se=1: sequential access	

#### OFF LINE (/38)

##### Magnetic tape:

This order switches the machine off.

##### Cassette Tape:

This order unlocks the cassette from the drive unit.

### COMPOUND READ (/3A)

#### Flexible Disc

With this order one full flexible disc track (26 sectors = 3528 characters) is read into the buffer in one revolution of the disc. This function is performed correctly only if the flexible disc unit is connected to the I/O Processor. If it is connected to Programmed Channel, the order is also accepted and executed, but then more than one revolution of the disc is required.

ECB1 must contain the buffer address.

ECB2 must contain the requested length: any even-numbered value from 2 to 3528. If an odd number is given ( $2n+1$ ), the driver transmits an even number of characters ( $2n$ ).

ECB5 gives the absolute sector number from which the read operation must be started. There is no automatic chaining of sectors on the next track.

### COMPOUND WRITE (/3B)

#### Flexible Disc

With this order one full track (26 sectors = 3528 characters) is written onto the flexible disc, without verification.

This function is performed correctly only if the flexible disc unit is connected to the I/O Processor. If it is connected to Programmed Channel, the order is also accepted and executed, but then more than one revolution of the disc is required.

ECB1 must contain the buffer address.

ECB2 must contain the requested length: any even-numbered value from 2 to 3528. If an odd number is given ( $2n+1$ ), the driver transmits an even number of characters ( $2n$ ).

ECB5 gives the absolute sector number from which the write operation must be started. There is no automatic chaining of sectors on the next track.

### SEARCH KEY WITH MASK (/3C)

#### Flexible Disc

With this order a given character configuration can be searched for anywhere in a sector.

This is done by introducing dummy characters to mask the characters preceding the key inside the sector in which it is supposed to be located.

The dummy characters are given as blanks and are not significant in the search operation.

To set up the mask and indicate in which sector the search must be done, a searched key block must be made, the address of which is to be put in ECB1.

For example, if the key (character configuration) to be searched for begins at the 6th character in a sector, the preceding 5 characters should be given as blanks:

	1		
2	3		
4			
sector number 1	}	boundary words	
sector number 2			

The boundary words contain the absolute sector numbers of the sectors between which the sector with the searched key is expected to be found.

ECB2 specifies the length of the character string (dummy characters plus key characters) which must be scanned for the key, excluding boundary words (maximum length: 128 characters, minimum length: 1 character).

ECB5 is not significant, but on completion of the search, if the key has been found, ECB5 contains the address of the sector containing the key.

The layout of the searched key block is as follows:

- for even number of characters:

A	B
ω	C
D	ω
E	F
G	H
sector number 1	
sector number 2	

}

length of character string is 10 (including 2 dummy characters)

boundary words

- for odd number of characters:

ω	ω
ω	A
K	P
H	████████
sector number 1	
sector number 2	

}

length of character string is 7 (including 3 dummy characters)

boundary words

#### WRITE DELETED DATA ADDRESS MARK AND VERIFY (/3D)

##### Flexible Disc

With this order a special mark, providing IBM compatibility, is written on the sector. The length of the record on which the deleted data address mark is written is that specified as the requested length in the ECB, which must be expressed in characters.

The buffer of the calling program must contain the information to be written on the sector(s) with a deleted data address mark. ECB5 must contain the number of the sector in which the deleted data address mark is to be written.

If the deleted data address mark is to be written on more than one sector, only the number of the first sector is to be given. After the record with the deleted data address mark is written, the control unit will use the next revolution of the disc to read and verify the mark.

The Deleted Data Address Mark may contain any information. It is a hardware recognized character implemented for IBM compatibility.

### SEARCH KEY (/3E)

#### Flexible Disc

With this order a given character configuration is searched for within a sector, starting from the beginning of the sector. A searched key block must be made, containing the searched key (character configuration) and the numbers of the sectors between which the sector with the searched key is expected to be found:

1	2	{	length of character
3	4		string is 5
5			
sector number 1			
sector number 2			boundary words

The address of this block must be given in ECB word 1. ECB2 specifies the length of the character string to be searched for, boundary words excluded. Maximum length is 128 characters, minimum length is 1 character.

ECB5 is not significant, but on successful completion of the search this word will contain the address of the sector containing the key.

### WRITE SECTOR AND VERIFY (/3F)

#### Flexible Disc

With this order any sector on a flexible disc can be written at random. After the record is written, the Flexible Disc Control Unit uses the following disc revolution to read and verify the written data.

The requested length, as expressed in characters in ECB2 determines the record length (1, 2, 3 or 4 sectors). The length must be an even number with a maximum of 512 characters. If an odd number is given ( $2n+1$ ), the driver transmits an even number of characters ( $2n$ ). ECB5 must contain the absolute sector number (0 - 2001) of the sector to be written. If more than one sector is to be written (128 requested length 512), only the number of the first sector must be given.

The P817 driver provides the interface between the DRTM and a P817 video display terminal, connected to the system via a data communication control unit ( AMASA, AMAS8, ALCU2 or ALCU4 ). The driver must be regarded as a user-written driver, i.e. it must be linked to the monitor as user interrupt routine during system generation.

P817 VIDEO DISPLAY TERMINAL

The P817 video display terminal is available in two types: the P817-001, which is TTY compatible, operates in character mode, and has a 64 character set ( upper case only), and the P817-002, which can operate in character mode or block mode, and has a 96 character set ( upper/lower case). Both types have a 12 inch screen, providing 24 lines of 80 characters.

The P817-001 can be extended with options to provide a 96 character set ( upper/lower case), display attributes ( blinking, blanking, underlining and low intensity, and hard copy facility via an auxiliary device. For the P817-002 options are available for display attributes, hard copy via an auxiliary device, and split screen operation.

The options, including the roll-up function can be selected by hardware straps; in P817-002 terminals with split screen option some strap selection can be changed via the software.

Character/Block Mode

In character mode, each character is transmitted to the system when it is entered on the keyboard. In block mode, the terminal transmits the line in which the cursor is present when the ETB key is pressed, or all lines up to and including the one in which the cursor is present, when the ETB key is pressed together with the Shift key.

Display Attributes

Characters can be displayed with or without display attributes. The attributes are: blinking (characters are displayed/not displayed at a rate of 1.5 Hz), blanking (characters are replaced by a blank on the screen, e.g. passwords), underlining (a continuous line underneath the characters), and low intensity (characters displayed at 2/3

of normal intensity). In PS17-002 terminals, characters displayed with the low intensity attribute are also protected against over-writing.

#### Split Screen Operation

In PS17-002 terminals with split screen option, the screen area can be divided into two independent parts; the number of lines for the upper screen part can be specified by the user. All control functions not specifying an explicit line number, apply to the screen part in which the cursor is present; control functions specifying an explicit line number apply to that line.

#### Hard Copy Facility

If a hard copy device is connected to the display terminal, one or more lines from the screen can be printed. The user can request a copy of the line in which the cursor is present, or a copy of the screen image from the first character position on the first line, up to the current cursor position.

#### I/O REQUESTS

A user program can perform I/O operations via the monitor by means of IKM-requests. Data transfer requests take the same form as the standard I/O-requests for other devices; for special PS17 functions the function code and parameters are specified in a separate Function Control Block.

#### Calling Sequence

The calling sequence for all PS17 functions is

LDK	A7, CODE
LDKL	A8, ECBADR
IKM	
DATA	1

Register A7 is loaded with CODE, specifying details of the I/O function as follows:

bit	7	8	9	10	15
	S	W	R		ORDER

S, W and R specify the mode of operation:

S=1: (W must also be 1): used by a swappable program; this program can then be swapped out immediately when its swap event count value becomes zero.

W=1: the requesting program waits for the completion of the I/O operation

W=0: the requesting program does not wait for completion of the I/O operation; the program will issue a Wait request later on, for synchronization

R=1: the program itself will process any abnormal condition concerning the requested operation. The system will return the hardware status in ECB word 4 (see below). No retry is possible.

R=0: any abnormal conditions will be processed by the system. The software status is returned in ECB word 4.

ORDER specifies which I/O function is required, by giving one of the following values:

/01: receive (read); characters are read until the required length is reached (see description of ECB below)

/05: send (write); characters are sent until the required length is reached

/12: disconnect; the terminal is disconnected from the line, the screen is cleared, and the cursor is put in the 'home' position

/13: connect; the terminal is connected to the line, the screen is cleared, and the cursor is put in the 'home' position

/1A: wait for incoming call; if the terminal is connected to a switched line, the program is put in wait state until data are received from the terminal; if the terminal is not connected to a switched line, this order is treated as an illegal order.

ECBADR is loaded into register A8, specifying the address of the Event Block.

### Event Control Block

The layout of the ECB is:

Y/X	EVENT CHARACTER	LINE CODE	WORD 0
X	BUFFER ADDRESS		WORD 1
X	REQUIRED LENGTH		WORD 2
Y	EFFECTIVE LENGTH		WORD 3
Y	STATUS WORD		WORD 4
X	TIME OUT VALUE		WORD 5
X	FCB ADDRESS		WORD 6

X: these words must be filled by the user

Y: these words are filled by the monitor

WORD 0: event character / line code

bit 0 : if '1', end of operation has occurred for this ECB

bit 1-7 : not used

bit 8-15: line code (file code of the required line)

WORD 1: address of the I/O buffer area

WORD 2: required length, in characters

WORD 3: number of characters which have been transferred

WORD 4: status word

bit 0 = 0 no error

= 1 error

bit 1 = 0 hardware status in bit 2-15

= 1 software status in bit 2-15

hardware status:

bit 2-4 not used

bit 5-7 line number (AMA 8 only)

bit 8 break detection

bit 9 calling indicator

bit 10 end of carrier detection

bit 11 time out error, or I/O command not accepted

bit 12 time out elapsed

bit 13 parity error

```
bit 14 throughput error
bit 15 modem not operable

software status:
bit 2-6 not used
bit 7 illegal FCB code
bit 8 device busy
bit 9 address not required
bit 10 illegal function specification
bit 11 illegal order
bit 12 illegal buffer size
bit 13 illegal buffer address
bit 14 device not attached
bit 15 illegal file code
```

WORD 5: time out value

the time out value specifies, in units of 0.1 seconds, the maximum waiting time between the I/O request and the setting of bit 0 in the event character. If the time out value is < 0, the monitor does not check the time out.

WORD 6: FCB (Function Control Block) address

Function Control Block

The Function Control Block is a 4-word block, containing a function control code and (where applicable) function parameters. The layout of the FCB is:

WORD 0	function code
WORD 1	parameter 1
WORD 2	parameter 2
WORD 3	parameter 3

WORD 0 : function code (hexadecimal)

if =0, a normal write operation (order /05) is carried out

WORD 1-3: function parameters, only used if applicable for the specified function. ( see note on next page )

Note: Function parameters must be specified in ASCII code (e.g.  
decimal 12 must be specified as /3132, not as /C)

FUNCTION CODES

The table below shows for all PS17 function codes:

- function name
- function code
- number of parameters (indicated by 'X', or by the fixed parameter value)
- the PS17 type for which the function is available

For all function codes, order /05 (send) must be specified in the I/O request.

function name	code	p1	p2	p3	PS17 -001	PS17 -002
bell	07				x	x
backspace	08				x	x
line feed	0A				x	x
cursor home	0B				x	x
clear screen	0C				x	x
cursor return	0D				x	x
disable manual input	60				x	x
enable manual input	62				x	x
media copy page	69	0			x	x
media copy line	69	6			x	x
set attributes	30	x			x	x
reset attributes	31	x			x	x
cursor position	48	x	x		x	x
cursor horizontal tabulation	49	x				x
cursor horizontal back tabulation	61	x				x
cursor forward	43	x				x
cursor backward	44	x				x

function name	code	p1	p2	p3	PS17-001	PS17-002
cursor up	41	x				x
cursor down	42	x				x
insert line	40	x				x
delete line	43	x				x
erase	44	x				x
fast output	70	x				x
write line	71	x	x	x		x
read line	72	x	x	x		x
write straps	34	x	x	x		x
set indicator	32	x				x
reset indicator	33	x				x

#### Function Description

The description of the PS17 control functions is divided in two parts: the functions available for PS17-001 and PS17-002, and the functions available for PS17-002 only. With each function, the relevant words of the Function Control Block are given.

#### Common Functions PS17-001/002

FCB                   function name / function description

**/07**

**BELL**

The buzzer in the display unit sounds for 0.1 seconds

**/08**

**BACKSPACE**

The cursor is moved one position to the left. If the cursor already is on the first character position of the first line, nothing happens; if the cursor was at the first position of any other line, it is moved to last position on the previous line.

/01

LINE FEED

The cursor is moved to the same character position on the next line. If the cursor already is on the bottom line and the roll up function is not wired, nothing happens.

If the roll up function is available, all lines move one position up, and the bottom line is filled with null-characters.

/02

CURSOR HOME

The cursor is moved to the first character position on the top line.

/03

CLEAR SCREEN

The screen image is cleared, and the cursor is moved to the cursor home position. The Keyboard becomes unlocked (see function ENI below).

/04

CURSOR RETURN

The cursor is moved to the first character position of the current line.

/05

DISABLE MANUAL INPUT

The keyboard is locked, and all keyboard input is ignored (except for the Clear Screen Key). If a key is pressed, the buzzer sounds for 0.1 second.

/06

ENABLE MANUAL INPUT

The keyboard is unlocked, and the Disable Manual Input state is reset.

/69
/3330

#### MEDIA COPY (PAGE)

The contents of the screen, from the first character on the top line up to and including the current cursor position, are copied on the auxiliary device. At the beginning of each line 'Carriage Return/line Feed' is generated for the auxiliary device.

/69
/3336

#### MEDIA COPY (LINE)

The line on which the cursor is present is copied on the auxiliary device, from the first character position of the line up to and including the current cursor position. At the beginning of each line 'Carriage Return/Line Feed' is generated for the auxiliary device.

Note: After the execution of the Media Copy function, the terminal returns '0' to the first parameter word if the copy is ready, or '3' if the copy is not ready (e.g. device inoperable).

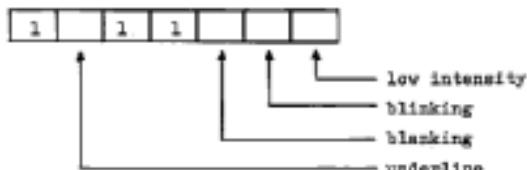
/30
/XX

#### SET ATTRIBUTES

The SAT function sets the format attributes, which control the way in which data are displayed on the screen. The format attributes are:

underline	- a continuous line under all characters
blinking	- characters displayed/not displayed at a rate of ca. 1.5 Hz
low intensity	- characters are displayed at about 2/3 of normal intensity (in P617-002 terminals operating in block mode, these characters are also protected against overwriting from the keyboard, and are not transmitted after a transmit command)
blanking	- characters are stored in the screen memory, but not displayed on the screen

The required attributes are specified by /XX as the hexadecimal value of the 7-bit character shown below:



A '0' bit sets the required attribute.

/31
/XX

#### RESET ATTRIBUTES

The RAT function resets the format attributes, specified by '0' in /XX in the same way as described for the SAT function.

Note: all format attributes are reset automatically for null codes resulting from the Clear Screen, Insert Line and Delete Line function or keyboard entry, and for the line filled with null codes, which appears on the screen after a roll up.

/48
P
n

#### CURSOR POSITION

The cursor is moved to the p-th character position on the n-th line. If p or n exceeds its maximum value (i.e. p>80 or n>24), it is assumed to be 0.

Control Functions for P617-002

The following control functions are available only for P617-002 terminals:

/49
P

**CURSOR HORIZONTAL TABULATION**

The cursor is moved to the first character position of the p-th following unprotected field (default value if p is not specified: p=1), or to the last character position on the bottom line, whichever comes first.

If p=0, the function request is ignored.

/6A
P

**CURSOR HORIZONTAL BACK TABULATION**

If p=1, the cursor is moved to the first position of the current unprotected field (or the first position in the preceding unprotected field if it already was in the first position of the current unprotected field).

If p>1, the cursor is moved to the first position of the p-th preceding unprotected field (default value: p=1), or to the home position, whichever comes first.

If p=0, the function request is ignored.

/45
P

**CURSOR FORWARD**

The cursor is moved to the p-th next character position, or to the end of the line, whichever comes first.

Default value p=1; if p=0 the function request is ignored.

/44
P

**CURSOR BACK**

The cursor is moved to the p-th preceding character position, or to the first character position of the line, whichever comes first.

Default value  $p=1$ . If  $p=0$ , the function request is ignored.

/41

p

#### CURSOR UP

The cursor is moved  $p$  lines up in the same character position, or to the same character position on the top line, whichever comes first.

Default value  $p=1$ . If  $p=0$ , the function request is ignored.

/42

p

#### CURSOR DOWN

The cursor is moved  $p$  lines down in the same character position.

If the  $p$ -th line is beyond the bottom line, the cursor remains on the bottom line; if the roll up function is strapped, roll ups are executed until the  $p$ -th line becomes the bottom line.

Default value  $p=1$ ; if  $p=0$  the function request is ignored.

/40

p

#### INSERT LINE

$p$  lines are inserted at the line on which the cursor is present. The contents of the current line and all following lines are shifted  $p$  lines down, and the lower  $p$  lines are lost.

Default value  $p=1$ ; if  $p=0$  the function request is ignored.

/4D

p

#### DELETE LINE

The  $p$  lines from the current cursor position down are deleted; all following lines move  $p$  lines up, and  $p$  lines with null-characters are added at the bottom of the screen.

Default value  $p=1$ ; if  $p=0$  the function request is ignored.

/70
p

#### PART OUTPUT

The first character from the I/O buffer is repeated p times from the current cursor position, or up to the end of the line, whichever comes first. The cursor remains in its old position. If the specified character is non-displayable (hexadecimal value </20 or >/7F) p white squares are displayed.  
Note: for this function, required length -1 must be specified in the HCR.

/71
p
n
x

#### WRITE LINE

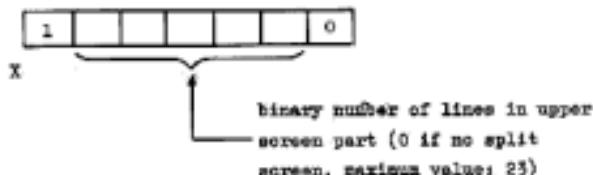
The string of x characters from the I/O buffer is written on line n, starting from character position p.

If the string is longer than the available line length, all remaining characters are written on the last character position of the line.  
 If x=0, it is assumed to be 1. If one of the parameters p or n is not specified, specified as 0, or too high (p>80 or n>24), it is assumed to be 1.  
 All non-displayable characters in the string are replaced by white squares.

/34
x
y

#### WRITE STRAPS

The WST function is used to change the strap-selectable split screen function. The new function specification is contained in hexadecimal code in the characters x and y, of which the 7-bit binary value is shown below:



1 1 0 0 0

Y  (1-character mode, 0-block mode)  
 roll up for lower screen part  
 roll up for upper screen part  
 (1-roll up, 0-no roll up)

If X or Y is specified with an illegal value, the function request is ignored.

132

### BEST INDICATOR

The indicators specified by  $x$  are lit. The binary format of  $x$  is

1	1					
---	---	--	--	--	--	--

The diagram illustrates a logic circuit where five indicators (labeled I1 through I5) are connected in parallel to a single output line. Each indicator is represented by a vertical line with a horizontal bar at the top, and an arrow points from the indicator to the common output line.

A 101 bit specifies that the indicator must be lit.

33

#### RECENT INDICATORS

The indicators, specified by a '0' bit in  $x$ , in the format described for the SIND function are extinguished.

744

**ERASE IN DISPLAY**

All unprotected characters in the screen area specified by p are replaced by null-characters.  
p=0 specifies the screen area from the current cursor position to the end of the screen.  
p=1 specifies the screen area from the first character on the top line up to the current cursor position.

$p=2$  specifies the complete screen.

If  $p=0$  the cursor remains in the current unprotected field, or the cursor is moved out of the current protected field to the first position in the first unprotected field.

If  $p=1$  or  $p=2$  the cursor is moved to the home position. If the home position is in a protected field, the cursor is moved to the first position in the next unprotected field.

#### KEYBOARD FUNCTIONS

##### Transmit Line / Transmit Page

In PS17-002 terminals, the ETB key is used to transmit a block of characters. If the ETB key is pressed without the CNTL key, all characters from all unprotected fields in the line where the cursor is present are transmitted, up to the current cursor position. The last character of each unprotected field is followed by code /09, the complete line is followed by code /19.

If the ETB key is pressed together with the CNTL key, all characters from all unprotected fields, from the first character position on the first line, up to the current cursor position, are transmitted.

However, if the ETB key is pressed when the cursor is on the last position of a line or an unprotected field, and the character in that position is the last typed in character, that character is also transmitted. The last character of each unprotected field is followed by code /09, each line is followed by the codes for Cursor Return (/0D) and Line Feed (/0A), and the completely transmitted page is followed by the code /19.

To receive a block of characters, the user program must issue an I/O request with order /01 (receive). The number of characters in the block must be equal to the required length specified in the ECB.

##### Program Keys

If one of the program keys on the keyboard of the PS17-002 is pressed, a program key report is sent to PCB parameter word 1, with value /303X if key X is pressed without the CNTL key, or with value /313X if key X is pressed together with the CNTL key.

**Appendix D****Control Unit Status Word Configuration**

Bit	Description	ASR	PTR	PTP	S.CU	CR	LP	CAS	MT	DK1215	DK40MB	DKFL	SCUZ	SALCU	HDLC
15	Not operable	X	X	X	X	X	X	X	X	X	X	X	X	X	X
14	Throughput error	X	X		Xin	X		X	X	X	X			X	X
13	Parity error				X			X			X		X	X	
	Data fault								X	X			X		X
	FCS error														
12	Incorrect length					X		X	X	X	X	X			X
11	'Break detection'						X								
	Program error							X	X	X			X		
	End of tape		X					X	X						
	Tape low			X											
	End of reception				X										
10	Hopp./Stack, empty/full					X									
	End of carr. detection												X	X	
	Retry											X			
9	Calling indicator													X	X
	Dev/tape/drive no.							X	X	X	X	X			
	Dev/tape/drive no.							X	X	X			X		
8	Device ready			X											
	Break detection												X	X	
7	1=A side/0=B side						X								
	4-7 No. of true data bits														X
	Write protected					X	X					X			
6	Seek error								X	X					
	4-7 No. of true data bits														X
	Begin of tape/load point						X	X							
5	Seek completed									X					
	4-7 No. of true data bits														X
	Record not found							X			X	X			
4	Del. data addr, mark											X			
	4-7 No. of true data bits														X
	Tape mark detection							X							
3	File mark detection								X						
	Idle detect														X
	Rewind								X						
	Abort detect														X
2	Flag bad data track										X				
	Key not found											X			
1	Unit rdy after not rdy							X	X	X	X	X			
0															

ASM 04

IDENT BEB010

00000

00001

00002

00003

00004

00005

00006

00007

00008

00009

00010

00011

00012

00013

00014

00015

00016

00017

00018

00019

00020

00021

00022

00023

00024

00025

00026

00027

00028

00029

00030

MEANING OF THE KEYS DISPLAYED ON THE CONTROL PANEL

\*\*\*\*\*

BIT 0 = 0 : CHARACTER EXCHANGE ON PROGRAMMED CHANNEL.

1 : WORD

1 = 0 : IPL FROM A NON-DISK DEVICE.

1 : IPL FROM DISK.

2 = 0 : FIXED HEADS DISK.

1 : MOVING HEADS DISK.

3 = 0 : DEVICE ON I/O PROCESSOR (IX).

1 : DEVICE ON PROGRAMMED CHANNEL (CP).

4 TO 7 = BCD LINES USED TO QUALIFY AN INSTRUCTION I/O.

8 = 0 : SINGLE DEVICE CONTROLLER.

1 : MULTIPLEXED DEVICE CONTROLLER.

9 = 0 : NON X1215-DISK USED.

1 : X1215-DISK USED.

10 TO 15 = DEVICE ADDRESS.

WARNING : NOTE THAT THE BIT 0 AND 3 ARE COMPLEMENTARY.

X1215 Cardridge 0110 0011 1100 0010

fixed port 0110 0011 1100 0010

Sector 4 disk

(phys sect 3)

EFFECT

00011  
00012  
00013  
00014  
00015  
00016  
00017  
00018  
00019  
00020  
00021  
00022  
00023  
00024  
00025  
00026  
00027  
00028  
00029  
00030  
00031  
00032  
00033  
00034  
00035  
00036  
00037  
00038  
00039  
00040  
00041  
00042  
00043  
00044  
00045  
00046  
00047  
00048  
00049  
00050  
00051  
00052  
00053  
00054  
00055

BOOTSTRAP FUNCTION DESCRIPTION 77/06/24

\*\*\*\*\*

THIS NEW BOOTSTRAP ALLOWS THE IPL LOADING PROCEDURE  
TO DIFFERENTLY PROGRAMMED CHANNEL OR I/O PROCESSOR  
FOR ANY DEVICES' EVEN FLOPPY.

IF THE CONCERNED DEVICE IS NOT READY WHEN PRESSING IPL  
BUTTON, REBOOT LOOPS UNTIL THE DEVICE BECOMES READY.

Loop Mask 1    " NOT OPERABLE.  
                  " THROUGHPUT ERROR.

                  " DATA FAULT.  
                  " DEVICE HAS BECOME READY.

REBOOT LOADS ONE RECORD ONTO LOCATION /80 THEN START AT /84.

THE RECORD IS THE SECTOR NUMBER 1 ON DISK, OR 256 CHARACTERS  
FROM THE INPUT DEVICE (LEADING NULL CHAR. IS IGNORED) OR SECTOR  
NUMBER 0 IF FLOPPY DEVICE USED.

EFFECT		USED, REGISTERS,		CONTROLS	
*****	*****	*****	*****	*****	*****
000056	000057	A1	E	BU1 LINES,	
000057	000058	A2	E	AT(0 INST),	
000058	000059	A3	E	AC(0 INST),	
000059	000060	A4	E	AS(0 INST),	
000060	000061	A5	E	IP(1 WORD TO BE SENT TO EXTERNAL REGISTER.	
000061	000062	A5	E	IP(1 WORD COUNT),	
000062	000063	A5	E	IP(1 CHARACTER COUNT),	
000063	000064	A6	E	IP(1 WORD FOR "THE MULTIPLEX",	
000064	000065	A6	E	IP(1 WORD FOR "A NEXT CHAR. TO BE LOADED"),	
000065	000066	A7	E	IP(1 WORD FOR "A NEXT CHAR. TO BE LOADED"),	
000066	000067	A9	E	WORK REGISTERS.	
000067	000068	A10	E	NOT USED.	
000068	000069	A15	E	CONTAINS THE KEYS VALUE FROM THE CONTROL PANNEL.	
000069	000070				
000070	000071				
000071	000072				
000072	000073				
000073	000074				
000074	000075				
000075	000076				
000076	000077				
000077	000078				
000078	000079				
000079	000080				
000080	000081				
000081	000082				
000082	000083				
000083	000084				
000084	000085				

```

00016          EFFECT
00017          ADRG  0
00018          * BASEI REGISTERS INITIALIZATION.
00019          * A2 = A1INR INST1;
00020          * A3 = A1CD INST1;
00021          * A4 = A1ST INST1;
00022          * A5 = A1ST INST1;
00023          * A6 = A1ST INST1;
00024          * A7 = A1ST INST1;
00025          * GET DEVICE ADDRESS TO INITIALIZE THE I/O COMMANDS.
00026          0006  861E          LDR    A6,15  46 = KEY VALUE.
00027          0007  861F          LDR    A6,15P.  KEEP THE DEVICE ADDRESS.
00028          0008  261F          LDR    A6,12P.  INIT IAR INST WITH D,A.
00029          0009  8004  AE29          DRS    A6,12  INIT CIO INST,.....,.....
00100          000C  AE20          DRS    A6,13  INIT M0.
00101          900E  AE21          DRS    A6,14  INIT M1.
00102          0010  0000  F          DRS    A6,15  INIT M0. .....
00103          * GET CONTROLLER TYPE AND ADDRESS.
00104          0012  A71E          LDR    A7,15  MULTI OR SINGLE.
00105          0014  3FC6          LDR    A7,16  DEVICE CONTROLLED?
00106          0016  5600  F          SLL    A7,16  IF SINGLE --> BRANCH.
00107          0018  260F          LDR    A7,20  KEEP THE DEVICE CONN. ADDRESS (CA).
00108          001B  0018          ANK    A6,17P.
00109          * INITIALIZATE THE MER INSTRUCTIONS AND SST ONE.
00110          * INIT SST INST WITH D,A OR C,A
00111          0011  AE31          INIT20 DRS    A6,14  COMPUTE EXTERNAL REGISTER ADDRESS.
00112          0012  AE31          INIT20 SLL    A6,15  UPDATE MER INSTRUCTION
00113          001C  3E01          SLL    A6,16  WITH THIS EXTERNAL REG.
00114          001E  AE81          DRS    A6,17R
00115          0022  AE81          DRS    A6,18R2  WITH THIS EXTERNAL REG.
00116          0024  0000  F          LDR    A1,17  A1 = BOU LINES.
00117          0026  811C          LDR    A5,17  BLOCK LENGTH TO BE LOADED.
00118          0028  0590          LDR    A6,18  IN LOCATION /#0.....,.....
00119          002A  0680          LDR    A6,19
00120          * CHECK DEVICE TYPE.
00121          * SIC
00122          002C  3F67          SIC    A7,7  IS THE DEVICE A DISK OR NOT ?
00123          002E  5600  F          SIC    A7,6  NO.
00124          0030  3FC1          SIC    A7,1  IS IT A FIXED HEADS DISK ?
00125          0032  5600  F          SIC    A7,1  YES (NO SEEK TO ZERO).
00126          0034  0103          RRF6  A1,2  EXECUTE A SEEK TO ZERO OPERATION
00127          0036  41C0          CIO    A1,2  ON MOVING HEADS DISK.

```



```

00175:          * EXECUTE A SST AND ANALYZE THE STATUS TO LOOP P12 : A READY IS SET.
00176          * SEND STATUS.
00177          0072 4F10  SST  A10  REFUSED I THIS ALLOW TO HANDLE THE
00178          0078 5C68  RB(8)  MX11CP  BOTH CASES (MAY ON CP OR MX).
00179          0079 5C68  *          BOTH CASES (MAY ON CP OR MX).
00180          0074 5C68  *          BOTH CASES (MAY ON CP OR MX).
00181          0081  ANKU  A17/A007  TEST & BIT 1 = DEVICE HAS BECOME READY.
00182          0076 A700  *          13 = DATA FAULT.
00183          0075 A007  *          14 = THROUGHPUT ERROR.
00184          0084  *          15 = NOT OPERABLE.
00185          0085  *          ONE OF THESE CONDITIONS THEN
00186          0074  0C00  *          START BOOTSTRAP AGAIN FROM THE
00187          0087  *          VERY BEGINNING.
00188          0088  007C  0F44  *          16 = NORMAL END BRANCH TO IPL ROOT.
00189          0090  *          END
00190          0090  *          HONT

```

SYMBOL TABLE

3001	0000 A	INR	0050 A	C10	0036 A	SST	0072 A
410	0070 A	INT120	0014 A	HER1	0042 A	WORD	0044 A
403E9K	0038 A	MX11CP	0041 A	IRQ10	0060 A	STORE	0066 A

ASS.ERK: 00000

1EOF  
PROG ELAPSED TIME: 00H-00M-00S-000ME=

Command	Meaning	Page
AS	Assign a file code	1-96
BF	Space file backwards	1-99
BR	Space record backwards	1-99
CN	Connect a program to a software level	1-91
CT	Connect a program to a timer	1-92
DF	Delete a file	1-98
DL	Delete a file code	1-97
DN	Disconnect a program from a level	1-93
DT	Disconnect a program from a timer	1-93
EE	End of commands	1-99
FF	Space file forward	1-98
FR	Space record forward	1-98
FP	Keep file	1-99
LD	Load a memory resident program	1-88
Magnetic Tape Control Commands		
RO	Declare a read only program	1-99
RW	Rewind tape	1-98
SC	Set clock	1-94
SD	Set date	1-94
ST	Start a program	1-94
SW	Declare a swappable program	1-90
TS	Define time slice	1-91
UN	Unload tape	1-98
WF	Write End-Of-File mark	1-99
WS	Write End-Of-Segment mark	1-99
WV	Write End-Of-Volume mark	1-99

Message	Meaning	Page
OC	Request SCL	1-102
CR	Correction	1-102
DD	Dump Disc	1-102
DM	Dump Memory	1-102
HD	Halt Dump	1-103
ST	Stop CPU	1-103
RD	Release Device	1-103
RT	Retry I/O Operation	1-103
WM	Write Memory	1-104

**Appendix H****Monitor Requests**

Request	LEM	Page
Input/Output	1	1-107
Wait for an Event	2	1-112
Exit	3	1-114
Get Buffer	4	1-115
Release Buffer	5	1-117
Connect Program to Timer	10	1-118
Disconnect Program from Timer	11	1-120
Activate a Program	12	1-121
Switch Inside a Software Level	13	1-123
Attach Device to Program	14	1-124
Detach Device from Program	15	1-126
Get Time	17	1-127
Set an Event	18	1-129
Connect Program to Software Level	20	1-130
Disconnect Program from Level	21	1-131
Wait for a Given Time	22	1-132
Assign a File Code	23	1-134
Delete a File Code	24	1-137
Read Unsolicited Operator Message	25	1-138
Cancel Unsolicited Message	26	1-140

## Appendix G

## Device Addresses

It is recommended to give device addresses according to the following table:

ADDRESS FIELD (6 bits)		000	011000	100000	11				
3 leftmost bits 3 right- most bits		/00	/08	/10	/18	/20	/28	/30	/38
	/00	(SLCU2)	System Console	(ALCU2/4)	PTR		P2P		
	/01	Disc#: X1216 X1215 Floppy	(SLCU2)	Disc#: X1216 X1215 Floppy	(ALCU2/4)	Disc#: X1216 X1215 Floppy	Disc#: X1216 X1215 Floppy		
010	/02	Disc#: X1215 X1216 Floppy	(SLCU4)	Disc#: X1215 X1216 Floppy	(ALCU4)	Disc#: X1215 X1216 Floppy	Disc#: X1215 X1216 Floppy		
	/03	Disc#: Floppy X1216 X1215	(SLCU4)	Disc#: Floppy X1216 X1215	(ALCU4)	Disc#: Floppy X1216 X1215	Disc#: Floppy X1216 X1215		
	/04	MP*	(SLCU4)	MP*		MP*	MP*		
	/05	TK*	(SLCU4)	TK*		TK*			
	/06	CR	(AMAS) Input	Disc#: CDC40 CDC80			Disc#: CDC40 CDC80		
	/07	LP	(AMAS) Output	Disc#: CDC80 CDC40			Disc#: CDC80 CDC40		

### Explanatory Notes:

- All addresses on one line marked with an asterisk are reserved for a multiple device control unit, even if only one device is attached.
- Where several disc types are given for one address, they are listed in order of priority for that address. Only the first one is to be considered standard.
- Columns 1, 3 and 5 are reserved for Data Communication control units; because of the wide variety of configurations and exchange rates, no fixed allocations can be defined and no standard is assumed. (cont.)

- Columns 1, 3 and 5 are reserved for Data Communication control units; because of the wide variety of configurations and exchange rates, no fixed allocations can be defined and no standard is assumed.
- For Data Communication control units the address is of importance only when using the I/O Processor for:
  - throughput possibility
  - optimizing the number of IOPs.
- In some configurations, the first I/O Processor may be used to connect Data Communication control units at addresses which are free; this is in contradiction with the recommended standard, but it saves one IOP.
- An example of a device address on 6 bits is: /38 = 111000.

**PART 2**

**DRTM CONFIGURATION**

---

2-1

The Bicc Real Time Monitor consists of a number of modules, from which the user can select those that are required for his application and configure his own monitor. The main parts of the monitor are:

- Nucleus, consisting of monitor modules running at level 48
- System Interrupt Modules, running at levels between 0 and 47
- System Programs, running at levels between 49 and 63, of which the System Command Language (SCL) module is the most important one.

All modules are centered around the dispatcher, a monitor routine which determines which routine is to be executed next, on the basis of priority levels.

#### NUCLEUS

The nucleus of the BRTM consists of a set of monitor modules which must be memory resident and run at level 48, and of a number of system tables.

Usually, the modules at level 48 are a continuation of an interrupt at a higher priority level: an interrupt is accepted at e.g. level 5, a branch is made to a module at level 48 and there the interrupt is processed. This enables other interrupts to be accepted again.

#### Monitor Modules

Monitor modules running at level 48 are the dispatcher and a number of service routines to handle certain monitor requests. The dispatcher's function is to decide which is the next routine or program which must be executed.

The monitor request handlers perform the following functions:

- Physical I/O (LKN1)
- Activate (LKN12)
- Exit (LKN3)

- Set Event (LKM18)
- Get Buffer/Release Buffer (LKM 4/5)
- Wait for an Event (LKM2)
- Switch inside a Program (LKM15)

They all run at level 48 and thus make use of the A75 stack just as interrupt routines.

#### System Tables

The following tables are part of the DRIM nucleus:

- T:CVT, Communication Vector Table, contains parameters concerning the memory configuration, such as memory size, stack base address, dispatcher address, etc.
- T:LKM, monitor request table, containing the addresses of the monitor request handlers.
- T:RMAC, contains the addresses of the memory resident monitor request handlers.
- T:PCT, Program Control Table, is actually a pool of PCT's, one for each user program. Each table consist of 16 words of information relevant to the program to which it refers.
- T:SLT, Software Level Table, consist of 15 entries, corresponding to priority levels 40 to 63. It is used by the dispatcher to find the current PCT connected to a given level.
- T:FCT, File Code Table, contains the links between logical file codes and the disc files or peripheral devices to which they have been assigned.
- T:SWP, containing the PCT addresses of swappable programs.
- DWT, Device Work Table, one for each peripheral device, containing the necessary parameters on the device and its logical handling.
- T:DCT, Disc Control Table, supplements the DWT for the disc I/O driver and facilitates communication between Data Management, disc driver and monitor. There is one DCT for each disc and DAD.
- T:LFT, Logical File Description Table, contains the information necessary for an I/O operation on a logical file. One table is assigned to each file.
- Timer Management Tables.
- T:DPT (Pool of DAD Control Tables (DADOT))
- T:BTB (Pool of Bit Tables (BTB))

### SYSTEM INTERRUPT MODULES

This set of modules consists of a number of routines which service certain hardware interrupts. The set can be extended with user-written interrupt routines.

- Power Failure Routine (I:PPAR) handles the power failure interrupt and normally operates on the highest priority level (0).
- LKM Handler (I:LKM) handles the LKM interrupt when a monitor request is given, then branches to the requested handler. Also handles the interrupt which occurs when an illegal instruction code is used. Normally operates on priority level 1.

- Real Time Clock Interrupt Routine (I:RTC), which handles the real time clock.
- I/O interrupt routines, which handle the interrupts coming from the various standard peripheral devices.
- User interrupt routines processing the interrupt from external user devices.

Note: Not-recognized interrupts, i.e. interrupts for which no routine has been included at system generation time or loaded at initialization time, will cause a system hang-up.

### SYSTEM PROGRAMS

Various programs, running at levels between 49 and 63, may be included in the system. They may be either memory resident or read only.

#### System Loader

Used to load memory resident, read only, swappable or background programs and interrupt routines. For the read only, swappable and background programs the loader is activated by the dispatcher. For memory resident programs it is called by the System Command Language.

### System Command Language (SCL)

To process the various DRTM control commands, different routines may be activated. These routines are read only and need not be memory resident. They can be connected to any level from 49 to 61 in case file code /EO (control command input) is assigned to the typewriter. When /EO is assigned to another device, the SCL routines must be connected to a level between that of D:OCOM +1 (see below) and 61. Output from the SCL takes place on file code /EP, which must always be the typewriter.

### Control Panel Programs

A number of programs is required to process the control panel interrupt and the operator commands. These programs may be connected to any level between 49 and 61, but the choice of level must be made very carefully:

- D:CTPP reads the operator commands and activates the routine required to process a specific command.  
D:CTPP must be memory resident and it is recommended to connect it to a high level, i.e. 49. This will not be a serious objection because it is a relatively short program and its main function is to read an operator command with 'implicit wait'.
- D:OCOM processes various commands, especially NY and RD, and prints error messages. It is recommended to use it as a read only program.

It may be connected to any level from 49 to 60, but if a higher-priority read only program is running, D:OCOM will not get control. Thus the system may be locked, if such a program accidentally requires operator intervention as a result of an error upon a request for I/O, since D:OCOM may not be loaded.

Therefore, D:OCOM must be connected to a priority level higher than those of all other read only programs requesting I/O on devices on which retry operations are possible. For example, if D:OCOM is connected to level 55, read only programs at levels 49 to 55 must not use devices such as line printer or punched tape equipment, but disc files only.

For these reasons, it is recommended to connect D:OCOM to priority level 49.

- D:DUMP is an optional program which is used in the system when the dump feature is required. It is a read only program. Since it makes use of the line printer (on which retry operations are possible) it must be connected to a level between that of D:OCOM +1 and 61.

#### Data Management

This program performs the logical input/output for disc devices. It may be connected to any level between 49 and 61 and can be used by any user program. If the priority level of the user program is higher than that of Data Management, the request is recorded in the Activate queue and serviced as soon as Data Management becomes the program with the highest priority in the system.

The Data Management program must be memory resident and it is recommended to connect it to level 49.

The system itself does not make use of disc logical I/O, so if the user wishes he can remove this module from the monitor to save memory space.

#### User Service Calls

These are routines handling certain monitor requests made by user programs, such as Assign File Code, Delete File Code, Connect a Program to a Level, etc.

They may be memory resident or read only and can be connected to any level from 49 to 61. It is recommended to use level 49.

If they are memory resident, they can be called by any user program. If they are read only, however, they can be called only by user programs of a lower priority level.

At system generation time, the user can select the monitor request handlers which he requires for his programs. They may be individually declared as read only or memory resident.

Giving a monitor request of which the corresponding request handler has not been included, may result in system hang-up.

#### Disc Allocation Program

The Disc allocation program, which may be connected to any level from 49 to 61, handles the allocation and deallocation of granules for disc files.

It must be memory resident and will be connected to level 49. The Disc Allocation Program is called by Data Management and by the monitor requests Assign File Code and Delete File Code. This implies that, if it is declared as a read only program, it must be of a higher priority than those programs.

#### Time Handler

This program is activated by the real time clock and handles all the programs connected to this clock or to one of the timers. It must be declared memory resident, because it is activated with every real time clock interrupt.

Note: For all system read only programs, the start address must be the first word of the program.

DISPATCHER

All monitor modules are centered around the dispatcher, a module which runs at level 40 and divides central processor time between programs according to their priority level.

When an interrupt has been handled, the dispatcher determines which program must be started and prepares its activation by loading its start address and register contents from the save area (a 16-word area in front of the program or in the Read Only Save Area). For hardware interrupt levels and level 40, control will be returned to the interrupted routine. For software levels (>40), the dispatcher will compare the priority levels of the programs which are active and not waiting for an event, in order to find the one with the highest priority. The dispatcher finds this information in the Program Control Tables, the addresses of which it looks up in the Software Level Table. If the highest priority program is not the interrupted program, it will receive control after the relevant data of the interrupted program (P-register, PSW, registers A1 to A14) have been stored in its save area.

If the interrupted program was also the one with the current highest priority, it will be restarted.

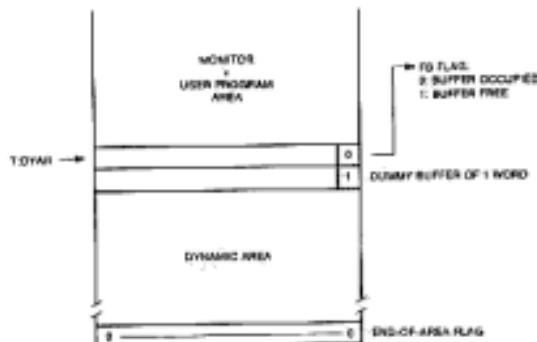
MEMORY ORGANIZATION

The memory layout has already been described in Part 1, Chapter 2. In this section the internal management of the various partitions will be discussed.

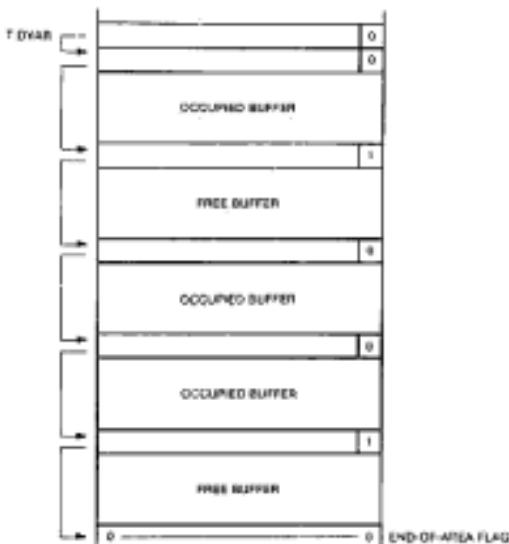
Dynamic Allocation Area

The area of memory which remains after the Initial Program Loader has loaded the monitor and user programs, is formatted by INIMON as are all other memory partitions. The minimum size of the area,

to be specified at system generation time, is 704 words. INIMON gives the following layout to this area, which is to be used for dynamically allocating buffer space to monitor or user programs:



Blocks of memory space can be requested either by the system itself, or by the user through a Get Buffer monitor request. When a buffer is allocated, a buffer guide of one word is set in front of it, in which bit 15 is set to 0, to indicate that the buffer is allocated. If a request is sent for deallocation of this buffer, either by the system or by the user (Release Buffer monitor request), this bit is set to 1, to indicate that the buffer is free again. After a number of such requests, the dynamic area might look as follows:

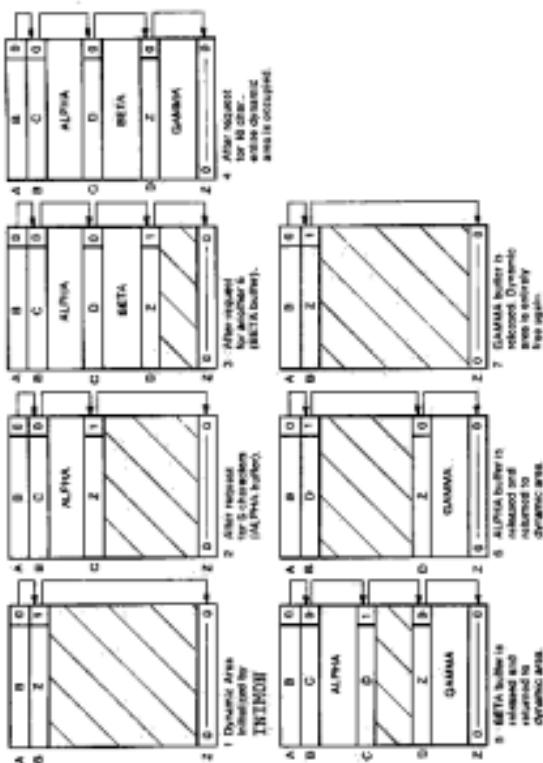


Any new requests for allocation are handled in such a manner that the first free area encountered, which is large enough for the request, is allocated.

Allocation is done by the M:DMA module.

Deallocation is done by the M:DML module.

Below an example is given of a sequence of requests for allocation and deallocation of buffers in the dynamic area.



### Synthetic Areas

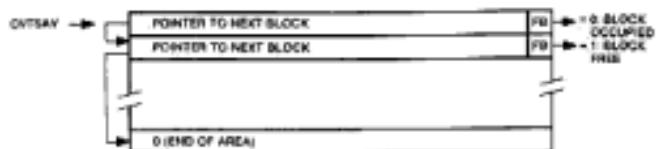
This area, if used, is located behind the memory resident area. Its length is defined at system generation time and this size will be put in location CPTSWP of the Communication Vector Table (T:CVT). After loading, this word will point to the first word of the Swap Area, which is initialized by INIMON.

The size of the Swap Area must be at least as long as the longest swappable program used. To optimize the swapping operations, however, it is recommended that this be a multiple of 200 words.

### Program Save Area

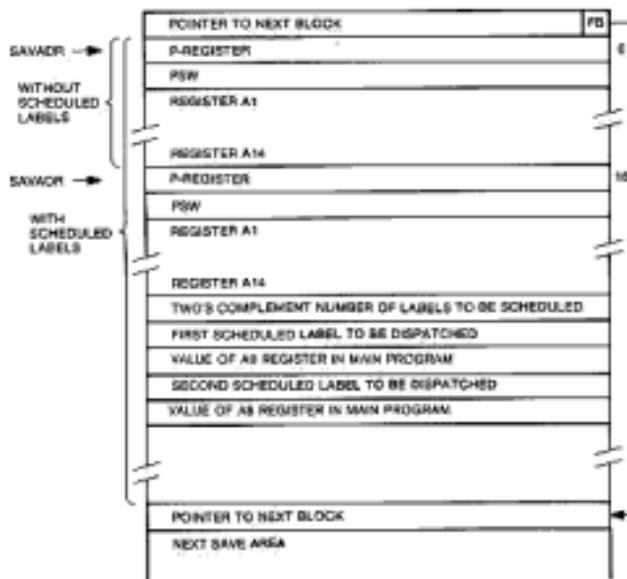
Each time a program on level 49 to 61 is declared (RO, LD and SW commands for read only, memory resident and swappable programs), the system allocates a save area for this program in the Program Save Area. This is done in the same way as buffers are allocated in the Dynamic Allocation Area.

At system generation time, only the size of the Program Save Area is defined, which must be at least 170 words. It can be redefined at monitor loading time. After the system has been loaded, the INTMON module will initialize this area as follows:



For one program's save area a block is allocated of 16 words or, if scheduled labels are used in the program, of  $16+1+16+2n$  words, where  $n$  is the maximum number of scheduled labels allowed in quest (FILLER, see Part 1, Chapter 4) at the same time. (The number 16 consists of: P-register, PSW, register A1 to A14).

This area is filled as follows:



The pointer SAVADR in the Program Control Table (PCT) will point directly to the second word of the Save area, as the first word is used for space management to link the blocks in the save area.

#### Read Only Area

This area is located behind the Program Save Area in memory. Its length is defined at system generation and must be at least 1068 words. The word CVIRBO in the Communication Vector Table (T:CVT, see below) contains this length at system generation. After loading, the monitor initialization program INIMON will store the

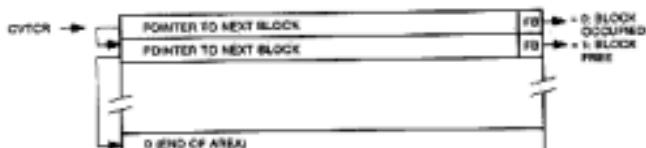
beginning address of the Read Only Area in the location CVTOR0,

#### Memory Resident Area

This area is used for memory resident programs.

At system generation time, the location CVTOR in the Communication Vector Table contains the length of this area.

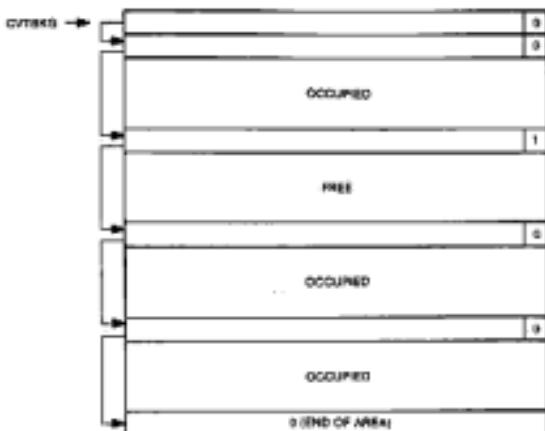
After loading, it points to the first word of this area, which is initialized by the INIMON program as follows:



Memory space is allocated to programs in the same way as in the Dynamic Allocation Area, i.e. each program or memory block is preceded by a pointer, containing the address of the next one and an FB bit.

#### Background Area

Any memory space remaining after the above mentioned areas is implicitly designated as Background Area. The size must be at least equal to the size of the longest background program used. This area is organised in the same way as the Dynamic Allocation Area:



The difference with the Dynamic Allocation Area is, of course, that here blocks are not allocated for buffers or work areas, but for complete programs.

The location CVTBKG in the Communication Vector Table contains the beginning address of the Background Area.

### FILE FORMAT

Apart from the user files, which are either sequential or random as described in Part 1, the system makes use of two other file formats, especially during initialization of the system and during the loading of the different user programs:

#### Load Module Format

When a program is created, e.g. under the Disc Operating Monitor, the result of the assembly and link edit process is a file of the load module type (LM; see Disc Organization in Part 1). All executable programs are recorded on disc permanently in this format, which is relocatable.

#### Core Image Format (D:CI File)

Every time the system is started from scratch, a special file, called D:CI file is created on disc, to optimize the loading speed for read only and swappable programs. This is done at monitor initialization time by the INIMON module.

At first this file contains all the system read only programs, in absolute format. It is expanded with all the read only and swappable programs declared by the user through the SCL commands RO and SW. The system can create this file when the beginning addresses of the Read Only Area and Swap Area are known, i.e. when the partitioning of the memory has been processed. Then the system can relocate all the necessary load module files and copy them onto the D:CI file. All this is done by the monitor segment D:USV3, which is first called by INIMON and after that by the System Command Language SCL when an RO or SW command is processed.

The size of the D:CI file must be declared at System Generation (see Appendix A to Part 1).

### SWAPPING

For each swappable program, two parameters control the swapping process:

- TS
- SBC

Only if they are both 0 a swapping operation may take place. TS indicates the time slicing and is decremented by one every 100 milliseconds.

SEC (swap event count) is incremented for each disc I/O and for each other I/O operation if the S option had been selected at declaration time (see SW system command). It is also incremented during the processing of any monitor requests performed by possibly disc-resident progs. It is decremented at the end of an I/O operation as well as at the end of a monitor request for which it had been incremented.

When both TS and SEC are 0, a swap-out followed by a swap-in is started only if a program of equal or higher priority than the current one is waiting to be swapped.

A program exit forces TS to 0 and in this case, when its SEC becomes 0, the program will be overwritten by the next swappable program, if any.

During wait requests, TS is managed as follows:

- explicit wait: if bit 15 of A8 is set to 1, TS is reset to 0.
- with non-disc I/O operations, if the control unit was busy or if the wait bit was set and if bit 7 of A7 was also set, TS is reset to 0.
- put in wait on dynamic allocation area overflow: no action.

Note:

- All Get Buffer requests issued by swappable programs must be issued with bit 0 in A7 set.
- When a program is put in wait because a device has already been attached to another program, TS is reset to 0.

## SYSTEM TABLES

Some of the monitor modules provide links between the user programs and system programs. These links are in the form of tables, some of fixed length, some of variable length. The tables provide for all communication between user and system software, especially between memory resident and non-resident programs. These tables must be created at system generation time as they define the main characteristics of the system.

The tables may be located anywhere in the first 16k of memory. One table, which contains pointers and information about all the others, has this address stored in a fixed location:

- location /82: address of T:CVT (Communication Vector Table).

The other tables are:

- Each program has a Program Control Table (T:PCT) in the monitor area, where all the necessary parameters about the program are stored and updated by the monitor.
- The Software Level Table (T:SLT) holds the PCT addresses of the current programs for each software priority level (49-63), to enable the dispatcher to find the current highest priority program in case it has to give control to a program.
- When a program wants to perform an I/O operation, this is done through a file code from the desired device or disc file. In the File Code Table (T:FCF) the monitor will find the address of a Device Work Table (DWT) or Logical File Table (LFT), which contains parameters about the device or logical disc file related to the file code specified. The DWT also contains the address of the I/O driver needed to perform the I/O operation. If the I/O operation is to take place on a disc, the Logical File Table (T:LFT) points to a Disc Control Table (T:DCT) which acts as a complement to the DWT for the disc driver and provides a means of communication between Data Management, disc driver and monitor.
- T:RNAC and T:LNM are tables containing the addresses of the routines which process the user program's monitor request.
- T:SWP contains the PCT addresses of the program which must be swapped next, for each software priority level.
- T:DPT (Pool of DADCTs); the DADCT is an extension of the Disc Control Table
- T:HTP (Pool of HTBs); the HTB is an extension of the DADCT.

### Communication Vector Table (T:CVT)

This table provides a useful link between system components which are memory resident and those which are not. It contains the addresses of system tables and common subroutines. By addressing them, in read only system programs, according to their relative positions in the CVT, an external reference from a non-resident to a memory resident system component may be avoided.

T:CVT\$	CVT\$B9
T:CVT\$B	CVT\$C0
T:CVT\$P	CVT\$KEY
T:CVT\$P	CVT\$LAB
T:CVT\$A	CVT\$TEST
T:CVT\$M	CVT\$OMA
T:CVT\$M	CVT\$OML
T:CVT\$L	CVT\$OYE
T:CVT\$T	CVT\$OMR
T:CVT\$R	CVT\$OAE
CVT\$C\$T	CVT\$BY
RESERVED	CVT\$BY+2
CVT\$OMT	CVT\$BY+4
CVT\$OMT	CVT\$BY+6
CVT\$OMT	CVT\$BY+8
CVT\$OMT	CVT\$BY+10
CVT\$OMT	CVT\$BY+12
CVT\$OMT	CVT\$BY+14
CVT\$OMT	CVT\$W\$F
CVT\$OMT	CVT\$W\$P
CVT\$OMT	CVT\$TS
CVT\$OMT	CVT\$C\$D
CVT\$OMT	CVT\$C\$F
CVT\$OMT	CVT\$C\$L
CVT\$OMT	CVT\$C\$C
CVT\$OMT	CVT\$D\$O
CVT\$OMT	CVT\$E\$P
CVT\$OMT	CVT\$W\$F
CVT\$OMT	CVT\$E\$T
CVT\$OMT	CVT\$D\$P
CVT\$OMT	CVT\$S\$B

T:CMSS gives the machine's memory size. It is calculated by the system at loading time. If it is 0, the memory size is 32k.  
 T:USSTB contains the base address of the system stack, as defined at system generation time.  
 T:CIDSP contains the address of the dispatcher.  
 T:CIDP contains the status word of the Idle Task.  
 T:GIDA contains the P-register value of the Idle Task.  
 T:CSLM contains the maximum number of scheduled labels allowed at the same time in one program.  
 T:CTIM contains the address of the Real Time Clock tables.  
 T:CPLS contains the number of pulses of the timer during one 20 msec period (PR): 1 for the 50Hz standard clock.  
 T:CRST contains the non-standard clock reset value.  
 T:DYAR contains, at system generation time, the length of the dynamic allocation area. After the system has been loaded it contains the address of the first word of this area.  
 CVTPCT contains the address of the File Code Table (TiPCT), i.e. of its first word. This word contains the length of the file code table, in characters.  
 CVTDWT contains the address of the first entry in the Device Work Table (DWT). The word (CVTDWT)-2 contains the entry length and the number of entries in the DWT:

entry length (in char.)	number of entries
0	7 8 15

CVTDCT points to word 0 (DCTHD) of the first entry in the Disc Control Table (DCT).  
 CVTPCT points to the first word of the first entry in the Program Control Table (PCT) Pool. This first word contains the name of the first program in the PCT Pool.  
 CVTSAV contains, at system generation time, the length of the Program Save Area. After the system has been loaded, it points to the first word of this area.

CVTRD0 contains, at system generation time, the length of the Read Only Area. After the system has been loaded, it points to the first word of this area. The Read Only Area is preceded by two reserved words used by the loader.

CVTRCR contains, at system generation time, the length of the Memory Resident Area. After the system has been loaded, it points to the first word of this area.

CVTBKG need not be set at system generation time. After system loading, it will contain the address of the first word of the Background Area.

CVTOCM contains the address of the 72-character buffer which is used to read an operator message. The buffer must be located in the first 16k of memory, because the sign bit is used to indicate whether the buffer is being processed (-1) or free (=0) to accept an operator message. If the sign bit is set to 1 and the operator presses the control panel interrupt button, it will be ignored by the system.

CVTRKEY is set to zero at system generation time and is used by the 'Read an Unsolicited Key-in' monitor request to build the chain of requests from user programs for reading the unsolicited key-in.

CVTLAB contains the address of a dispatcher subroutine which manages the scheduled label queue in the Save Area.

CVTBDE, CVTBKE, CVTCBE and CVTSAB are set to /8000 at system generation time. They are used to indicate overflow (sign bit = 0) or not (-1) in the Dynamic Allocation Area, Background Area, Memory Resident Area and Program Save Area. In case of overflow, these event bits are used to put a requesting program in wait state.

CVTSSY is a status word containing various DPTM flags. The sign bit and bit 1 are used by the Dump program when the HD command is given.

Bit 2 = 1: the automatic restart routine stops after a power failure to allow the operator enough time to perform any manual operations required.

Bit 2 = 0: no halt in the automatic restart routine.

Bit 3 = 1: the system is ready to run. This bit is reset to 0 at system generation time. After completion of loading and initialization, it is set to 1 to indicate that all system read only programs are now recorded in the B:CI file, in core image format.

Two other status words are also available for user programs. They can be used by any application either to contain table addresses or any information about the application. These words are all reset to zero at system generation time.

CVTPWF Power failure/automatic restart option, set by system generation. If it contains 0, the option has not been selected and the system will issue a halt in case of power failure. If the option has been selected, this word contains the address of the automatic restart routine.

CVTSSY + 2 contains the address of the M:ENAM routine (search PGT).

CVTSSY + 4 contains the address of the ETIMER chain pointer (B:POIN)

CVTSSY + 6 contains the address of the CTIMER chain pointer (C:POIN)

CVTSSY + 8 contains the address of the RTIMER chain pointer (E:POIN)

CVTSSY + 10 contains the address the release timer block routine (F:SLK)

CVTSSY + 12 and +14 are status words reserved for the user for inter-program communication.

Bit 15 is used to denote a power failure which has not been processed.

CVTSWP At system generation time this word contains the size of the swap area (at least 2 characters). After initialization it contains the beginning address of this area. The area is preceded by 2 reserved words used by the loader.

CVTTS This word contains a value used as the standard or default value of the time slice for swappable programs. It is set either at system generation or by the command TS.

CVTCID Contains the file code of the disc which contains the D:CI file. It is set at system generation time.

CVTCIP Contains zero at system generation time. After initialization it points to the first free sector of the D:CI file.

CVTCIL At system generation time this word contains the size (in number of granules) of the D:CI file. After initialization it points to the last sector of the D:CI file.

CVTPSC This word is the time slice counter of the current swappable program (reset at zero at system generation time). When this value reaches zero, the program is swapped out and another swappable program is activated.

CVTFDC } 2 words used by EDPM  
CVTEFT }

CVTWAT contains the entry point in the Wait routine

CVTSET contains the entry point in the Reset routine

CVTDP contains the address of the first word of the DADCT Pool.

CVTEFP contains the address of the first word of the BTFAB Pool.

#### Program Control Table (T:PCT)

This table contains all relevant information about any program and relates to all the activities taking place in the system at a specific moment. There is one 18-word entry for every program, some of them for system programs, some, reserved at system generation time, for use by the System Command Language for the processing of control commands. When a program is deleted, the corresponding PCT entry is released.

The address of T:PCT is stored in the Communication Vector Table (T:CVT), to facilitate access by the System Command Language for creating a PCT entry.

At system generation time the number of PCT entries must be specified. This number cannot be modified. Reference to the PCT must be made through the label indexed by the PCT address, i.e. the word STATUS in the PCT.

One PCT entry has the following layout:

	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487	1488	1489	1490	1491	1492	1493	1494	1495</

E=0: the program has not made its exit.  
E=1: the program has made its exit (set by Exit (M:EXIT), when the event counts (words 22 and 24) are not zero; reset when the event counts have become zero).  
RO=1: this is a read only program  
BG=1: this is a background program  
C=1: this is a memory resident program  
RO=BG=C=0: this is a swappable program  
L=0: the program has not yet been loaded into memory.  
L=1: the program has been loaded into memory.  
LP=1: this is the last entry in the Program Control Table.  
NO=1: this program must not be dispatched (not operable, suspended).  
SL=1: at least one scheduled label has been recorded for this program. This bit is set when a scheduled label is sent to the dispatcher and reset when all scheduled labels have given their exit.

The program cannot become inactive until the counter (PGTNVC) has become zero again (see bit E).

ECBWT: ECBWT or ECBWOL (SL bit set in last case) of the next PCT waiting on the same ECB as the one on which the main program waits (see T:EVT table)

ECBACT: A block of three words to enable the activating program to synchronize itself with this program, i.e. the program activated by it. The block contains:  
- ECB address  
- PCT address  
- scheduled label address

The activating program can synchronize itself by giving a Wait monitor request after the Activate request or by giving a scheduled label at activation time. When the activated program is completed, the label will be scheduled. However, if a Wait request has been given, the program itself must send a Set Event monitor request. This is not

done implicitly upon exit of the program.

CHLK: Chaining link (see below).

CHPTD: The left character of this word contains the time slice value for this program. It is set by the control command SW (Declare a Swapable Program). Default value = CVTTS (see Communication Vector Table).

DISK1: Contains the relative sector number of the first sector of this program on disc. If it is a background program, it points to the sector GRANTD of the load module. If it is a core image program, it points to the first sector of the program within the D:CI file.

DISK2: Bits 0 - 7 contain the file code of the disc (F0 to FF) on which this program is located.

Bit I = 0: All pending I/O operations of the current swapable program must be completed before swapping the program.

Bit I = 1: Only the pending disc I/O request must be completed before swapping the program (ECB and buffer areas used by non-disc devices are outside the swap area).

Bit LD = 1: This is a background program which is being loaded.

Bits 10 - 15 contain the level to which this PCT is connected.

ECBSCL: ECBWT or ECBSCL (SL bit set in last case) of the next PCT waiting on the same ECB as the one on which the scheduled label waits (see T:EVT table).

TIMAD: Contains the memory space required to load the program, if this is a background program, or the beginning address of the program if it is loaded into the memory resident area. It then points to the first word of the memory block into which the program is loaded, i.e. the block pointer.

PCTEVC: Event Count used for Exit Control.

PCTEBC: Swap event count used for swap control. The program may be swapped out only when this word reaches the value zero.

### Use of Event Counts

Generally speaking, these words are incremented whenever an action such as an I/O operation, a scheduled label, execution of a non-resident monitor request, is requested by the user and decremented at completion of the action.

For the event counts PCTEVC and PCTSEC, these are the following actions:

- PCTEVC: - incremented by IORM  
decremented by ENDIO
- incremented by I:LEM if scheduled label requested  
decremented by M:LAB (M:DISP) when scheduling occurs
- incremented by I:LEM when a non-resident monitor request is requested  
decremented by the monitor request handler itself when the request has been processed.

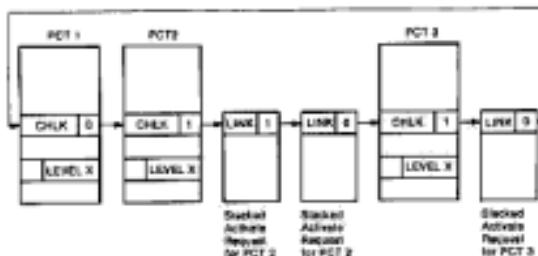
  

- PCTSEC: - incremented by IORM if bit I of DISK 2 is set or if it is disc I/O or bit 7 for a non-disc I/O order is not set  
decremented by ENDIO under the same conditions
- incremented by I:LEM when a possibly disc-resident monitor request is requested  
decremented by the monitor request handler itself when the request has been processed.

### Chaining Link

If, in this word,  $P = 0$ , it indicates a link to another PCT. If  $P = 1$ , it is a link to a stacked Activate request.

All PCT blocks connected to the same priority level and the stacked Activate requests for them are linked together in a chain as follows:



- PCT's are inserted into the chain each time a monitor request 'Connect to a Level' is given for a program on the same level. If there is only one PCT, the chain loops on itself.
- Activate requests are stacked each time an Activate request is given for a program which is already active: an entry is created (by dynamic memory allocation) and inserted into the PCT chain after the PCT block itself. The entries for these stacked Activate requests have the following layout:
- Level 49 stacked entry:

LINK	F
ADDRESS OF CALLED ROUTINE	(A0)
PCT ADDRESS OF ACTIVATING PROG.	(A1)
SCHEDULED LABEL ADDRESS	(A2)
PROGRAM NAME ADDRESS	(A3)
EOF ADDRESS	(A4)

These blocks are created by the Activate monitor request handler (M:ACT) and erased by the Exit monitor request handler (M:EXIT).

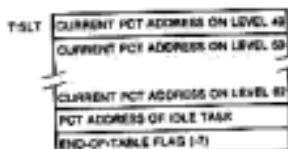
#### Software Level Table (T:SLT)

This table, for which the entry points and the format are defined in the monitor module T:STS, is used by the dispatcher to find the current PCT for a given level.

The priority level of a Software Level Program is specified by the position of its PCT-address word in the Software Level Table.

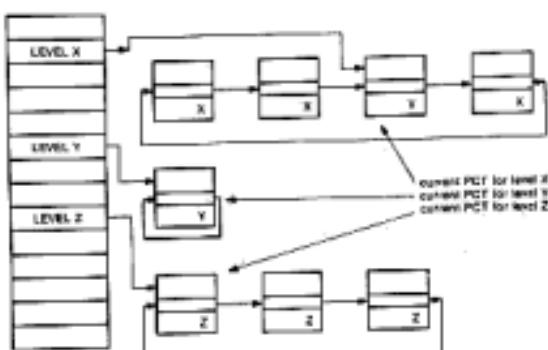
The SLT contains 15 entries:

- one for each of the levels 49 through 62
- one for level 63 (level for the Idle Task).



#### Example of PCT-SLT Organization

The links between SLT and PCT entries are created by the 'Connect a Level' monitor request handler (M:CNLV) and deleted by the 'Disconnect a Level' request handler (M:DNLV), when there is only one PCT in the chain. Within the chain itself the links are changed by the Switch Inside a Level request handler (M:SWC).



### Event Tables (T:EVT)

This table contains up to 16 entries of 2 words each. These entries are created when a program for the first time wants to wait for one specific event (explicit or implicit wait) and they are released when the last program waiting for that event is restarted (by a Set Event request). The entries in the table are shared by both system and user programs.

Only one table is delivered as standard in a separate module. If it becomes necessary, the user may expand it with more such tables.

This standard module is called M:EVTB and its entry point is called T:EVT, referred to in the M:WAIT and M:REEV modules.

The entry point T:EVT must correspond to the word NEXTADD of the first table:

NEXTADD	0	Address of next table, if any; 0 otherwise
BUSPAT	2	Busy entries pattern
1st entry	4	ECB address
	6	Addr. of ECBWT or ECBBSU of waiting PGT SI
	8	
	10	
16th entry	64	
	66	

NEXTADD : Address of the word NEXTADD of the next event table, if there is any; otherwise, this word contains 0.

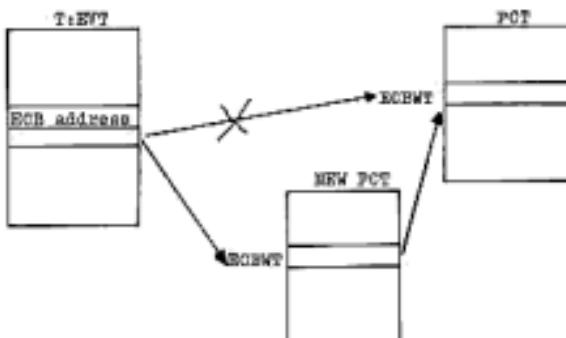
BUSPAT : Pattern of the busy entries in the table, each bit corresponding to one entry in the table (bit 0 = 16th entry, bit 15 = 1st entry). A set bit indicates that the corresponding entry is free.

Each entry is 2 words long:

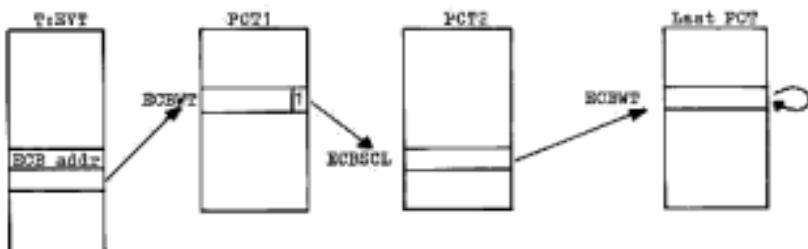
- the first word contains the address of the ECB on which the program is waiting
- the contents of the second word depends on the sequence type of the

waiting program: if it is the main program sequence it contains the address of the word ECBWT of the PCT of the program, if it is a scheduled label sequence, it contains the address of the word ECBSCL of the PCT (in this case bit 15, SB, is set).

As soon as another program wants to wait on the same ECB, the link between the T:EVT and the first PCT is broken and the new PCT is inserted into the chains:



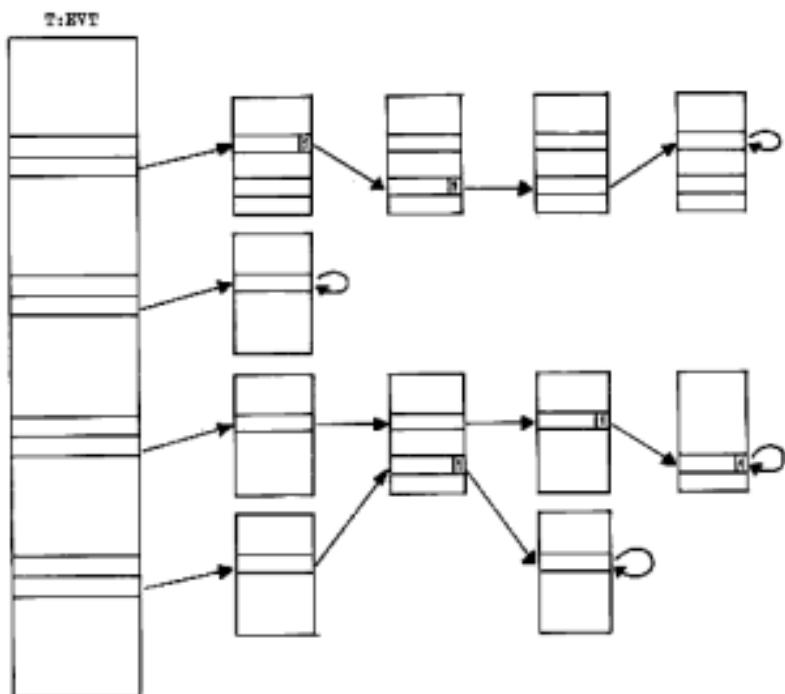
All PCTs waiting on the same ECB are chained as follows:



Event Table Overflow: when no more free entry is left in the Event Table or Pool of Event Tables, a branch is made to SYNAH, with register A2 = 6.

Note: Due to the response time of the requests Wait for an Event and Set Event, it is not recommended to extend the pool of event tables excessively.

Example of T:EVT - PCT organizations:



### Swapping Table (T:SWP)

When a time-slice has elapsed for one program, the dispatcher (M:DISP) needs to know the next program to be swapped in. It is found in this table where these programs are recorded (one per level). If there is no program to be swapped in, the entry in T:SWP is set to zero.

T:SWP →

PCT address of next program to be swapped in for level 49
PCT address ..... for level 50
PCT address ..... for level 51
PCT address ..... for level 52
PCT address ..... for level 53
PCT address ..... for level 54
PCT address ..... for level 55
PCT address ..... for level 56
PCT address ..... for level 57
PCT address ..... for level 58
PCT address ..... for level 59
PCT address ..... for level 60
PCT address ..... for level 61
PCT address ..... for level 62

### Device Work Table (DWT)

The DWT is a monitor module, which consists of fixed length 19 word entries, one entry being built for each peripheral device. Note that the AEE is considered as consisting of three different devices and, consequently, occupies three entries in the DWT: one each for the typewriter, the tape reader and the tape punch.

T:DWT points to the first entry of the table which is preceded by a word which contains, in bits 0 - 7, the length of one entry in characters, and in bits 8 - 15, the number of entries in the DWT.

An entry in the IWT has the following layout:

ENTRY POINT	ENTRY LENGTH	NUMBER OF ENTRIES
DWTDN	8	1
DWTDA	2	1
DWTBLG	4	1
DWTDRV	6	1
DWTSTS	8	1
DWTDCB	10	1
DWTBIF	12	1
DWTBLG	14	1
DWTBLG	16	1
DWTORD	18	1
DWTRY	20	1
DWTBAS	22	1
DWTCHM	24	1
DWTCTL	26	1
DWTAS	28	1
DWTAS	30	1
DWTCHN	32	1
DWTATT	34	1
DWTST	36	1

(# These words should not be modified by any user written drivers; see chapter 3 of this part).

where:

WORD 0 contains the device name, consisting of 2 ASCII characters.

WORD 2 contains a binary value giving the physical address of the device.

WORD 4 contains the recommended maximum record length for this device. This number depends on physical device limitations, the transfer rate of the I/O processor and on compatibility with other devices:

typewriter, tape punch, tape reader, card reader: 80,  
cassette tape: 256, magnetic tape: 4095. Line printer: 80 or 136.

WORD 6 contains the address of the I/O driver belonging to this device.

WORD 8 Software status, used to record any I/O error during data

transfers on the I/O bus or to register the specific basic I/O operation to be performed for an order for Tape Cassette or Magnetic Tape (see Part 1, Chapter 4)

WORD 10 contains the address of the user's ECB.

WORD 12 contains the user buffer address, or the character address of the next character to be input or output.

WORD 14 Requested length of the I/O operation, given by the user.

WORD 16 Effective length, used to count the number of characters actually transferred during an I/O operation.

WORD 18 The order for the I/O operation, as specified by the user in the A7 register to define the particular I/O function to be performed.

WORD 20 is used to record that the user wants the standard recovery feature, even for Basic I/O orders.

WORD 22 Next character or word to be output, for devices on the I/O bus. For disc devices, this word points to the DCT (Disc Control Table) address.

WORD 24 For an object write order this word contains the checksum. For Line Printers, this word contains the last character, saved from the buffer.

WORD 26 specifies, in case of 4 x 4 object input for devices on the I/O bus, whether the right of left character must be entered. In case of line printer output it is used to save the control code (first two characters of a buffer).

WORD 28 is used to save the program POT address as recorded at initialization time by the IORM module. It is returned to the dispatcher at completion of the I/O operation.

WORD 30 is used to save a scheduled label address, as recorded at initialization by the IORM module. It is returned to the dispatcher at completion of the I/O operation. This word contains 0, if no label has been scheduled.

WORD 32 For disc, address of location DCTMD in the Disc Control Table. For non-disc devices, address of the control unit status word. With every DWT entry, a control unit status word is associated, indicating whether an I/O operation

can be initiated or not (busy/not busy). The sign bit indicates whether the device is free or not.

For non-disc devices, this word is used as follows:

F			
0	7	8	15

F = 1 if the control unit is free; 0, if busy.

If busy, bits 7 to 15 are used as follows:

bit 7 = 1, set by M:HUNTR to indicate that the device is waiting for an EY or RD operator message. If EY is typed it is reset to zero; if RD is typed, it remains 1, to indicate to M:RETR not to output the NY flag in any following PU error message and return the status to the calling program.

Bits 8 to 15 contain the contents of the BOU lines, to be sent when a CIO instruction is received.

WORD 34 This word is used to record the PCT address of the program which has been attached to this device by means of a monitor request. If this device is not attached, this word contains the hexadecimal value /8000.

WORD 35 This word is used to find the SST (send status) sequence address relative to a device, in case of a throughput error on a device connected to the I/O bus.

### T:DCT (Pool of DCTs: Disc Control Tables)

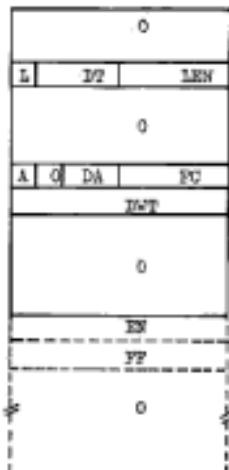
At system generation time, one entry must be created for each disc in the configuration. For it, the user specifies disc type, interrupt level and number of queue entries.

The queue at the rear of each DCT may be omitted if the user will not make use of DPM in accessing the disc.

The DCT length is variable, because the length of the queue is variable.

The entry point T:DCT points to the start address of the first DCT in the pool. This is not the first word of the DCT (DCT address 0 = DCTHD).

At system generation time a number of fields are initialised already and the initial layout of the DCT is as follows:



### DCT (Disc Control Table)

The Disc Control Table is a complement to the DWT (Device Work Table) for the disc drivers. It facilitates communication between Disc File Management, disc driver and monitor.

Layout is as follows

DCTCHP	-16	CURRENT HEAD POSITION																			
DCTCDA	-14	CURRENT DADCT ADDRESS																			
DCTFPDA	-12	ADDRESS OF FIRST DADCT																			
DCTERY	-10	RETRY COUNT																			
DCTLG	-08	L	DISC TYPE	DCT LENGTH																	
DCTED0	-00	EVENT CHAR.				LOG. FLD P.C.															
DCTEB1	-01	BUFFER ADDRESS																			
DCTEB2	-08	SECTOR LENGTH																			
DCTEB3	-06	EFFECTIVE LENGTH																			
DCTEB4	-04	RETURNED STATUS																			
DCTEB5	-02	ABSOLUTE SECTOR NUMBER																			
DCTHD	00	A	SY	DISC DA	PHYS. DISC I.C.																
DCTDWT	02	DWT ADDRESS																			
DCTCUR	04	S	RD	I	SE	S	R	W	B	RT	RR	P	DI								
DCTCIO	06	AREA FOR I/O CONTROL																			
DCTQA	14	QUEUE AREA																			

Note: - Layout of the I/O Control Area is variable, depending on disc type. See below  
- Queue area is present only if specified by user at system generation time. See below.

DCTCHP: current head position; used by the driver to check if a seek operation is necessary.

The value depend on the disc type:

X1215: 0-202

X1216: 0-

CDC 40M: 0-

CDC 80M: 0-

DOTCDA: address of the current DADCT.

Set by IORN via the DAD file code and used by the driver to calculate physical cylinder number physical sector and so on. For physical disc access this word contains 0.

DOTCPDA: first pointer of DADCT chain. (See DADCT).

DOTNRY: Number of times an I/O operation is to be retried before it is marked down as irrecoverable I/O error.

At the first erroneous I/O operation, the retry indicator is set and the retry count initiated.

The initial value depends on the type of operation:

- seek: 1

- read: 26

- write: 3

DOTLG: length of DCT (in characters)

Only the right hand character is used to specify the DCT length.

Bit 0 (L) is used as an end indicator:

L=1: last entry in T:DCT

L=0: not the last entry.

Bit 1 and 2 are not used.

Bit 3 and 4 indicate the disc type:

00: X1215 or X1216

01: CDC disc

Bits 5, 6 and 7 further specify the disc type:

X1215 or X1216: 000 = X1215

001 = X1216

CDC disc : 000 = 40M

001 = 80M.

DCTEB0 to DCTEB5 is the ECB used only by the Disc File Management routine to make a request to the physical disc driver:

DCTEB0: bits 0-7 : event character  
bits 8-15: file code of the DAD on which the file is stored.

DCTEB1: address of the buffer (one sector) used by the physical disc driver to read or write a sector. For sequential access this is the address of the blocking buffer, for direct access of the user buffer.

DCTEB2: length of the buffer used to read or write a sector:  
410 characters.

DCTEB3: effective length, returned by the physical disc drivers:  
410 characters.

DCTEB4: status of the request made for a module on level 49, as returned by a module on level 48 (disc I/O driver).

DCTEB5: relative sector address (relative to beginning of the DAD) of the sector to be read or written.

DCTHD : Base address of the DCT entry.

Bit 0(A) is a software busy indicator:  
= 0 : device is active, request has been received.  
= 1 : device is free.

Bit 1(SY): system disc indicator (set by INTION):  
= 1 : this DCT belongs to the system disc.

Bits 2-7 : physical device address.

Bits 8-15: physical disc file code (/FO to /FF).

DCTDWT: Address of the DWT (Device Work Table) to which this DCT belongs.

DCTOUR: Indicators:

Bit 0(N) : operable indicator:  
= 0 : disc operable (set at system generation)  
= 1 : disc not operable (set by INTION, when disc not ready).

Bit 1(RD): ready indicator:  
= 1 : disc ready, new pack has been mounted, but not yet initialized.

Note: When bit N and/or bit RD is 1, the disc driver will refuse a request, even from a system

program. RD is set to 1 when the system receives an unexpected interrupt from the disc unit while the 'disc ready' bit in the status word is 1.

When RD is 1, the program DISKINIT (part of D:USV1) will be called to reinitialize the disc by

- suppressing the file codes assigned to the files on the old disc,
- scratching the DAD administration of the old disc,
- resetting N and RD to 0,
- resetting bit A in DCTHD

Bits 2 to 7 indicate the operation to be performed:

Bit 2(I) : interrupt indicator:

- = 1 : an order has been sent to a device and the system is waiting for the interrupt. Set to 1 after a CIO instruction, reset to 0 after execution of SPT instruction.

Bit 3(SZ) : seek to zero indicator:

- = 1 : a seek to zero must be performed. Set only on detection of a seek error, either via hardware status or software identifier check.

Bit 4 : not used (must be 0).

Bit 5(S) : seek indicator:

- = 1 : a differential seek must be performed. The contents of register A3 will be given in word DCTSK.

Bit 6(R) : read indicator:

- = 1 : a read order must be sent to the disc.

Bit 7(W) : write indicator:

- = 1 : a write order must be sent to the disc.

Note: The operation indicators SZ, S, W and R are initiated by the disc driver, depending on the requested order:

	SZ	S	R	W
read	0	0	1	0
read (+ seek)	0	1	1	0
write	0	0	0	1
write (+ seek)	0	1	1	1

The driver attempts to execute the operation indicated by the leftmost 1-bit. When the driver resets this bit after a successful SFT instruction, it tries the next one and so on until all bits are zero.

When a seek error is detected, bits SZ, S and R are set and bit W remains unchanged.

Bit 8(S) : position lost indicator:

- 1 : the head position recorded in DCTHMP does not correspond to the actual head position, i.e. the following operation for this disc will be a seek to zero (Note: for CDC discs a seek operation is sufficient).

Bit 9(R1): retry indicator:

- 1 : the disc driver is trying to recover an error. This bit is initiated to 0 when an I/O event is prepared and set to 1 when an error indication is returned after an SFT instruction. This bit is used to check if the retry count in word DCTHRY must be incremented or decremented.

Bit 10(BB): drive busy indicator (only for X1215/X1216):

- 1 : a request is being processed on the other part of an X1215 or X1216 drive: do not start an operation on this part of the drive.

Bit 11(P) : prepared indicator:

- 1 : the DOT is ready to start the I/O operation (but was held up because the control unit was busy).

Bit 12(DD): disc structure indicator

- 1 : this pack has a DAD structure.

Bits 13 to 15 are not used.

DCT10 : Area for I/O control, where the layout depends on the disc type:  
XL215/XL216:

DOTSK	06	SEEK DIFFERENTIAL (BOU LINES)
DOTRD	08	READ COMMAND (BOU LINES)
DOTRM1	0A	MULTIPLEX CW1 FOR READ
DOTRM2	0C	MULTIPLEX CW2 FOR READ
DCTW	0E	WRITE COMMAND (BOU LINES)
DCTW01	10	MULTIPLEX CW1 FOR WRITE
DCTW02	12	MULTIPLEX CW2 FOR WRITE

CDGDISC:

06	SEEK TO ZERO
08	SEEK
0A	WRITE OR READ COMMAND
0C	TOTAL LENGTH CW FORMAT
0E	LENGTH CW FORMAT
10	BUFFER ADDRESS

DCTQA : Queue area, used only by MDFM to queue the requests for files on this disc.  
If DFM is not used on this disc, the user can omit specification of this area at system generation time.  
The number of entries in the queue is also specified at system generation time.

#### The layout of PC7SA 1a:

BCTQEN	14	ADDRESS OF LAST ENTRY
BCTQNR	16	ADDRESS OF NEXT FREE ENTRY
BCTQFR	18	ADDRESS OF FIRST ELEMENT
BCTQHR	1A	ADDRESS OF NEXT ELEMENT
BCTQBR	1C	FIRST ENTRY
	20	

卷之三

DCTQEN gives the address of the last entry in the queue area (i.e. the last 2 words of the DCT). Set at system time.

DCTQBR gives the address of the next free entry in the queue area (initialised to 00000000).

Meditation for MATURES

program gives the address of the first element in the array.

Digitized by srujanika@gmail.com

When HADM is active with a file on this disc, the queue

DCTQBR gives the address of the next element in the queue. It is initiated at 0 and maintained by M:DWN. When it is 0, the queue is empty.

1997-1998 Academic Year

### T:BTP (Pool of BTBs-Bit Tables)

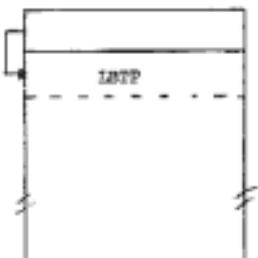
The length of this pool is specified at system generation time: 2+LBTP characters, where LBTP specifies the maximum total length of the BTBAG which are kept in memory simultaneously, for administration of allocation for temporary files.

The first word of the pool contains information regarding the pool itself, the rest is available for BTBs. When a new temporary file is assigned for a DAD, an entry is taken from this pool to create its BTB. It is returned to the pool when the DAD is scratched from the system.

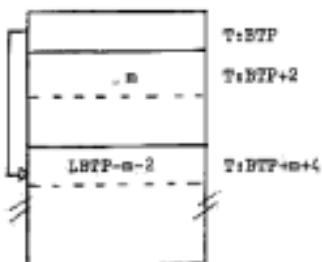
When no more entries are free in the pool, the system releases the "scratchable" BTBs (DADHTP(first character of DADCT word 04)=0).

The unused area of the pool is always located at the end, i.e. when an entry is released any entries following it are pushed up.

The layout of the T:BTP is as follows, where the first word points to the start address of the free area and the first word of the free area contains the remaining length, in characters:



Initial State

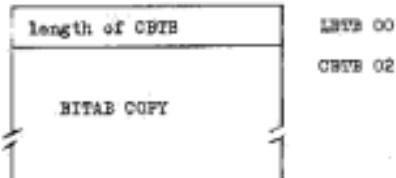


Run-time State  
(one BTB with length=m)

### BTB (Bit Table)

The Bit Table is an extension of the DADCT, required only when temporary file(s) have been assigned to the DAD. It remains linked to the DAD, even when no more temporary files are assigned for it. In this case however, it may be released when no more space is available in the T:BTP for a new Bit Table for another DAD; A BTB is also released when its DADCT is scratched from the system.

The BTB has a variable length and is part of the T:BTP where it has the following layout:



where:

LTBT: length of field CBTB in characters (odd or even).

CBTB: copy of the BITAB on the DAD.

This extension of the DADCT is used only for dynamic allocation of granules to a temporary file. It is read from disc after creation of the BTB, i.e. when the first temporary file is assigned for a DAD. The table is never written back to disc.

Note: The BITAB on disc is updated via the granule table of the file concerned. This update does not affect the BTB. This implies, that a request for allocation of granules could be denied even if there is still space on the DAD (resulting from a previous 'Delete File' command).

T:DP (Pool of DADCT - Direct Access Device Control Tables).

The DADCT is an extension of the Disc Control Table. All DADCTs are contained in the pool T:DP, where the number of entries depends on the number of file codes the user wishes to assign to DADs. This is specified at system generation time and concerns the file codes /CF to /CF- the number of DADs to be assigned (max. 16).

When a file code is assigned to a DAD or when a file is assigned to a DAD or disc, an entry is taken from this pool.

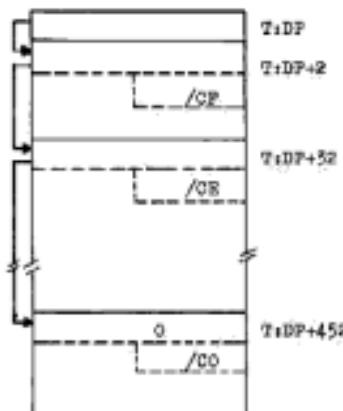
When the disc is dismounted the entry is returned explicitly.

It is returned implicitly when another DAD requires it and the DADCT is "scratchable", i.e. no file codes are assigned to this DAD (DADNFC=0) and no files are assigned to it (DADNFI=0). Non-used entries in the pool are queued from the first word in the pool (T:DP), the last entry being marked by a 0. This queue is initiated at system generation time.

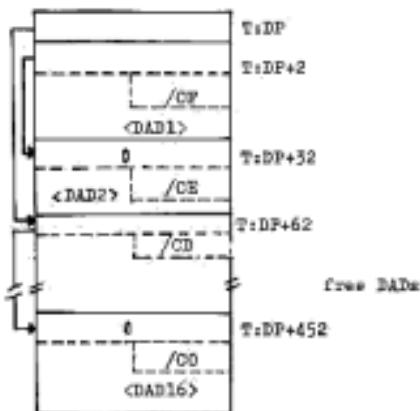
The relation between implicit DAD file code and DADCT is defined and fixed at system generation time, when the field DADIFC in the DADCT is set.

Note: The implicit DAD file code appears in the LPT only when a file has been assigned to that DAD or when a file code has been assigned to it.

The layout of T:DP (for NDAD=16) is as follows:



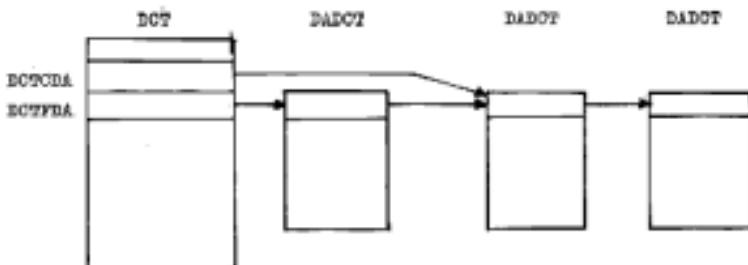
Initial State  
(as defined at synagen)



Run-time State  
(DAD1 and DAD2 are  
assigned on the same disc  
pack).

#### DADCT

The Direct Access Device Control Table is an extension of the DCT. One DCT may be extended by a number of DADCTS linked by a forward chain starting in the DCT (DCTFDA). The DCT also contains the address of DADCT on which I/O is currently taking place (DCTCDA).



All DADCTs are located in the DADCT-pool T:DP and have the following layout:

DADLINK	00	FORWARD CHAIN-LINK		
DADPC	02	# OF ASS.	DAD FILECODE	
DADNP	04	# OF TEMP. FILES	# OF PERM. FILES	
DADDOT	06	DCT ADDRESS		
DADNAME	08			
	0A	DAD NAME		
	0C			
DADINT	0E	NUMBER OF INTERLACES		
DADSEPT	10	NUMBER OF SECTORS/TRACK		
DADSIG	12	SECTOR LENGTH		
DADNBO	14	NUMBER OF CYLINDERS		
DADPCIL	16	NUMBER OF FIRST CYLINDER		
DADSPG	18	NUMBER OF SECTORS/GRANULES		
DADSTA	1A	STATUS	DAD NUMBER	
DADSTS	1C	BTS ADDRESS		

Where:

DADLINK: pointer to the next DADCT belonging to the same DCT (word DCTYDAD)

If 0, this is the last DADCT in the chain.

DADPC : Is split in two characters:

DADHPC: contains the number of file codes assigned to this DAD. If it contains 0, no explicit assign has been made.

DADIPC: implicit file code for this DAD; related to the entry number in the DADCT pool, independent of the DAD, but fixed at system generation time.

H:DPM requires this field to put the file code in the BCB for physical read/write operations.

DADNP : is split in two characters:

DADNTP: Number of temporary files assigned to this DAD.

It is incremented each time a temporary file is assigned for this DAD, and decremented when one is deleted.

This word is checked to determine if a BTS can be scratched

in the BTB pool.

DADNPF: Number of permanent files assigned for this DAD. It is incremented each time a permanent file is assigned for this DAD and decremented when one is deleted.

These two characters, DADNTZ and DADNPF are checked to determine if a DADOT can be scratchedit.

DADOT: Address of the DCT of which this DADOT is an extension.

DADNAME: Name of the DAD (6 ASCII characters).

In four situations, the DAD name is involved, depending on the following factors:

- a disc has a DAD structure : D
- a disc does not have a DAD structure : 7 D
- assign is done by disc number + DAD name : N
- assign is done by disc number only (first DAD): 7 N

Thus:	N	7 N
D	1.	3.
7 D	2.	4.

1: DAD name is set at time of assignment and checked against the VTAC.

2: assign is refused.

3: DAD name is read from the VTAC

4: DAD name is left as six blanks.

DADINT: Number of interlaces on this DAD.

Fixed per disc type as follows:

XI215: 3

XI216: 3

CDC 40M: }  
CDC 80M: } Fixed at Premark.

DADSPF: Number of sectors per track in this DAD.

Fixed per disc type as follows:

XI215: 16

XI216: 16

CDC 40M: 39

CDC 80M: 39

PL 250K: 496

DADSLC: Sector length on this DAD.

Fixed at 410 characters per sector.

DADNDC: Number of cylinders on this DAD.

maximum, per disc type:

X1215: 200

X1216: 400

CDC 40M: 404

CDC 80M: 808

As sector length is at present fixed at 410 characters per sector, the maximum is:

X1215: 200

X1216: 400

CDC 40M: 103

CDC 80M: 103

DADFC1: Cylinder number of the first cylinder of this DAD.

DADSPG: Number of sectors per granule.

Fixed for all disc types at 8 sectors per granule.

DADSTA: DAD number is the sequence number of this DAD on the disc-volume. If 0, it is the first DAD on the volume.

DADSTB: address of the BTB, located in the BTB pool (T:BTB).

This word is used only if a temporary file has been assigned for this DAD.

### Logical File Description Table (T:LFT)

This table contains the information required by the system for performing an I/O operation on a logical file.

The table is of fixed length, with one entry for each file. The number of entries is fixed at system generation time and cannot be modified.

LFTORD	Z	LENGTH	ORDER										
LFTREAD			USER I/O ADDRESS										
LFTREC			USER RECORD AREA ADDRESS										
LFTLGT			USER REQUESTED LENGTH										
LFTRCT			AS (PCT) ADDRESS										
LFTLAS			AS (SCHEDULED LABEL ADDRESS)										
LFTMS1	A	P	S	D	U	C	T	R	S	M	I	W	C
LFTMS2			ASONT										
LFTDCT			DICT ADDRESS										
LFTSOT			ABSOLUTE ADDRESS OF SECTOR GRANTS OF THIS FILE										
LFTRC			RELATIVE NUMBER (9-bit) OF CURRENT SECTOR										
LFTSAC			ABSOLUTE ADDRESS OF CURRENT SECTOR										
LFTBAD			BLOCKING BUFFER ADDRESS / NBR. OF SECTORS OF DIR. ACC. FILE										
LFTNCS			DISPLACEMENT OF NEXT RECORD IN BLOCKING BUFFER										
LFTBUF			CURRENT BUFFER ADDRESS (FOR CURRENT OPERATION)										
LFTSEC			CURRENT SECTOR TO BE READ OR WRITTEN										
LFTORD			CURRENT ORDER TO BE PERFORMED										
LFTSTG			CURRENT STATUS										
LFTWD													
LFTNG													
LFTSLU													
LFTSLD													
LFTSLC			-										
LFTSLT													
LFTSLR													
LFTSLR													
LFTATT													

LPTORD: X=0, if this is the current entry.  
1, if this is the last entry.  
LENGTH is the length of one entry, in characters.  
ORDER is the I/O order given by the user in ECB word 0.

LPTEAD: contains the address of the user ECB.

LPTRBC: is the address of the user record area, where the Data Management routine must read or write the logical record.

LPTLGT: is the requested length as specified in the user ECB.  
If the requested length is greater than 8 sectors (i.e. 3200-8=3192 characters) incorrect length status will be returned and the record will be truncated at the first 3192 characters. In read mode, if the requested length differs from the record length given when the record was written, the shortest length will be moved from the disc to the record area. If this is equal to the requested length, no incorrect length status will result. Otherwise, when the record is truncated, incorrect length status will be set in the ECB. To avoid this, the user must read with the maximum record length in his request, and write with any length less than 3192 characters.

LPTORD: is used to save the I/O order given in the monitor request.

LPTPCT: contains the address of PCT of requested program.

LPTLAB: contains the scheduled label address. If no scheduled label is used, it contains 0.

LPTMD1 and LPTMD2 are two mode words defining the status of the LPT:

LPTMD1:

- A=1: LPT inactive, free for processing a request.
- =0: LPT busy. A user request has already been recorded and is not yet terminated.
- F=1: file is write-protected.
- S=1: source file.
- O=1: object file.
- C=1: load module.
- U=1: undefined file; contains user data.

These bits are  
always 0  
under DRTM

T=1: temporary file. This bit is set when the file code is assigned. It is used to find out whether the file can be extended and, for Data Management, to read the GRANTB to find the next granule address in case of sequential access.

R=1: random access is used for this file.

SE=1: sequential access is used for this file.

PN=1: the last write operation for this file was a write EOF, so the file is closed.

This bit is not modified by read and rewind operations. When an assignment is made for a catalogued file, this bit is also set to 1.

E=1: the current entry is already in queue. 0, if not.

W=1: write request.

  0: read request.

CD=1: consecutive file.

  0: non-consecutive file.

#### LPTMD2:

Bits 0 to 3 contain the number of file codes assigned to this LPT. At system generation time, it is initialized with the value 0. It is incremented each time a file code is assigned to this LPT and decremented when the assignment of a file code changes. Thus, when ASCNT becomes zero again, LPT is free to be used by the assign routine. Bits 4 to 15 give the number of sectors of the file, if it is a consecutive one.

LPTDOT: contains the address of the D:CT of the disc on which this file is stored.

LPTBOT: contains the address of the GRANTB sector of this file (second sector of first granule). It is initialized when an assign request is given and remains unchanged until the file code is deleted.

LPTSRC: is the relative sector number within the file.

LTFRAD: contains, for sequential files, the current blocking buffer address. For random access files, it contains the number of sectors in the file.

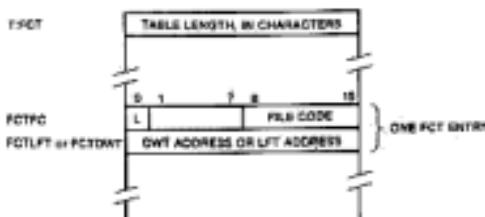
The words LPTBUF to LPTIK2 are used as work area by the logical I/O routines for performing their operations. LPTBUF, LPTSEC, LPTONG and LPTSRC are used by a subroutine which performs the physical part of an I/O operation and handles the request queue. The other words are used as work area.

LPTATT: is the Attach/Detach flag.

If the value is /8000, the file is not attached. If it is not /8000, it contains the PCT address of the program to which the file is attached.

#### File Code Table (T:PCT)

The file code table establishes the connection between a file code and a physical device or a logical file assigned to that file code. The table is of fixed length and created at system generation time. The user must at that point declare all the file codes used by the system, i.e. /EF, /E0, /02, /F0 to /F8. The others can be assigned when they are needed, but they may also be declared at system generation time.



Each entry consists of two words:

PCTPCT: contains the file code, in bits 8 to 15.  
L=1 if it is assigned to a logical disc file.  
L=0 if it is assigned to a physical device.

If  $L=1$ , the second word of the entry contains the address of the file description table LFT and is called FOLFT.

If  $L=0$ , the second word contains the address of the Device Work Table (DWT) and is called LFTDWI.

- For spare entries in the table, File Code is reset to zero. It can then be used for new file code assignments.
- The second word of an entry is reset to zero to indicate an assignment to NO device. In this case, bit L of the first word is also reset to zero.

#### Monitor Request Address Table (T:LRM)

This table contains the addresses of the routines which handle the different monitor requests. When a monitor request is made, control first goes to the I:LRM interrupt routine, which then finds the corresponding monitor request handler via this table, in accordance with the DATA number specified in the request. The table is generated at system generation time.

A table entry (each entry occupies two words) reset to zero indicates that the corresponding monitor request is not included in the system.

T:LRM		MAX NUMBER OF MONITOR REQUESTS
		DATA 0
LRM0		# [Entry number in core-dependent for LRM-0 monitor requests]
LRM0TR		# [ ]
		# [ ]
		# [ ]

} ENTRY FOR  
LRM 1 (INFORM)  
} ENTRY FOR  
LRM 2 (INPORT)  
etc.

There must be as many 2-word entries as the number in the location T:LRM specifies (not including the first 2 words).

The words in the entries have the following meaning:

-LKMPG: F=0: the monitor request is processed by a possibly non-resident program running at level 49 to 62.

The contents of LKMPG must be passed to the routine handling the monitor request via register A3, which is used to identify the entry point in the called program. The sign bit of A3 is set:

- to zero, if the request comes from the main program sequence;
- to one, if the request comes from a scheduled label sequence.

F=1: the monitor request is processed by a memory resident routine running at level 48.

LKMPTH: If F=1, this word points to the start address of the routine processing the monitor request at level 48.

If F=0, the monitor request is processed at a level equal to or greater than 49. In this case, LKMPTH points to the program name (PRNAME) in the Program Control Table of the program which must process this monitor request. If the monitor request is read only, the program is either D:USV1, D:USV2 or D:USV3. If it is memory resident, LKMPTH points to D:RMAC. In the latter case, the table T:RMAC must be updated with the appropriate entry point. (see also below).

#### Resident Monitor Request Table (T:RMAC)

In the DRTM a number of monitor requests are handled by routines running at level 48:

- I/O
- Wait for an Event
- Activate
- Exit
- Get Buffer
- Release Buffer
- Set Event
- Switch inside a Level

The other monitor requests are handled by system programs running at a level equal to or greater than 49. If these monitor requests have been declared read only, they are processed by the system programs D:USV1, D:USV2 and D:USV3.

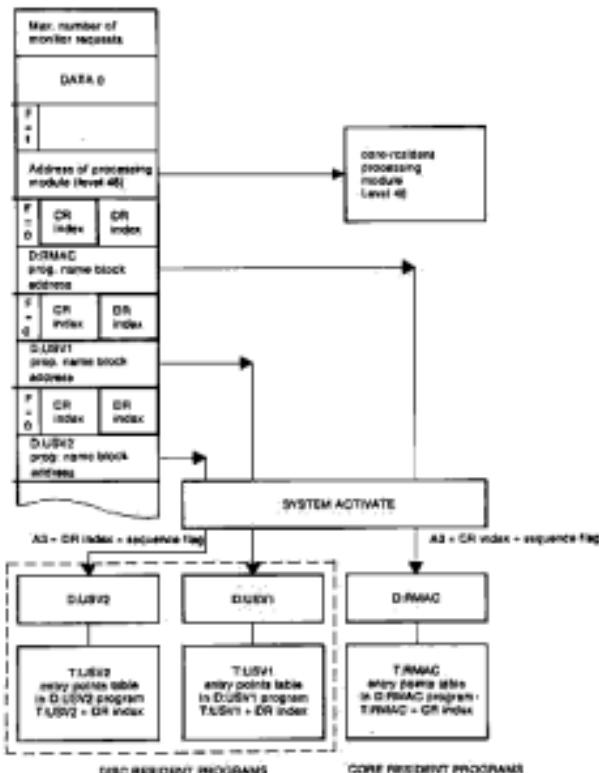
If they, or any number of them, have been declared as memory resident, they will be processed by a memory resident program D:RMAC. This program is link-edited with the monitor. A table must be created for the D:RMAC program to be able to branch to the correct entry point when one of these monitor requests is encountered in a program. This table must contain the start addresses of the selected memory resident monitor request handlers. The length of this table is variable, depending on the number of monitor requests declared memory resident, but each address must be stored in a specific location in the table:

1	DATA M:DFM (Data Management)
2	DATA M:HDUG (Data Management)
4	DATA D:ONTM (Connect Timer)
8	DATA D:DNTM (Disconnect Timer)
9	DATA D:GTTM (Get Timer)
10	DATA D:ATDV (Attach Device)
12	DATA D:DTDV (Detach Device)
14	DATA D:CNLV (Connect Level)
16	DATA D:DNLV (Disconnect Level)
18	DATA D:WTGT (Wait for given Time)
20	DATA ASSPRO (Assign File Code)
22	DATA DELPRO (Delete File Code)
24	DATA KEYPRO (Read Key-In)
25	DATA CANKEY (Cancel Key-In)

Thus, if the monitor request Disconnect a Level is required to be memory resident and any (or none) of the requests preceding it, but not the requests Assign, Delete, Read Key-in, the table must be 9 words long. If only disc logical I/O is required, only the first two words have to be generated.

Note: A similar table is required for the programs D:USV1, D:USV2 and D:USV3. It is described in the paragraph dealing with these programs.

## T-LKM



### TIMER MANAGEMENT TABLES

The following tables are connected with the management of the real time clock and the software timers.

#### Real Time Block

W:DAY	DAY (ASCII VALUE)
W:MONTH	MONTH (ASCII VALUE)
W:YEAR	YEAR (ASCII VALUE)
H:TIME	HOURS (BINARY VALUE -16)
M:TIME	MINUTES (BINARY VALUE -10)
S:TIME	SECONDS (BINARY VALUE -06)
S:TIME	TENTHS OF A SECOND (BINARY VALUE -10)
A:TIME	FIFTIETH OF A SECOND (BINARY VALUE -04)
C:TIME	NON-STANDARD CLOCK

C:TIME contains, if a non-standard clock is included in the system, zero minus the number of non-standard clock cycles during 20 msec.

#### Chaining pointers for programs connected to timers

H:POIN	Address of first block connected to timer H:TIME
	Address of last block connected to timer H:TIME
M:POIN	Address of first block connected to timer M:TIME
	Address of last block connected to timer M:TIME
S:POIN	Address of first block connected to timer S:TIME
	Address of last block connected to timer S:TIME
A:POIN	Address of first block connected to timer A:TIME
	Address of last block connected to timer A:TIME
B:POIN	Address of first block connected to timer B:TIME
	Address of last block connected to timer B:TIME
C:POIN	Address of first block connected to timer C:TIME
	Address of last block connected to timer C:TIME
R:POIN	Address of first block connected to hrs. time HH MM SS
	Address of last block connected to hrs. time HH MM SS

### V:FLAG

This is a table built by the I:RTC module.  
When a chain of programs connected to a timer must be scanned,  
I:RTC sets a flag which is to be detected by the timer management  
module M:CLK2 and reset by it.  
If one of the values in the table  $\neq 0$ , the chain connected to that  
timer must be scanned:

W:LOAD	Block chain scan chain No. H:TIME
	0:TIME
	1:TIME
	2:TIME
	3:TIME
	4:TIME
	5:TIME

### V:RESET

This block contains the values necessary to reset the real time  
block values.

W:RESET	24
	80
	00
	10
	0
	0

### Blocks built by Connect Program to Timer request (D:CNTR module)

#### Standard Block

Format before first activation  
of the connected program

Format after the first  
activation

CHANNING LINK
-MC
-PR
PROGRAM PCT ADDRESS

CHANNING LINK
-PR
-PR
PROGRAM PCT ADDRESS

NC: number of timer cycles before the first program activation  
PR: number of timer cycles between two activations of the same program.

Block for programs connected to absolute time

CHANNING LINK			
TIMER NUMBER	HOURS		PR
	MINUTES	SECONDS	
PROGRAM PCP ADDRESS			

T.N. is the timer number.

Block for programs waiting for a given time (D:WGT module)

CHANNING LINK			
NEGATIVE NC			
S	S	F	F
ADDRESS ECB FOR WHICH PROG WAITS			

Timer Numbering

- With declaration of the 20 msec. clock:
  - 0 - clock timer (20 msec)
  - 1 - tenths of seconds
  - 2 - seconds
  - 3 - minutes
  - 4 - hours
- With declaration of a non-standard clock:
  - 0 - non-standard clock
  - 1 - 20 milliseconds
  - 2 - tenths of seconds
  - 3 - seconds
  - 4 - minutes
  - 5 - hours

Note: If a Connect Timer request is given (via the System Command Language or in a user program) before the SCL command Set Clock (SC) has been given, the numbering accepted by SCL is the second one.

Input/Output operations involve a number of monitor modules, aside from the Device Work Table, Disc Control Table, Logical File Table and File Code Table already mentioned in the previous chapter. The most important of these are the I/O drivers, one for each type of peripheral.

Every I/O monitor request is handled by a driver. The user can, if he wishes, write his own drivers and insert them into the system. Such user-written drivers must interface with certain tables and modules which make up the I/O system of the monitor. The necessary information which must be taken into account is described in the following paragraphs.

#### TABLES

Three tables are used by the I/O system to know the necessary details about the peripheral devices used:

- File Code Table, which enables the user to assign a file code (a logical number) to a device and use this file code in programming. See Chapter 2 of this part.
- Device Work Table: contains all parameters about the peripherals which are necessary for the I/O system to know. See Chapter 2 of this part.
- Controller Status Table or Disc Control Table: one for each device control unit. The CST is a free format table of unspecified length in which the user can put information for use by his I/O driver for non-disc devices. The I/O system knows one word of this table (referred to by word EWT-32) which contains the status of the control unit. The first bit of this status word is reset to 0 (-busy) as soon as an I/O operation for a device is started and set to 1 when it is terminated (!-free).  
For disc, Disc Control Tables are created. The layout is given in Chapter 2.

## I/O SYSTEM

The physical I/O system can be divided into four main parts:

- The I/O request module (IORM)
- Driver
- End of I/O module (ENDDIO)
- Service routines (COMMIO and N:RETR)

### IORM

When the user has given an I/O monitor request (LEM !), this module will receive control, via the I:LEM module and T:LEM table.

First this module

- checks the validity of the request
- increments the event count (PCT)
- computes the Device Work Table address
- checks whether the device control unit is busy or free (via DWT+52). If the unit is busy, the user is put in wait.
- checks whether the device is busy or not. If it is busy, the user is put in wait.

Then IORM sets some parameters in the DWT and in the controller status table:

- controller is set to busy (bit 0 = 1)
- retry flag in controller status (bit 7) is set for non-disc devices
- any scheduled label parameters are set in DWT+28, DWT+30, DWT+24 and DWT+26 are set to zero
- the user Event Control Block is initialized, i.e. the left character of ECB 0 and ECB+8 (status) are set to zero
- the I/O order (A7) is analyzed which may result in initialization of some DWT location.

At the end of this process a branch is made to the I/O driver concerned, the address of which is found in DWT+6.

### Driver

On entry into a driver, these registers must contain the following parameters:

A4: User order without wait bit

A7: User I/O order

A8: User ECB address

A6: DWT address.

The DWT must contain:

Words 0 to 8: not modified.

Word 10: User ECB address

Word 12: User buffer address

Word 14: Requested length (may be non-significant)

Word 16: Zero

Word 18: I/O order (without wait or retry bit; may be non-significant, e.g. skip orders)

Word 20: Retry bit

Word 22: not significant.

The first part of the I/O driver performs the I/O initialization.

An exit to the C:WAIT module is made with the user order (with

wait bit!) in A7 and the user ECB address in A8.

The second part of the driver is made up by the interrupt routine:

- interrupt sequence generated by SYSGEN:

for single unit controllers: STR A1, A15

                          |

                          STR A8,A15

                          |

                          LDKL A6,DWT address

for multi-unit controllers: STR A1,A15

                          |

                          STR A8,A15

                          |

                          LDKL A6,controller status or  
                          DCTDH (DCT) address

- the exit from the interrupt routine is made to R:TURN (in ENDIO module) if the I/O is not yet finished or to R:TUR4 (in ENDIO) if it is finished. The exit to R:TUR4 must be made with A2 containing the I/O status and A6 containing the DWT address.

- for Retry procedures a branch must be made:

ABL M:RETR

In this case, the calling sequence is:

A6: DWT address

A2: Status to be printed

A1: Retry flag (0, if retry)

A5: Hardware order (contents of register with which CIO Start will be sent).

Notes: The I/O processor control parameters must have been reinitialized.

#### ENBIO

The calling sequence to be used by user I/O drivers for this module is:

A6: DWT address

A2: I/O software status (see Part 1, chapter 11)

ABL R:TUR4

Moreover, ENBIO must find the following parameters set in the DWT:

- effective length (word 16)
- ECB address (word 10)
- Scheduled label parameters (words 28, 30).

Then, the following functions are performed by the ENBIO module:

- controller status is updated (bit 0 is reset to 0: free)
- the event count is decremented
- the event character in the user ECB is updated (bit 0=0)
- the effective length is set in the user ECB (word 3)
- the software status is set in the user ECB (word 4)
- a branch is made to the dispatcher (M:DISP), with register A5 containing the PCT address and A6 the scheduled label address, if any.

## Service Routines

### OCOMIC

This routine has two functions: executing I/O hardware instructions and building the I/O processor control words and putting a requesting program in Wait State. Below the calling sequences for the functions are listed.

Note, that in those sequences the instructions INH - MCR - ABL must be given in the order specified:

- CIO Start: A6: INT address  
A2: hardware order  
A3: return address  
INH  
STR A1,A15  
STR AB,A15  
ABL S:TIO  
Return: - condition register set  
- inhibit mode.
- CIO Stop: A6: INT address  
A3: return address  
INH  
STR A1,A15  
STR AB,A15  
ABL H:L:TIO  
Return: - condition register set  
- inhibit mode.
- OBR: A6: INT address  
A3: Return address  
A1: word to be output  
INH  
STR A1,A15  
|  
STR AB,A15  
ABL O:TIO  
Return: - condition register set  
- inhibit mode.

- INR:  
     A6: DWT address  
     A3: return address  
     INE  
     STR A1,A15  
     STR AB,A15  
     ABL I:NRIO  
     Return: - condition register set  
             - inhibit mode  
             - word or character to be inputs in A1.

- SET:  
     A6: DWT address  
     A3: return address  
     INE  
     STR A1,A15  
     STR AB,A15  
     ABL S:SET  
     Return: - condition register set  
             - inhibit mode  
             - status in A2.

- the I/O processor control words are built as follows:  
     A1,A 2: two-word contents of control word  
             (the user must give this as:  
                 - length (in 2's complement) + 4 control bits  
                 - ending buffer address  
             This is converted by routine M:TEX to  
                 - length (in 2's complement) + 4 control bits  
                 - beginning buffer address.  
     A6: : DWT address  
     A3 : return address  
     INE  
     STR A1,A15  
     STR AB,A15  
     ABL M:TEX

The requesting program is put in Wait, if this is necessary, as follows:

    A7: User I/O order  
     AB: User ECB address  
     ABL C:WAIT

### M:RETR

This module is involved when the operator wants to retry an I/O operation.

The calling sequence is:

- A1: Retry flag (0, if I/O must be retried)
- A2: Status to be printed
- A3: Request (hardware order for CIO Start)
- A5: DWT address

ABL M:RETR

The I/O processor control words must have been reinitialized.

Then the following actions are taken:

- activate D:OCOM with message formatted in buffer
- if the I/O operation must not be retried, branch to ENDIO with the value /8000 in register A2
- if the I/O operation must be retried, test whether bit 7 in the control unit status is set:
  - a) if not, set bit 7 and go to the dispatcher
  - b) if set, (i.e. M:RETR must be called after an RD command by the operator), reset bit 7 and then branch to ENDIO with the value /8000 in register A2.

When a Retry message is sent by the operators

(D:OCOM is normally connected to level 49)

- compute CIO and execute it
- Reset address of buffer in DWT
- reset effective length to zero in DWT
- reset DWT words 24 and 26 to zero
- reset the effective length and status in the user ECB to zero
- for RD messages: do not reset bit 7 of the control unit status, but EXIT
- for RY messages: reset bit 7 of the control unit status, then EXIT.

	Page
INIMOS . . . . .	2-72
D:LIBER . . . . .	2-74
M:DISP . . . . .	2-76
I:PPAR . . . . .	2-90
I:BTC . . . . .	2-94
I:LEM . . . . .	2-98
M:ICBN . . . . .	2-102
M:WAIT . . . . .	2-106
M:EXIT . . . . .	2-108
M:GBUF, M:FBUF . . . . .	2-113
D:CNTR . . . . .	2-116
D:DMTR . . . . .	2-120
M:ACT . . . . .	2-122
M:SWTC . . . . .	2-124
D:ATDT . . . . .	2-126
D:CTIM . . . . .	2-130
M:BSRT . . . . .	2-132
D:CHLV . . . . .	2-134
D:DMLV . . . . .	2-136
D:WGTR . . . . .	2-138
ASGPRO . . . . .	2-140
DELPRO . . . . .	2-147
KEYPRO . . . . .	2-150
CANKEY . . . . .	2-152
D:SWP . . . . .	2-154
M:DFM . . . . .	2-156
D:CTIM, D:OCOM . . . . .	2-158
M:DMA . . . . .	2-162
M:DML . . . . .	2-164
M:DOC . . . . .	2-166
P:BLK . . . . .	2-172
M:AOO . . . . .	2-174
I:TRAP . . . . .	2-178

## NUCLEUS INITIALIZATION PROGRAM (INIMON)

### Calling Sequence

This module starts the monitor when it has just been loaded from disc. Therefore it is entered from the IPL without any parameters.

### Work Areas and Tables

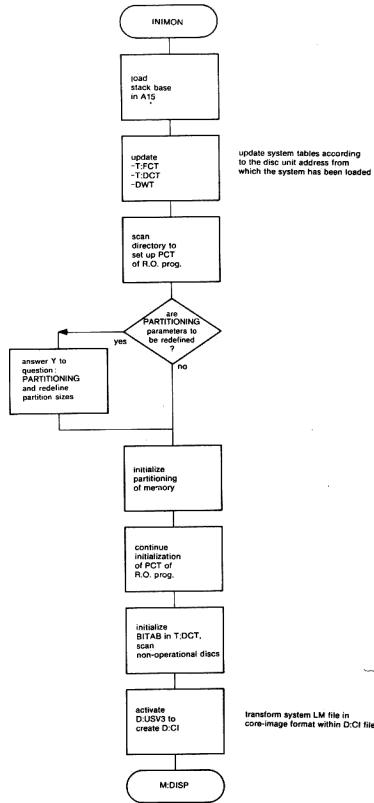
INIMON initializes:

- the disc tables: T:PCT, DWT, T:DCT
- memory partitioning: GVT
- the program control tables: T:PCT.

Besides, it uses one buffer (GATEBUF, 205 words) to read the directory from disc.

### Input/Output File

The INIMON module outputs messages on the typewriter, if necessary, and reads the directory from disc. It also activates D:USV3 which will create the D:CI file.



## LOADER (D:LDER)

### Calling Sequence

The loader is activated as any other software level program by the dispatcher or the command language.

Upon entry, register A3 must point to the PCT of the program which must be loaded. Therefore the PCT must already have been initialized by the command language before the loader is activated.

### Work Areas and Tables

The loader requires about 250 words in the dynamic allocation area to read the program which is to be loaded. As soon as loading has been completed, this work area is released and the PCT of the program is updated. This work area is not allocated for core image programs (read only and swappable programs).

### Input/Output Files

The loader has to use the disc driver directly to load the program, for Disc File Management is optional in this system. The input file of the loader is therefore the disc unit file code (from /F0 to /FF). Error messages are output via the operator's typewriter (file code /EP).

### Memory Layout

The loader must be resident in memory and be link-edited with the supervisor. It can be executed at any software level, without restriction.

### Functional Description

The principal function of the loader is to load a program from disc into memory. The program must be stored on the disc in the standard load module format.

In memory, the program can be stored in any area: Memory Resident, Read Only, Swap or Background Area. The program's PCT must have been initialized before the loader is called. The loader therefore

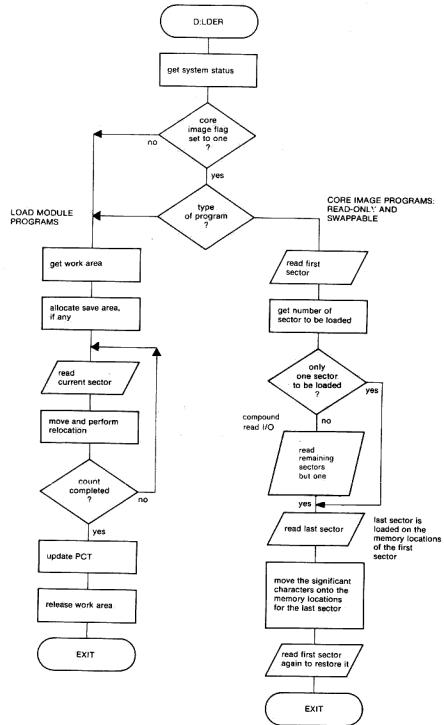
does not have to search for the program on disc, but read the sector GRANTB (Granule Table) of the load module file, to find the absolute sector number where it must start reading. Having read a sector of the program into the dynamic allocation area, the loader will move instructions and data from the dynamic area to the area where the program is stored. Relocation is done at the same time. On completion of the loading operation, the PCT is updated and the buffer in the dynamic allocation area is released.

The function of the Core Image Loader is to load a program from the D:CI file. The program is stored on disc in core image format.

#### Error Messages

In case of a disc I/O error during loading, one of the following messages is output:

- during loading of a load module:  
LDER <program name>
- during loading of a core image module:  
CI ER <program name>



## M:DISP (DISPATCHER)

### Calling Sequence

A6: Scheduled Label address, if a scheduled label is to be dispatched. If not: 0.  
A5: Address of the PGT entry to which the scheduled label applies.  
(Irrelevant if A6 = 0).  
A8L M:DISP

The dispatcher can also be called indirectly, via the third word of the Communication Vector Table; this allows a program, which has been link-edited separately from the dispatcher, to branch to it.

It is assumed that the A15 stack contains P-registers, Program Status Word and registers A1 to A8 of the interrupted program.

### Work Areas and Tables

- A15 stack, to save and restore the program context.
- T:SLT (Software Level Table) } of the program it has to start
- T:PGT (Program Control Table) }
- Pointers to the program currently getting CPU time and the programs currently in the Read Only, Swap or Background Area are modified, as well as:
  - T:PGT , which points to the first PGT of the PGT Pool;
  - P:CUR , which contains the PGT address of the current program;
  - P:ROPC , which contains the PGT address of the read only program currently occupying the read only area;
  - P:SWAP , which contains the PGT address of the swappable program currently occupying the swap area;
  - T:SWP , is the address of a 16-word table in which each word contains the PGT address of the next swappable program to be initialised on a certain software level when its time slice comes up;
  - CVTTSC , which is a time slice counter initialized by the loader and updated by the real time clock package;
  - D:PGTB , which contains the PGT address of a background program waiting to be loaded.
- P:DISP , a flag which is set by:
  - Activate: only if the activated program is found to be inactive
  - Exit: in every case
  - Wait: only if the Set Event has not yet occurred
  - Set Event: only if a program is waiting on the given ECB address
  - Scheduling a Label: only if it is the first

- one for the current program
- Switch inside levels: only if the request is effective, i.e. if another PCT waiting for CPU time is found waiting on that level.

#### Input/Output Files

None.

#### Memory Layout

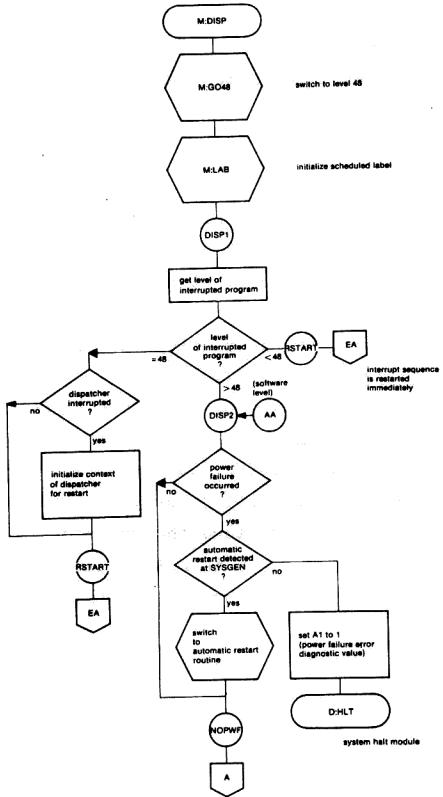
M:DISP is always resident in memory and belongs to the supervisor part of the BRTM.

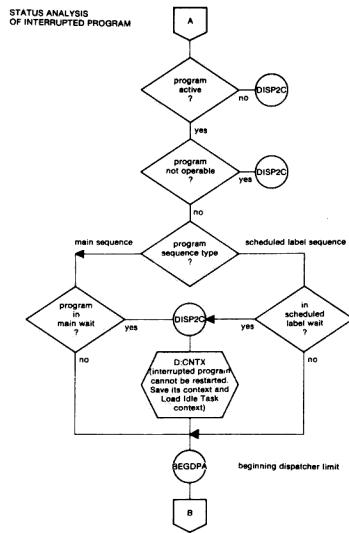
#### Functional Description

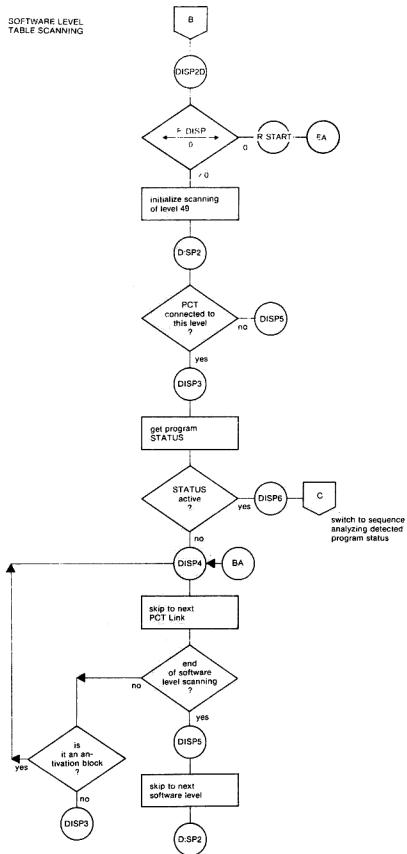
The dispatcher is called by any system routine running at a level equal to or below 48, especially when an event, such as I/O, has occurred and a scheduled label must be started.

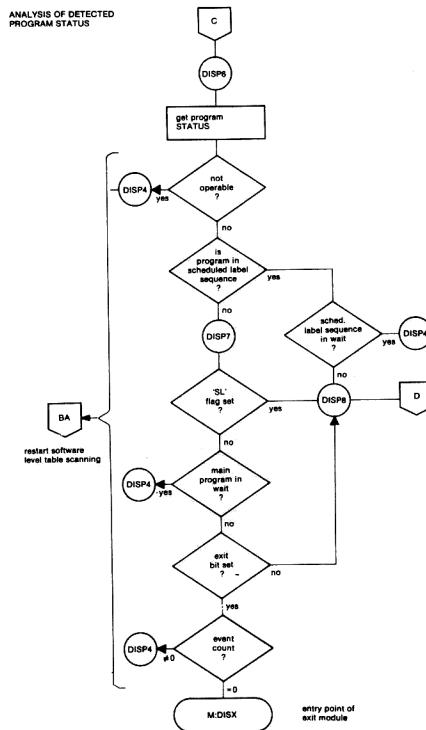
Its main function is to determine which program must get CPU time. If the interrupted program was running at a level equal to or below 48, control is returned to it immediately. Then the P:DISP flag is checked: if it is not set, control is returned also to the interrupted program. Otherwise it will scan the Software Level Tables (T:SLT) and Program Control Table (T:PCT) to find the highest priority active program. If this program is already in memory, it will restore the registers from the stack and start it immediately. If it is not in memory, M:DISP will activate the loader to load the program, and look for the highest priority program.

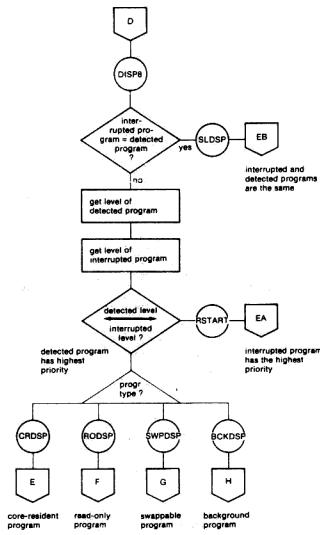
If no action is to be taken for a user program, control is automatically transferred to the Idle Task.

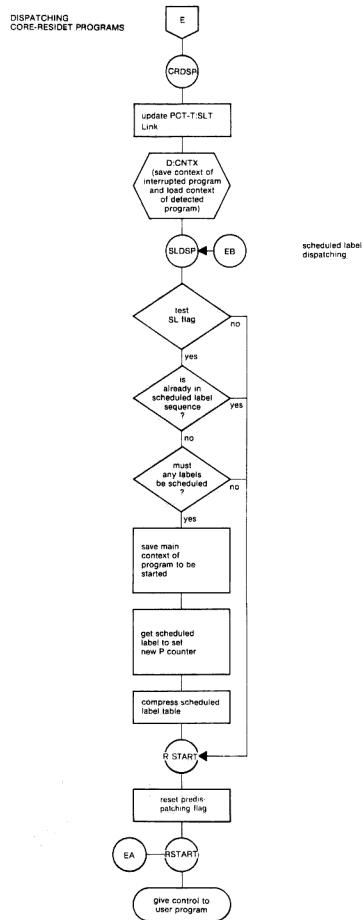


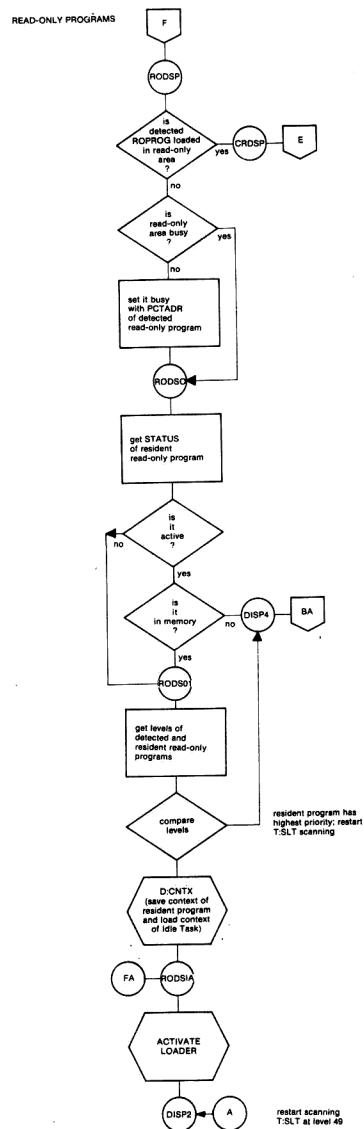




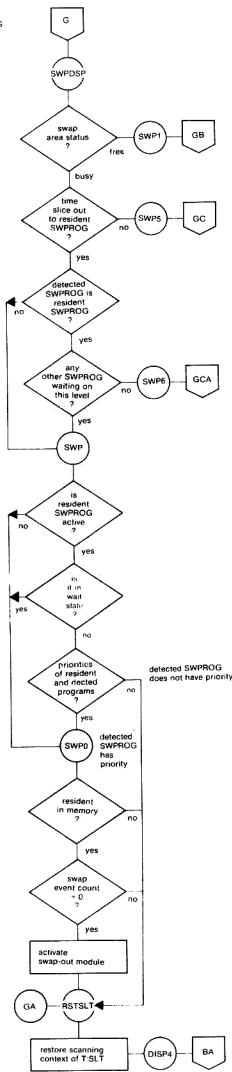


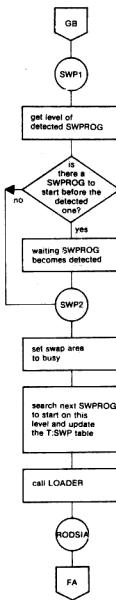


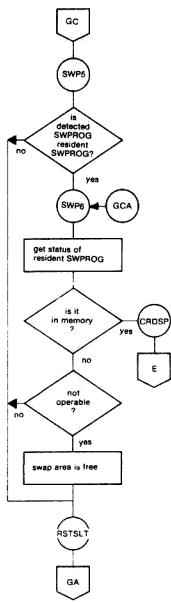




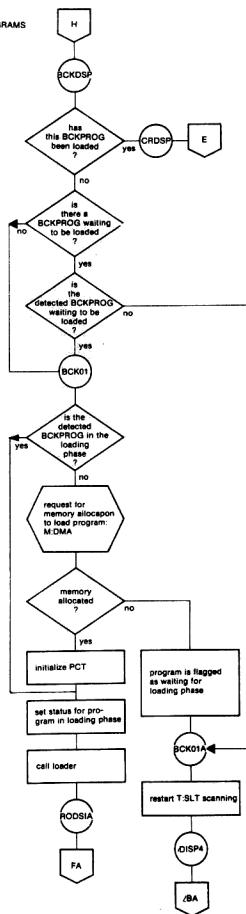
## DISPATCHING SWAPPABLE PROGRAMS

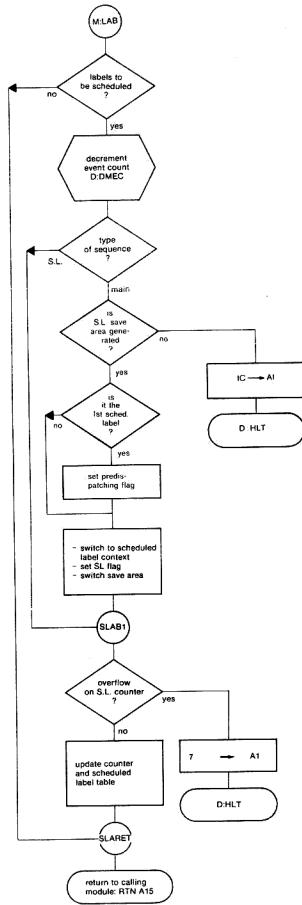






DISPATCHING  
BACKGROUND PROGRAMS





## I:PPAR (POWER FAILURE - AUTOMATIC RESTART ROUTINES)

### Calling Sequence

This routine is called every time there is a power failure/automatic restart interrupt, and the machine key is in the LOCK position.

Entry Points: I:PPAR (from interrupt)  
I:ARES (from dispatcher)

### Work Areas and Tables

This module uses A15 Stack and CTV to denote power failure. For automatic restart, the DWT and control unit status tables are updated, together with the user ECB.

### Input/Output Files

None.

### Functional Description

The three optional routines mentioned in the flowchart for this module:

U:PPAL

U:ARES

U:RST

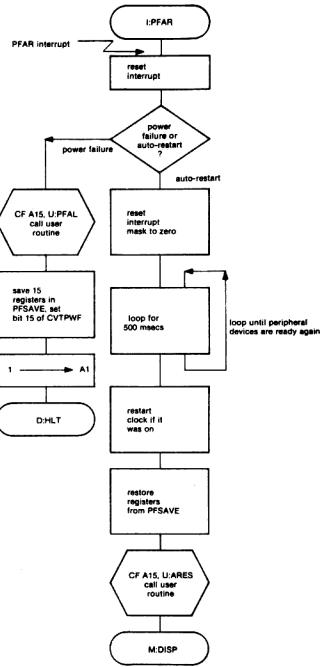
must be provided by the user. If they are not used, they must be simulated with an RTW A15 instruction.

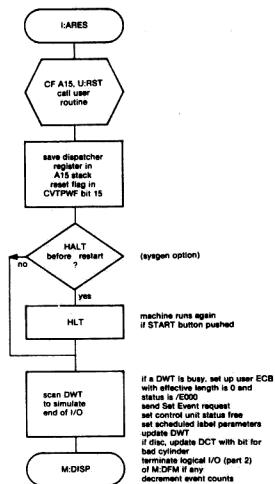
- When a power failure interrupt has been recognized and the interrupt has been reset, the system calls the user power failure routine, before saving the i5 registers:  
CF A15, U:PPAL
- When the automatic restart interrupt has been recognized, the system restores the i5 registers and the user automatic restart interrupt routine:  
CF A15, U:ARES
- The routines U:PPAL and U:ARES run in inhibit mode, at the level

of the interrupt. They can be completed by a restart routine running at level 48 and called by the dispatcher:

CP A15, U:RST

This restart routine is called before the system simulates the end of all pending I/O operations. These will all receive the status /E000 (power failure occurred), so that they must be requested again, regardless of the device.





I:RTC (REAL TIME CLOCK INTERRUPT ROUTINE)

Calling Sequence

This routine is called by hardware interrupt.

Work Areas and Tables

I:CPIS: a word in the Communication Vector Table giving the

pulse rate.

V:FLAG: a flag vector indicating the timer to be scanned.

V:REST: a value vector to reset the timers.

H:TIME: start address of the timer block.

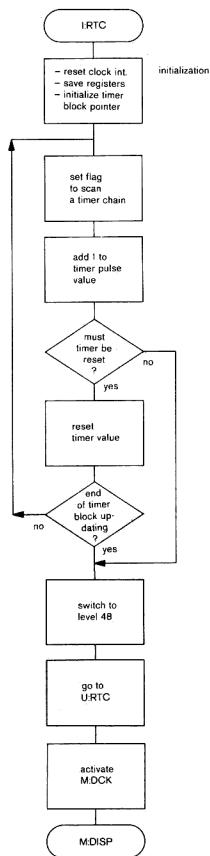
Input/Output Files

None.

Functional Description

I:RTC is the real time clock driver. It is started by a real time clock interrupt and normally runs at level 2. It starts by updating the timer block and setting flags in the V:FLAG vector in case a chain of program blocks connected to a timer must be scanned, analyzed and updated. This task is managed by the module M:DCK which runs at level 49.

Then, I:RTC enters the U:RTC module, activates the module M:DCK and gives control to the dispatcher.



## USER REAL TIME CLOCK ROUTINE (U:RTC)

### Calling Sequence

This module is called through the instruction RTW A15, U:RTC. It must, therefore, contain at least one entry point named U:RTC.

### Work Areas and Tables

As this is a user module, it contains none from the system point of view.

### Input/Output File

None

### Memory Layout

Not applicable.

### Functional Description

The U:RTC routine is called on each real time clock interrupt by the I:RTC module.

The interfaces with the system are as follows:

- Input: control is given at level 48 and in inhibit state
- Output: the user must return control to the I:RTC module through an RTW A15 instruction.

The module is selected as follows:

A standard U:RTC module, which executes only an RTW A15 instruction, exists already in the standard library.

If the user wants to replace this standard module by a routine of his own, he must include this routine during the link-edit of system generation, before link-editing the standard library. This procedure is described in the Appendix on System Generation.

## MONITOR REQUEST HANDLER (I:LEM)

### Calling Sequence

This routine is activated by an LKM interrupt.

The parameters of the monitor request constitute the calling sequence, where the DATA word following the LKM instruction identifies the function to be performed and registers A7 and A8 contain the required parameters. See part 1.

### Work Areas and Tables

No work area is necessary in the dynamic allocation area, unless a monitor request is made for which the processing routine runs at a level equal to or above 49.

To check the validity of the request and be able to branch to the required processing routine, the T:LEM table is consulted.

### Input/Output Files

None.

### Memory Layout

Not applicable.

### Functional Description

Upon interrupt, the LKM handler will save the program context in the A15 stack and check the validity of the request.

If the required function is performed at level 49, the LKM handler branches directly to the processing routine and then the processing routine will switch to level 49 as soon as possible. If the function is performed at a software level, the calling program will be put in wait state, the event counts incremented and the processing routine activated.

Before the processing routine is called, the LKM handler has to communicate parameters via the following registers:

A5: PCT address

A6: Scheduled label address, if any

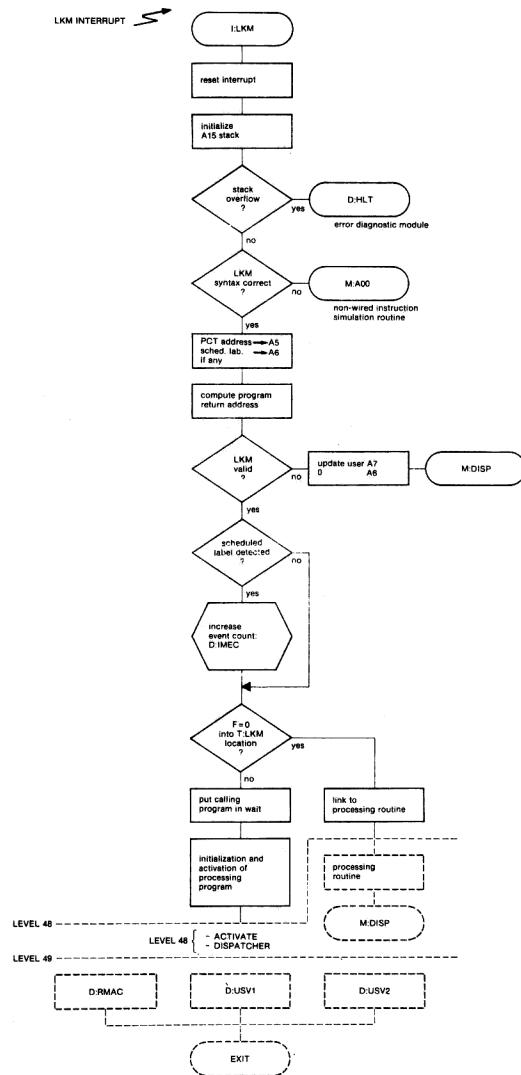
A7: User A7 parameter

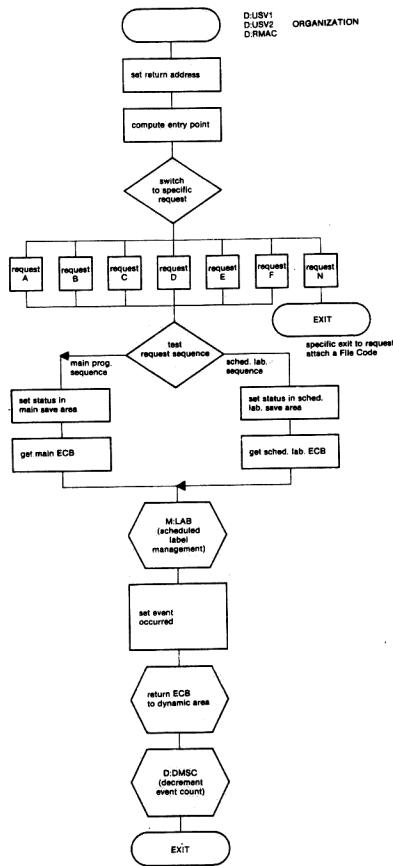
A8: User A8 parameter.

If the processing routine runs at level 48, the A15 stack contains the user context. If the processing routine is one of the following:

- D:USV1
- D:USV2
- D:USV3
- D:RMAC

A5 contains the second word of the corresponding entry in T:LEM. If the function is requested by a scheduled label sequence, the sign bit of A5 is set to 1, and if it is requested from a main program sequence it is reset to zero.





### INPUT/OUTPUT HANDLER (M:IOHM)

#### Calling Sequence

M:IOHM is called by the IHM handler (I:IHM).

The parameters are given in registers A5, A6, A7 and A8:

- A5: PCT address
- A6: Scheduled label address, if any
- A7: User A7 parameter
- A8: User A8 parameter

#### Work Areas and Tables

No work area is required but all I/O tables are used:

- T:PCT (File Code Table)
- T:DMT (Device Work Table)
- T:LPT (Disc Logical File Table)
- T:DCP (Disc Control Table)

#### Input/Output Files

None.

#### Memory Layout

The I/O supervisor and all required I/O drivers are always resident in memory and must be link-edited with the supervisor part of the monitor.

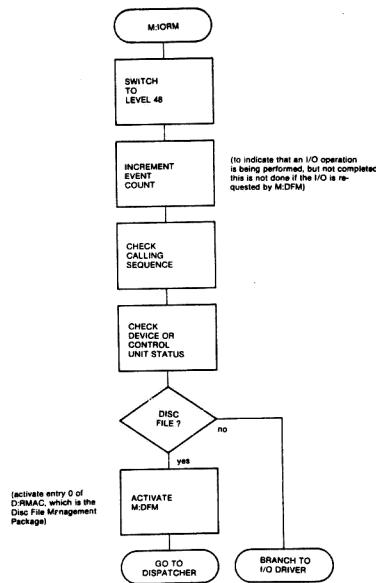
#### Functional Description

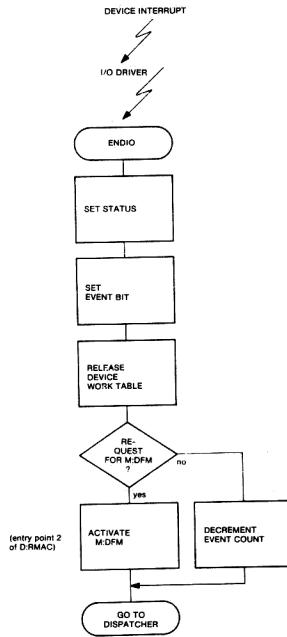
The I/O Supervisor can be regarded as combining three functions:

- pre-processing
- processing
- post-processing

Pre-processing is done by the M:IOHM module by analyzing the I/O requests, checking for device availability and verifying the validity of the user parameters.

The actual processing is done by the I/O driver. It will send the I/O command to start the operation, receive the interrupt and process any error recovery. When the operation is completed, the driver will ask for the status of the operation, perform all code conversion and formatting and finally call the ENDIO routine. ENDIO performs all post-processing functions. It manages the Device Work Table and is the interface with the user program.





## M:WAIT (WAIT FOR AN EVENT - LKM2)

### Calling Sequence

A5: POF address  
A6: Scheduled Label  
A8: Event Control Block address

ECB format:

X = 1: event has occurred.

Entry Points:

- M:WAIT; M:PUTW; M:WANI  
(wait without reinitialization of user request)
- M:WAIC  
(wait with reinitialization of user request)
- M:WAEV  
(same function as M:WAIC but pointed to through GVTWAT in order to be used by the read-only system programs)

### Work Areas and Tables

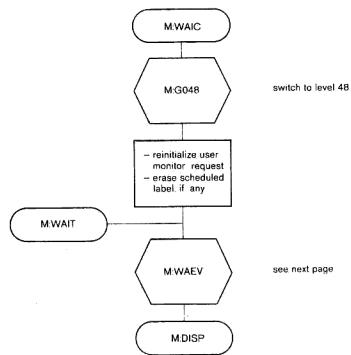
None.

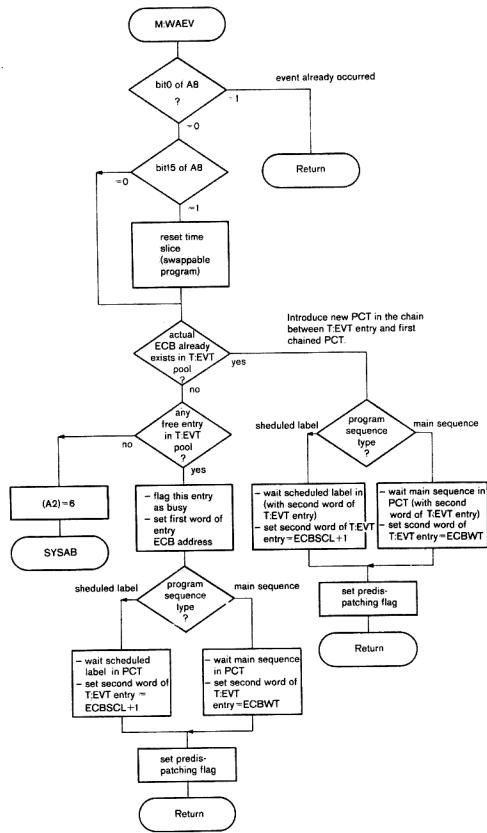
### Input/Output Files

None.

### Functional Description

This module processes the Wait for an Event monitor request (LM2). There are different entry points to this module, depending on whether the request must or must not be reinitialized. A wait is done on the occurrence of an event which is notified by the setting of bit 0 of the ECB word of the program or scheduled label in which the event occurs.





## M:EXIT (EXIT-LKM)

### Calling Sequence

A5: PCT address of program requesting exit.

Entry Points: - M:EXIT (entry point from I:LEM: user request)  
- M:DIX (entry point from dispatcher; used when exit was delayed, because event count was not yet zero).

### Work Areas and Tables

T:SLT (Software Level Table).

PCT entry of program requesting exit.

Save area of this program.

Dynamic memory allocation area.

### Input/Output Files

None.

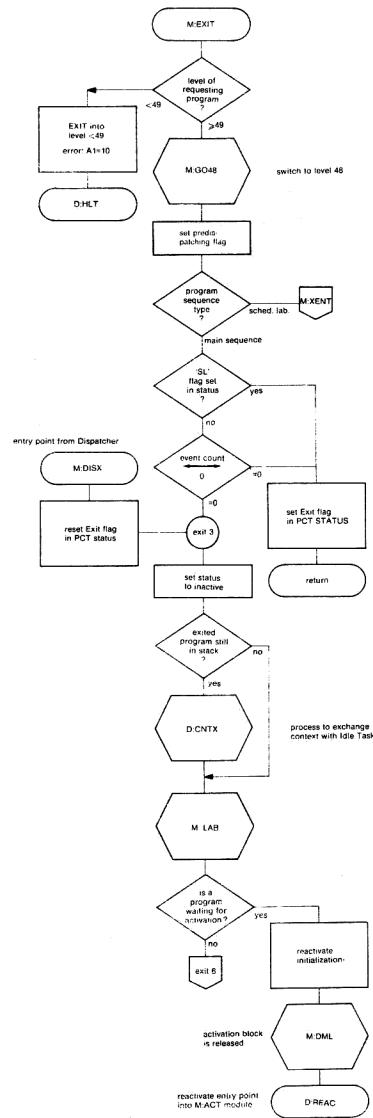
### Functional Description

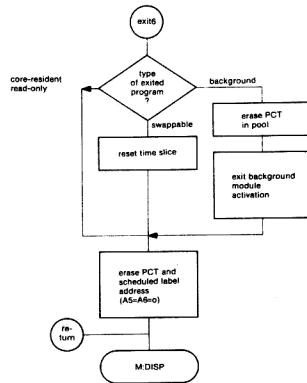
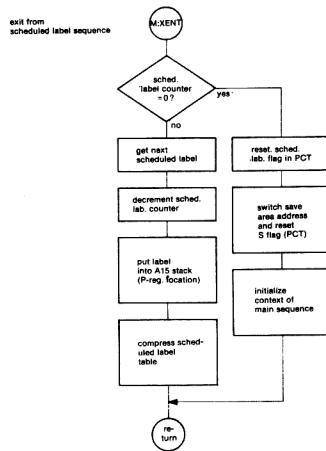
This module processes the Exit monitor request (LEM3). If all I/O is terminated and there are no more labels to be scheduled, the main program exits:

- the program is put in inactive state;
- the PCT chain is checked to see if there are any stacked activate requests for this program, in which case it is reactivated. If the program is not yet finished, i.e. event count  $\neq$  0, the exit bit is set in PCT word 0 (Status) and control is given to the dispatcher. When the event count becomes zero, the dispatcher returns control to the M:EXIT module (via M:DIX) and the main program exit is performed.

If the program which exits is a background program, part of the exit process is performed by a specific sequence in the D:USV2 module (D:EDCK) which releases all the resources allocated to this background program (background area, save area, PCT).

If the program which exits is a swappable program, the time slice counter is reset to zero.





M:GBUF, M:PBUF (GET BUFFER - LROM/RELEASE BUFFER - LEMS)

Calling Sequence

- M:GBUF:

A7: Length of requested buffer, in characters.

If zero is specified, the memory size in characters is returned in the A7 register (If memory size = 32k, 0 is returned).

Entry Point: M:GBUF

- M:PBUF:

A14: address of the buffer previously reserved for the calling program by a Get Buffer request.

Entry Point: M:PBUF

Work Areas and Tables

T:CVT: Communication Vector Table.

PGT entry of calling program.

Dynamically Allocated Memory Area (buffer).

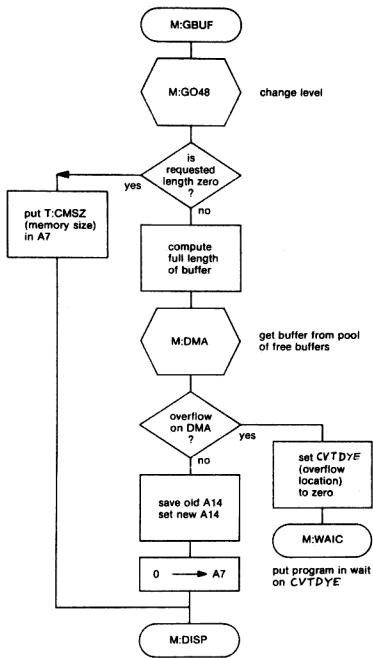
Input/Output Files

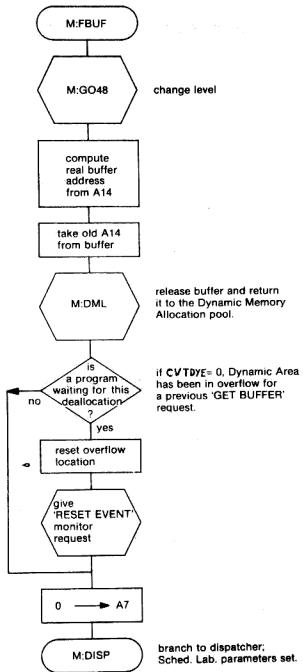
None.

Functional Description

These modules handle the dynamic allocation of memory space for user programs.

If there is no space available when a request is made, the user program is put in wait state (with reinitialization of the request). When memory space becomes available again, i.e. after a Release Buffer request has been given, the user program is restarted to repeat the request for a buffer.





D:CNTRM (CONNECT A PROGRAM TO A TIMER - LIM610)

Calling Sequence

Entry Point: D:CNTRM

A7: address of program name block.

AB: address of 2-word parameter block, which may be of one of the following two formats:

Standard Connection:

3	1	3	4	5	15
0	TIMER NUMBER	W	0	PULSE RATE (PR)	
NUMBER OF CYCLES BEFORE FIRST ACTIVATION (NC)					

where

TIMER NUMBER is the timer to which the program specified via A7 must be connected.

PR is a value from 0 to 2047.

NC is a value from 0 to 32677.

Bit 4 is set to 0 by the system (calling level  $\neq$  48) and its use is reserved to the 'Wait for a given Time' module (D:WGT).

Absolute Time Connection:

3	1	3	4	5	9	15
1	TIMER NUMBER	HOURS		PULSE RATE		
MINUTES			SECONDS			

where

PR is a value from 0 to 127.

The program specified via A7 is connected to the absolute time chain. At the time defined by the user (HH MM SS), it is started, disconnected from this chain and connected to the chain on the timer defined by the user in bits 1 to 3. This is managed by the M:DCK module.

Note: When PR=0, only one program activation takes place and the program is automatically disconnected from the timer. This is managed by the M:DCK module.

#### Work Areas and Tables

##### H:POIN Chain Pointer

When a connection is requested from D:CNTR, a 4-word block is automatically reserved in the dynamic allocation area. The format of such a block is as follows:

###### - Standard Connection

CHAINING LINK
NEGATIVE PR
POSITIVE PR
PROGRAM RET ADDRESS

After the first program activation, the block format is as follows (unless PR was 0, in which case automatic disconnection will follow):

CHAINING LINK
POSITIVE PR
NEGATIVE PR
PROGRAM RET ADDRESS

where the third word contains the PR value as updated by M:DCK.

- Absolute Time Connection

CHAINING LINK		
U.H.	HOURS	PULSE RATE
MINUTES	SECONDS	
PROGRAM PCT ADDRESS		

After the first program activation, this block is reinitialized in standard format.

- 'Wait for a Given Time' Connection

CHAINING LINK			
NEGATIVE NO			
F	F	F	F
FOR FOR WHICH PROGRAM IS WAITING			

Input/Output Files

None.

Functional Description

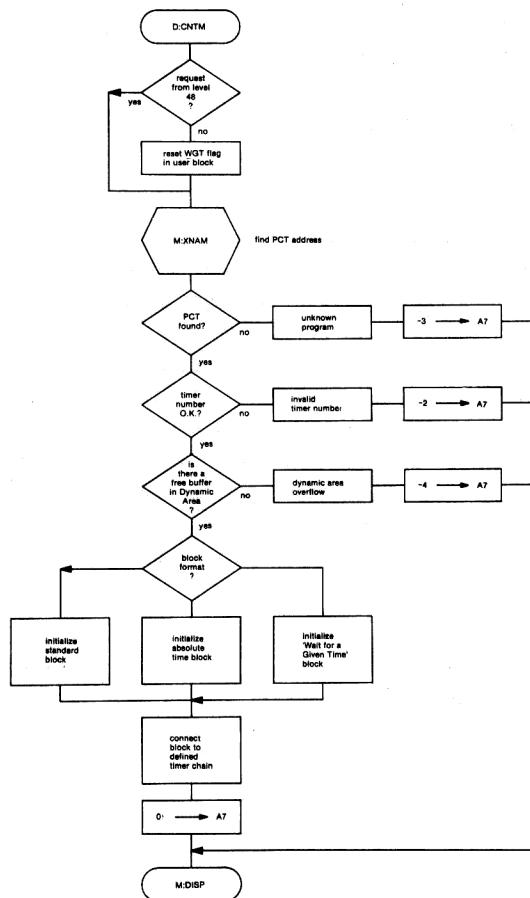
This request builds and initializes a timer block, to establish a link between a timer and the calling program. It is started by the LWM request, which activates D:RMAC if it is core resident or D:USV2 if the request handler is disc resident.

First, D:CNTRM checks the type of request in order to set or reset the WOT flag in the two-word parameter block. Then program name and timer number are checked and a request is given for a four-word block in the dynamic allocation area via the M:DMA module. If an error is detected at this point, it is set in the A7 register. If not, a timer block is initialized and connected to the timer chain defined by the calling block (see Work Areas and Tables).

At the end of this process D:CNTRM returns to D:RMAC or D:USV2 with the Status set. One of the following values is returned to the

user in the A7 register:

- A7 = 0: connection accomplished.
- 2: timer does not exist.
- 3: request to an unknown program.
- 4: dynamic area overflow.



D:DNTM (DISCONNECT A PROGRAM FROM A TIMER - LKM11)

Calling Sequence

A7: address of program name block

A8: timer number.

Entry Point: D:DNTM

Work Areas and Tables

H:POIN: Chain pointer

Input/Output Files

None.

Functional Description

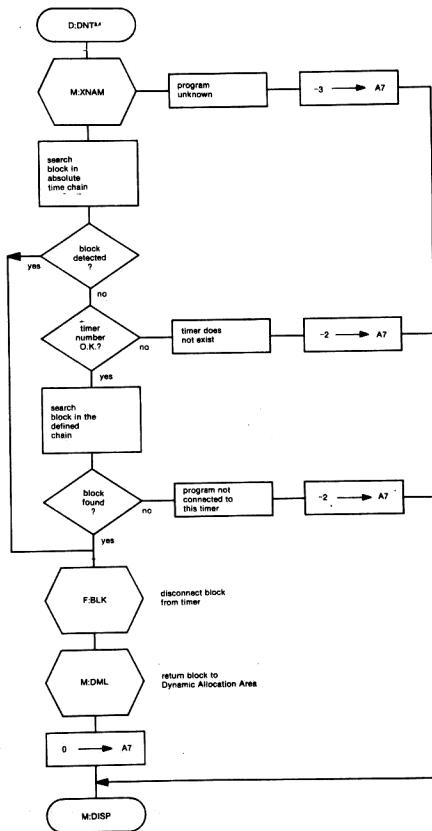
D:DNTM is started by LKM request, activating D:EMAC if the request handler is core resident or D:USV2 if it is disc resident. First, the address is found of the PCT entry for this program. If the M:NAME module cannot find this address, the program does not exist. If the address is found, D:DNTM scans the absolute time chain to look for the program's timer block. If no block is found here, the timer chain specified in register A8 is checked. When the block is found, it is disconnected from the chain and returned to the dynamic allocation area.

If an error is found, this is indicated in the A7 register of the calling program:

A7 = 0: disconnection accomplished

--2: timer does not exist or the program is not connected to the specified timer.

--3: the program does not exist.



### M:ACT (ACTIVATE - LHM12)

#### Calling Sequence

- If request comes via an LHM interrupt:  
A7: address of program name block of program to be activated.  
A8: ECB address.
- If request comes from D:ASTB (i.e. a system request for activation):  
A3: system parameter  
A4: return address  
A5: 0  
A6: 0  
A7: address of program name block of program to be activated.  
A8: ECB address.  
CF D:ASTB

#### Work Areas and Tables

PCT of activated program.

Save area of activated program.

Dynamic allocation area, if the activated program is active when this request is given.

#### Input/Output Files

None.

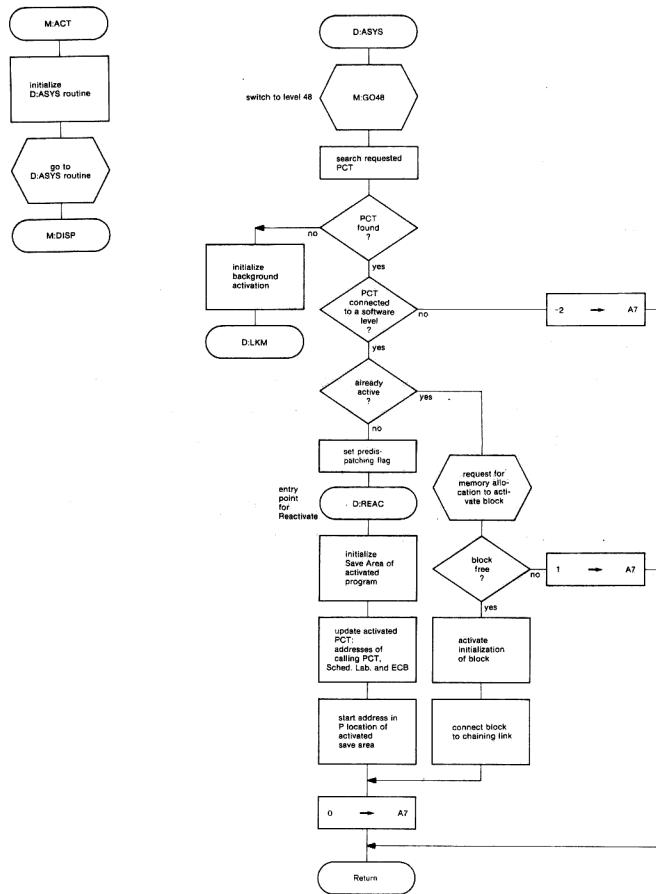
#### Functional Description

This request is handled at level 48.

First the PCT entry of the activated program is searched for in the PCT Pool. If no PCT is found, a specific module is activated to seek the required program on all the discs of this configuration (This module, D:IBCK, is part of the D:USV2 module and runs at level 49). If the PCT of the activated program is found, the activate module initializes the corresponding save area if the program is not yet active and sets its status to active. If the program is already active, a request block is built and connected to the activated PCT

(stacked Activate). At the end of the process a status is set in register A7:

A7 = 0: activation accomplished.  
 ==2: called program has not been connected to a level  
 ==3: unknown program (set by D:ABCK)  
 ==4: dynamic area overflow  
 ==5: overflow of PCT Pool (set by D:ABCK)  
 ==6: overflow of Save Area (set by D:ABCK)  
 ==7: disc I/O error (set by D:ABCK)



M:SWTC (SWITCH INSIDE A SOFTWARE LEVEL - LBN15)

Calling Sequence

A5: PCT address of calling program

A6: Scheduled Label

A7: Level to be switched. If this is zero, the level to be switched is equal to the level of the calling program + 1.

Entry Point: M:SWTC

Work Areas and Tables

SLT (Software Level Table).

PCT (Program Control Table).

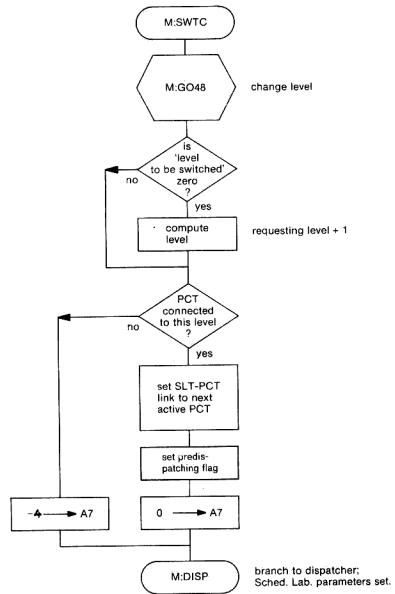
Input/Output Files

None.

Functional Description

There may be several PCT entries connected to one level. A link in the Software Level Table points to one of them. When this request is given, the link will be changed to the next active PCT entry in the chain which is not in wait state.

This request may be given by user as well as system programs. From the system, this request is given by the Wait, Activate, Exit and Set an Event modules.



D:ATDET (ATTACH/DETACH A DEVICE TO/FROM A PROGRAM - LEM 34 + 15)

Calling Sequence

A5: PCT address of calling program  
A6: Scheduled Label, if any  
A7: Wait Flag (- 0: no wait; / 0: wait)  
A8: Address of ECB containing the related file code.

Entry Point: N:ATDW

Work Areas and Tables

PCT (File Code Table)  
DWT (Device Work Table)  
LPT (Logical File Table)  
PCT (Program Control Table)  
An 8-word block in Dynamic Allocation Area.

Input/Output Files

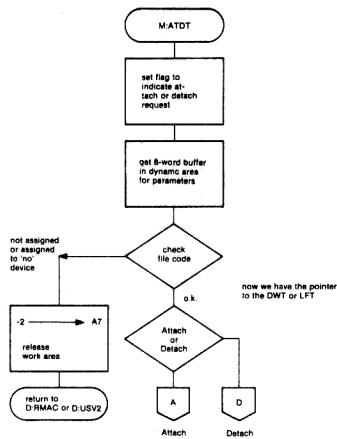
None.

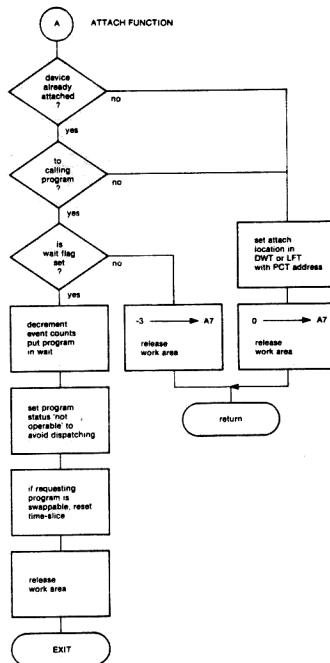
Functional Description

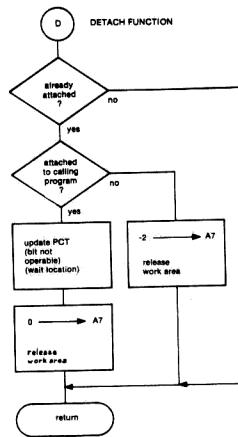
This module provides for attaching or detaching a device to or from a program.

If the device is already attached to another program, the requesting program may, depending on the value of the Wait Flag in A7, be put into wait state (with reinitialization) until the device is detached. Corresponding to the action to be taken, word 34 in the DWT (PCT address of program) is filled or set to /0000.

Note: The ASR is considered as 3 devices, so if the whole ASR is to be attached or detached, 3 requests must be given for the file codes corresponding to the ASR typewriter, ASR tape punch and ASR tape reader.







D:GTIM (GET TIME - LKM17)

Calling Sequence

A7: contains a binary flag.

If this flag is zero, the date and time will be given in ASCII in the 6-word user block.

If it is not zero, they will be given in binary.

A8: address of a 6-word block, containing date and time.

Entry Point: D:GTIM

Work Areas and Tables

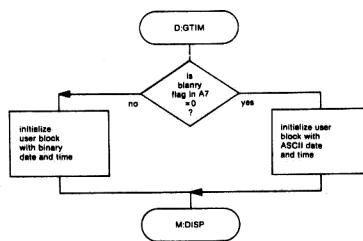
The timer block is read.

Input/Output Files

None.

Functional Description

Depending on the value in the A7 register, a 6-word user block will be filled with a binary or ASCII value specifying the date and time (DD,MM,YY,HH,MM,SS).



M:EEEV (SET AN EVENT - LACK18)

Calling Sequence

A5: PCT address of calling program  
A6: Scheduled Label  
A8: Event Control Block address.

Entry Point: M:EEEV

M:EEEV (pointed to through CNTSET and used by the read-only system programs)

Work Areas and Tables

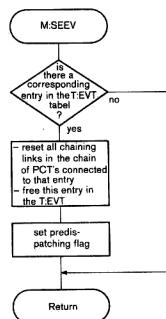
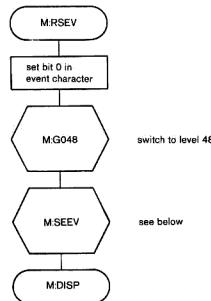
T:SLT (Software Level Table).  
PCT (Program Control Table).

Input/Output Files

None.

Functional Description

When an event has occurred, the system may give this request so that the programs waiting for this event can be restarted. The M:EEEV module looks for the EUB address in the pool of T:EVT tables and updates all PCTs chained to this entry, i.e. waiting for this event, allowing corresponding programs to be restarted by the dispatcher.



D:CNLV (CONNECT A LEVEL TO A PROGRAM - LWM20)

Calling Sequence

A5: PCT address of calling program

A6: Scheduled Label

A7: Level to which the program must be connected.

A8: Address of name block of program which must be connected.

Entry Point: D:CNLV

Work Areas and Tables

T:SLT (Software Level Table)

PCT (Program Control Table)

Save area of program which must be connected.

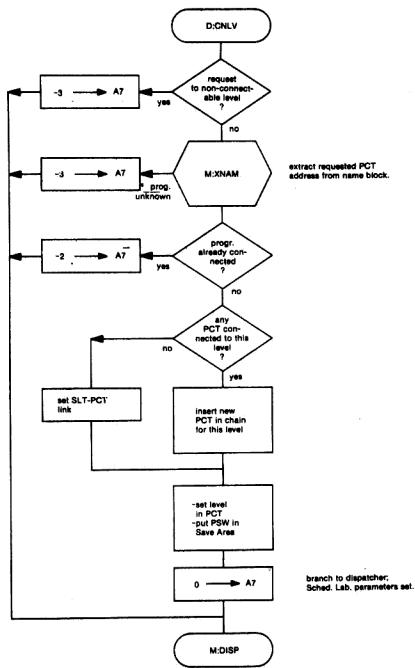
Input/Output Files

None.

Functional Description

By means of this request a link is established between the Software Level Table and the PCT entry of the requested program.

This is simple in case the level is free, i.e. no PCT entry is yet connected to it. If the level is not free, the new PCT entry is inserted into the PCT chain for this level.



D:DNLV (DISCONNECT A PROGRAM FROM A LEVEL - LKM21)

Calling Sequence

A5: PCT address of calling program  
A6: Scheduled Label  
A7: Level which must be disconnected  
A8: Address of name block of program to be disconnected.

Entry Point: D:DNLV

Work Areas and Tables

T:SLT (Software Level Table).  
PCT (Program Control Table).

Input/Output Files

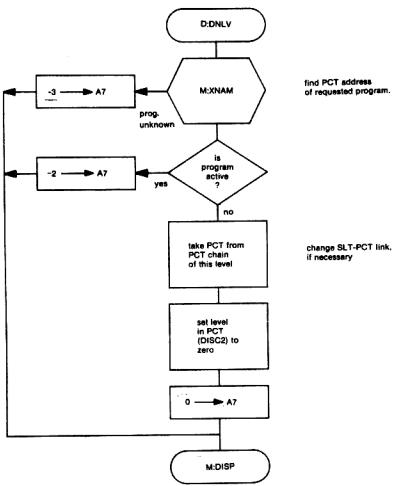
None.

Functional Description

With this request the PCT entry of the program which must be disconnected is taken out of the SLT-PCT chain, provided the corresponding program is in inactive state.

If there is only one PCT entry in the chain for this level, the SLT-PCT link is reset to zero.

If there are more entries in the chain, the requested PCT entry is removed. In both cases the level specified in the PCT entry (DISK2), is reset to zero.



### D:WGT (WAIT FOR A GIVEN TIME - LHM22)

#### Calling Sequence

A8: address of an Event Control Block initialized as follows:

ZERO			
0	TIME NUMBER	W G T	IRRELEVANT
NC			
0 1	3 4 5	6	18

where NC is the number of timer cycles before the program is restarted.

Entry Point: D:WGT

#### Work Areas and Tables

None.

#### Input/Output Files

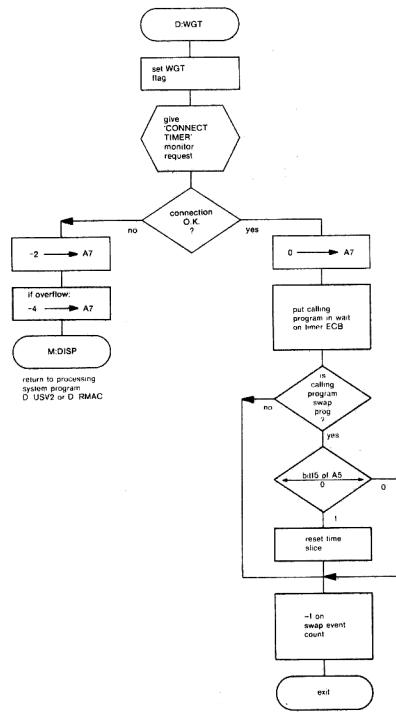
None.

#### Functional Description

This module processes a special time connection, i.e. to a Wait for a Given time block.

If the connection is accomplished, D:WGT puts the program in wait state on the EOB of the timer. If not, the error code -4 is set in the A7 register and control is returned to the dispatcher.

The user program is restarted by the MIDCK4 module after a number of cycles of a specified timer, as defined in a block pointed to by the A8 register. At the same time, the program is disconnected from the timer.



### ASCPROC (ASSIGN A FILE CODE - LEM23)

#### Calling Sequence

A5: PCT address of calling program  
A6: Scheduled label, if any  
A8: Pointer to user parameter block  
A11: PCT address of D:USW1 or D:EMAC  
A12: Return address (to D:USW1 or D:EMAC)  
A13: CVT address

#### Work Areas and Tables

PCT (File Code Table)  
DWT (Drive Work Table)  
LFT (Logical File Table)  
DCT (Disc Control Table)

A 6-word block in the Dynamic Allocation Area for use as work area and a 215-word block if disc I/O is necessary.

#### Input/Output Files

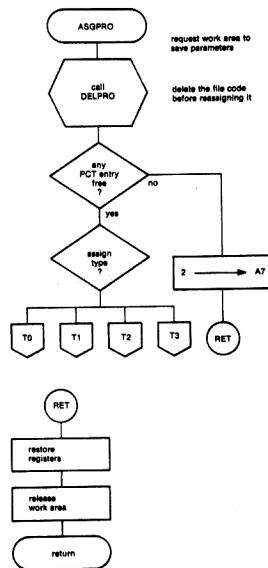
Disc is used to read in the directory or GRANTB.

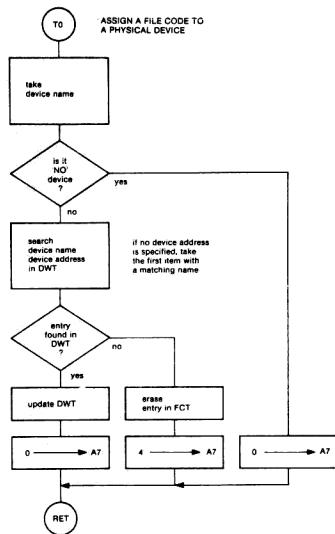
#### Functional Description

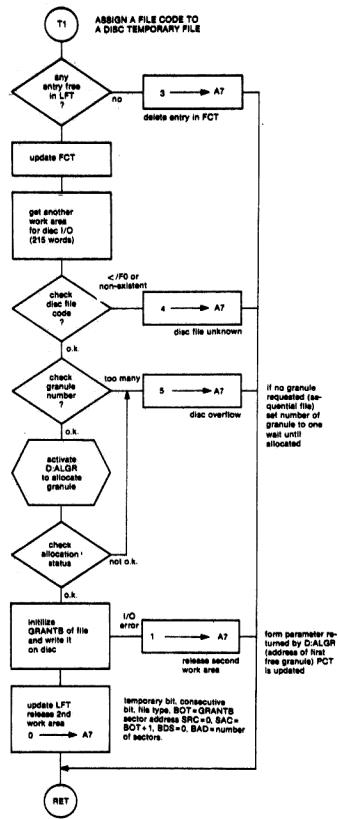
The action taken depends on the type of file code assignment

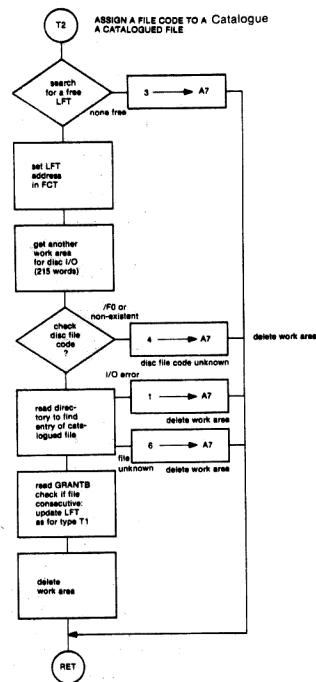
- for physical devices, the Device Work Table is updated
- for disc temporary files, the File Code Table is updated, any necessary granules are allocated, the granule table GRANT is initialised and the Logical File Table is updated.
- for catalogued files the LFT address is set in the File Code Table, the directory is scanned and, for consecutive files, the Logical File Table is updated.
- for equalising file codes, the File Code Table is updated. If it is a disc logical file, ASCNT in LFT is incremented as well.

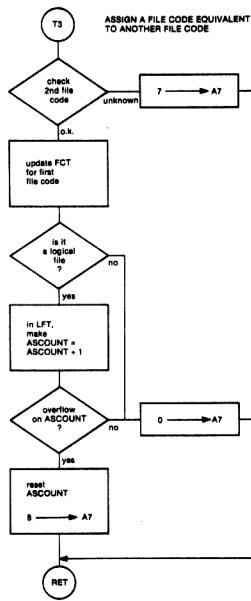
On return, registers A5, A6, A11 and A13 are not destroyed and A7 will contain the status of the operation.











DELPFO (DELETE A FILE CODE - LKN24)

Calling Sequence

A5: PCT address of calling program  
A6: Scheduled Label, if any  
A8: User parameter block address  
A11: PCT address of D:USVT or D:RMAC  
A12: Return address (to D:USVT or D:RMAC)  
A13: CVT address

Work Areas and Tables

POT (File Code Table)  
LPT (Logical File Table)  
DOT (Disc Control Table)

A 9-word block in the Dynamic Allocation Area for use as work area and a 215-word block if disc I/O is necessary.

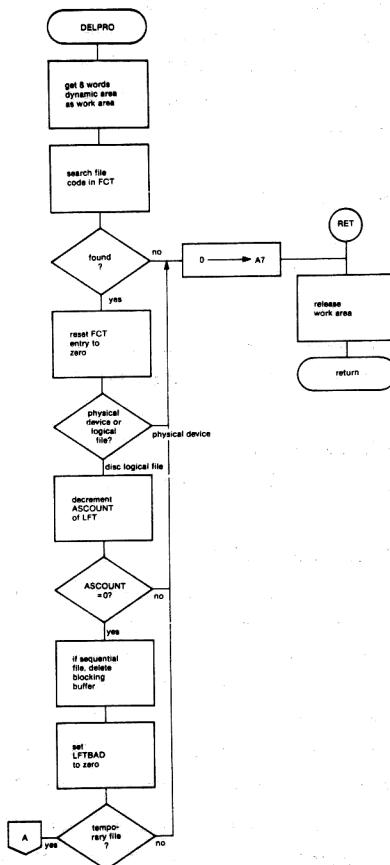
Input/Output Files

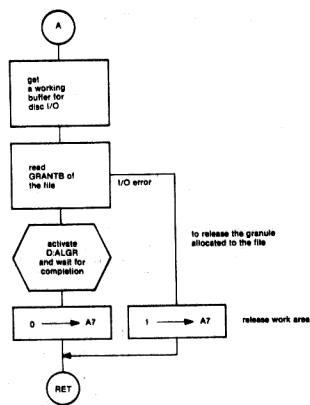
Disc is used to read in GRANTB.

Functional Description

This request deletes a previously assigned file code. For this purpose the File Code Table is scanned and if the file code is found, it is removed from the table. If it is a disc logical file code, ASCNT in word LPTMD2 of the Logical File Description table is decremented and word LPTBAD is set to zero. For sequential files, the blocking buffer is released. Granules allocated to a temporary file are made available again.

On return, registers A5, A6, A11 and A13 are not destroyed and A7 will contain the status of the operation.





## KEYPRO (READ AN UNSOLICITED ANI-IN - IWM25)

### Calling Sequence

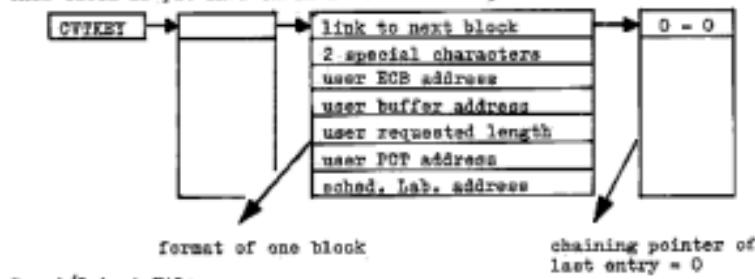
- A5: PCT address of calling program
- A6: Scheduled Label, if any
- A7: User A7 (not used)
- A8: User parameter block address
- A4: A8 + 2
- A11: PCT address of current program (D:USV1) where ECBACT contains the address of the event word on which the user is waiting.
- A12: Return address (to D:USV1 or D:MMAC)
- A13: CVT address.

### Work Areas and Tables

PCT (Program Control Table).

One block in the dynamic area to record the request.

This block is put in a chain used later on by D:000M:



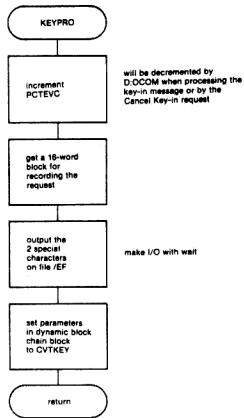
### Input/Output Files

File Code /EP (normally the system typewriter) is used to output the 2 special characters specified by the request to inform the operator that he can enter his message.

### Functional Description

When this request is given, the event count is incremented and a 16-word block is requested in the dynamic allocation area. The special characters defined by the user in a 5-word parameter

block are output on file code /EF. The user parameter block is chained to word CVTKEY of the Communication Vector Table.



CANCEL (CANCEL AN UNSOLICITED KEY-IN REQUEST - LKH626)

Calling Sequence

A5: PCT address of calling program  
A6: Scheduled Label, if any  
A7: User A7 (not used)  
A8: User parameter block address  
A4: A8 + 2  
A11: PCT address of current program (D:USV1) where ECBACT contains the address of the event word on which the user is waiting.  
A12: Return address (to D:USV1 or D:RMAC)  
A13: CVT address.

Work Areas and Tables

PCT (Program Control Table).

One block in the dynamic area to record the request.  
This block is put in a chain used later on by D:OCOM:

See under KEYPRO

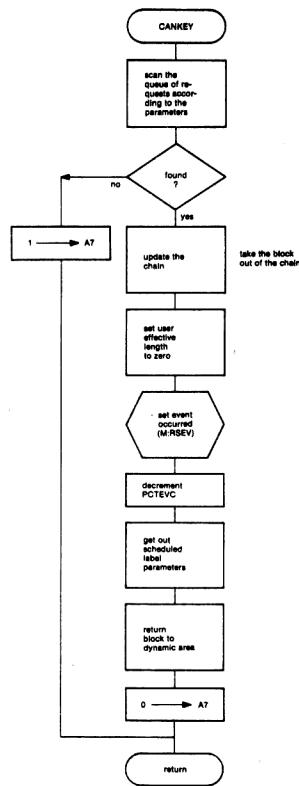
Input/Output Files

None.

Functional Description

A previously requested unsolicited key-in must be cancelled if it is not used or not required. This involves finding the block with the request's special characters, the resetting of certain parameters, especially event counts, and releasing the block.

Register A7 contains the status of the request after completion: if 0, the operation was successful, if 1, it was not.



**D:SWP (SWAPPING MODULE)**

Calling Sequence

This module is activated by the dispatcher every time a swappable program must be swapped out of the Swap area in memory and written on disc in the D:CI file. It normally runs at level 49.

A3: PCT address of program which must be swapped out.

Work Areas and Tables

PCT (Program Control Table)

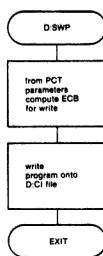
A 6-word block to format an EOB for the I/O operation.

Input/Output Files

This module writes on the D:CI file.

Functional Description

See page 2-17 for a description of the swapping procedure.



## DISC FILE MANAGEMENT (M:DFM)

### Calling Sequence

Upon entry, M:DFM requires the address of the Logical File Description Table (T:LPT) of the file on which the operation is to be performed. It is given in the A4 register. A3 contains the entry point number for D:EMAC, because Disc File Management is part of the D:EMAC program.

### Work Areas and Tables

A buffer of 208 words is required in the dynamic allocation area to block and deblock records for sequential files.

Two tables are used:

- Logical File Description Table (T:LPT)
- Disc Control Table (T:DCT)

### Input/Output Files

Disc File Management uses the disc file codes /FO to /FF to perform physical I/O operations.

### Memory Layout

M:DFM must be link-edited with the supervisor and is therefore always loaded into the monitor partition in memory. It cannot be declared as a read only program.

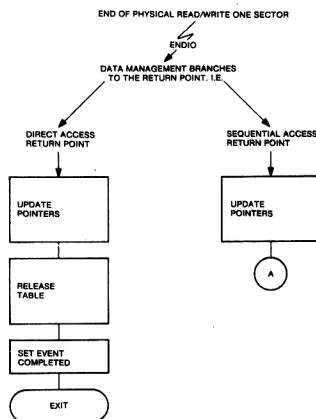
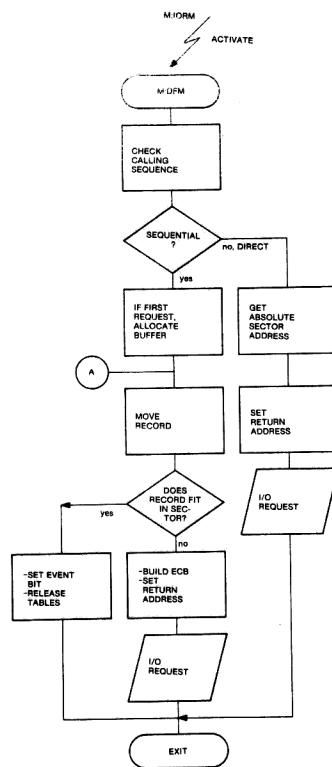
### Functional Description

The M:DFM module performs all I/O operations for user logical files. It is activated by the M:IFORM module.

For sequential files, it blocks and deblocks records.

For random files, it calculates the absolute sector address from the relative sector number and vice versa.

M:DFM also allocates any buffers or granules which are required.



## OPERATOR COMMUNICATION (D:CTPN / D:OCOM)

### Calling Sequence

The module D:OCOM handles the conversation with the operator via the typewriter. It prints error messages and reads commands typed in by the operator. Therefore, different entry points are available:

- to process a command typed by the operator, where the following registers must contain parameters before the module is activated:  
A3 = 0 (entry point number)  
A4 = input buffer address.
- to print error messages, where these registers must contain the following parameters:  
A3 = 2, 4 or 6, depending on the type of message.  
A4 = address of a parameter block which is actually a work area in the dynamic allocation area, where the message is formatted, or:  
DWT address of the device on which the error occurred.

### Work Areas and Tables

To process a command typed in by the operator:

- the buffer used to read the operator command
- in the word CVTOCM in the Communication Vector Table (CVT) the sign bit must be reset to zero after processing, to allow the module to accept a new operator command.

To print the message PW\_DWD4\_Status\_RY:

the work area is used as follows:

- word 0: retry flag: 0, if RY must also be printed out.
- 1: status to be printed, in binary.
- 2: address of the Device Work Table (DWT) of the device.
- 3 - 6: ECB used to print the message.
- 7 - n: are used to format the message.

To print the message DEEE the work area is used as follows:

- word 0: status to be printed
- 1: sector number
- 2: address of the Device Work Table (DWT)
- 3 - 8: ECB used to print the message
- 9 - n: are used to format the message.

To print the message TC OFF, the work area is used as follows:

- word 0 - 5: ECB used to print the message
- 6 - n: are used to format the message.

#### Input/Output

The input and output files for this module are mainly combined in the operator's typewriter, with file code /EP.

For the dump command, however, D:OCOM also must use the disc unit file code (/PO to /PP) and the print file (/PO) which can be assigned to any output device.

#### Memory Layout

The operator communication package consists of several modules. The D:CTPN module (used to read control panel key-ins) is the only one which is memory resident. The others are always read only.

If desired, the user may also declare D:CTPN a read only program, but the priority level to which it is connected must be selected carefully, because otherwise the system response time to the control panel interrupt may become too high.

#### Function Description

The following components are included in the operator communication package:

INTCP is the control panel interrupt routine.

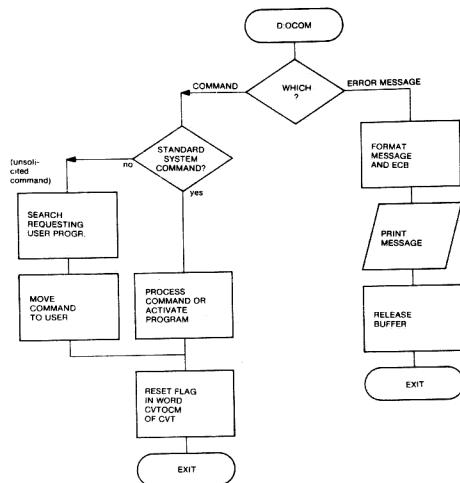
D:CTPN is the control panel program, i.e. it handles the input of operator messages (memory resident).

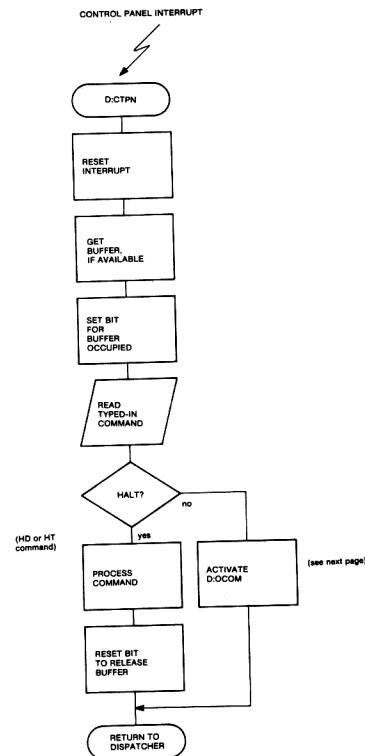
D:OCOM processes the input of operator messages or the output of system messages (normally disc resident).

The D:OCOM program is used mainly to print error messages to the operator and to process commands typed in by the operator, such as CC, RY, RD, DM, DD, etc.

After the command has been read by the D:CTPN module, control is passed on to D:OCOM to analyze and process the command.

All commands, excluding Dump and Connect, are processed by D:OCOM. The Dump and Connect commands require the use of external devices and are therefore processed by the D:DUMP module, which must run at a lower priority level than D:CTPN and D:OCOM.





## M:DMA (DYNAMIC MEMORY ALLOCATION HANDLER)

### Calling Sequence

A1: Length of requested buffer, in characters (bits 1 to 15).  
    If bit 0 = 1: user request (from M:GSUP)  
    If bit 0 = 0: system request.

A3: Base address of area in which buffer is allocated.

INH

CF A1S, M:DMA

Entry Point: M:DMA

### Work Areas and Tables

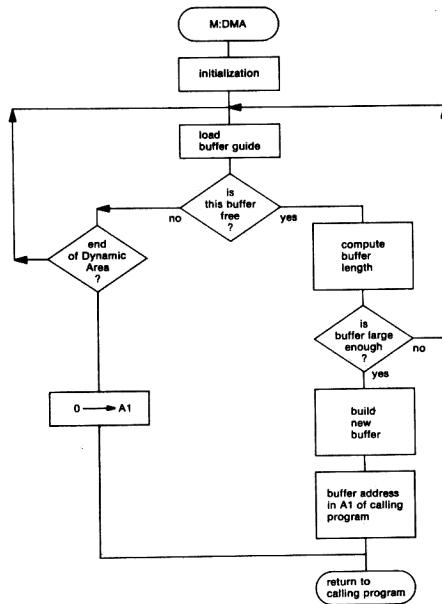
Dynamic Allocation Area (See Chapter 3).

### Input/Output Files

None.

### Functional Description

When a request is made to this module, it searches the dynamic allocation area for a free buffer. If no buffer can be found which is large enough for the request, the search is restarted. When a suitable buffer is found, it is initialized and the status flag in the buffer guide is reset to 0. On return the address of the requested block can be found in register A1. In case a request cannot be satisfied, M:DMA returns A1 with the value 0, to indicate dynamic area overflow.



## M:DML (DYNAMIC MEMORY DEALLOCATION HANDLER)

### Calling Sequence

A1: address of the buffer which must be deallocated.

A2: base address of area containing the buffer

DHE

CF A15, M:DML

Entry Point: M:DML

### Work Areas and Tables

Dynamic Allocation Area (see chapter 3).

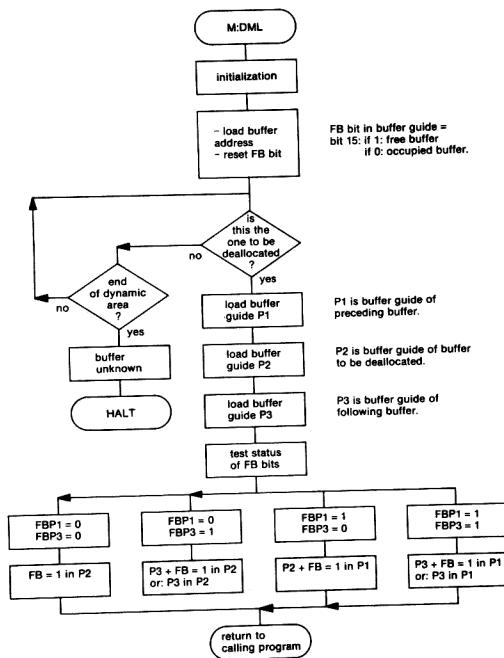
### Input/Output Files

None.

### Functional Description

When a request is given to the M:DML module, it searches the buffer which must be deallocated in the dynamic allocation area (address in A1). If the buffer is found, M:DML updates the necessary buffer guides and the status flag in the deallocated buffer guide, thus returning the buffer for allocation.

If the buffer is unknown, the system halts, i.e. a branch is made to D:HLT, with error code 9 in A1.



## M:DCK (TIMER HANDLER)

### Calling Sequence

This program is started by I:RTC activation.

On entry the parameters are contained in V:FLAG, which is set by the I:RTC module.

### Work Areas and Tables

V:FLAG: flag vector indicating the timer chain to be scanned.

V:RESET: a value vector to reset the timers.

H:TIME: timer block start address.

H:POINT: chain pointer start address.

All programs connected to timer blocks (built by the M:CNTM module) are managed by N:DCK.

### Input/Output Files

None.

### Functional Description

This module is started on activation by the I:RTC clock driver and runs at level 49.

It is composed of three modules:

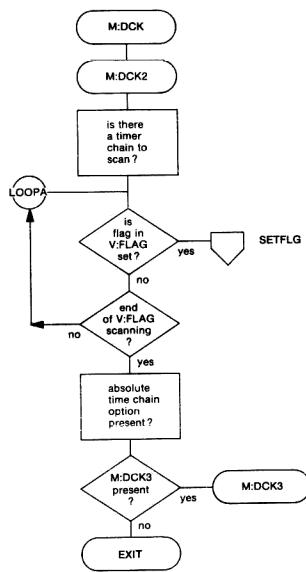
- M:DCK2 manages the chain of programs connected to standard timers.
- M:DCK3 manages the chain of programs connected to the absolute time (NC = HH\_MM\_SS).
- M:DCK4 manages the programs which are put in 'Wait for a Given Time'.

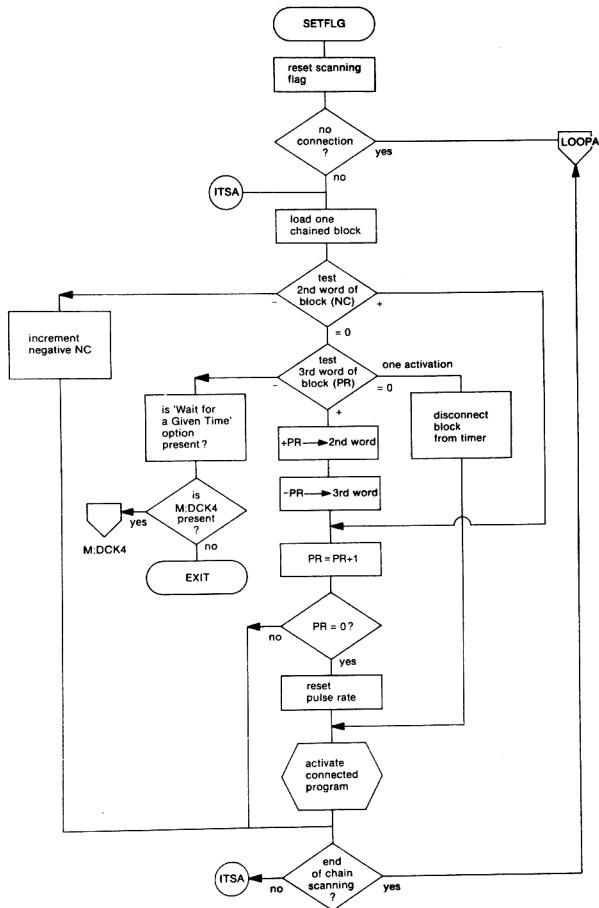
M:DCK2 first checks V:FLAG (initialized by I:RTC) to find out if a chain of programs connected to a timer must be analysed.

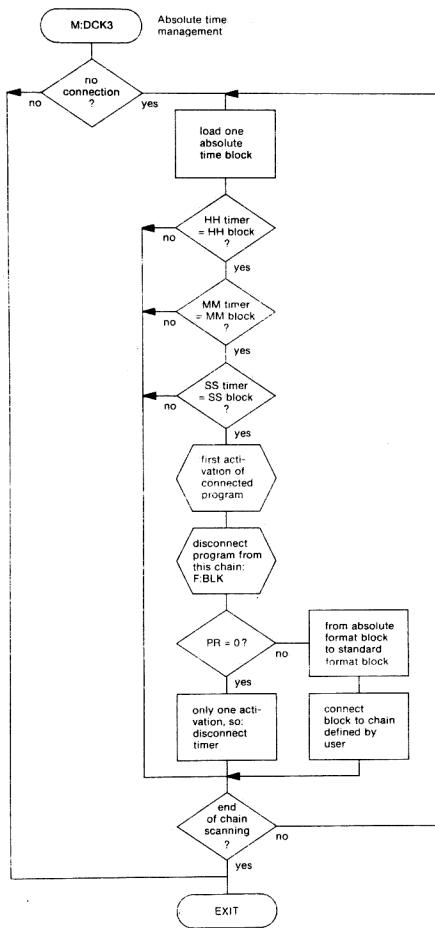
At the end of M:DCK2, or in case no programs are connected to the scanned timer chains, the absolute time option is asked for. If it is present, M:DCK3 is started, otherwise M:DCK2 exits.

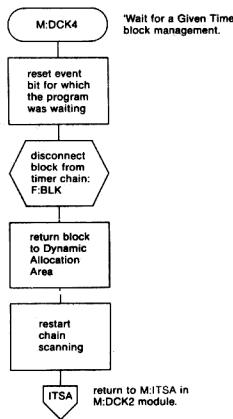
If there are programs connected to a chain flagged by I:RTC in V:FLAG, the timer blocks are updated and the programs are activated,

if necessary. If a Wait for a Given Time block is detected, that option is asked for and if it is present, McDCK4 is started. Otherwise, the block is ignored. At the end, control is returned to the dispatcher through an Exit request.









## P:BLK (RELEASE A BLOCK FROM A TIMER CHAIN)

### Calling Sequence

A1: Chain pointer address.

A2: Address of block which must be disconnected from the chain.

Entry Point: P:BLK.

This module is called only by:

- D:DTTM (Disconnect a Timer module)
- M:DCK3 (Absolute time management module)
- M:DCK4 ('Wait for a Given Time' management module).

### Work Areas and Tables

H:POIN: chain pointer

### Input/Output Files

None.

### Functional Description

This module is a service routine by means of which a block can be disconnected from a timer chain.

The block which must be disconnected is locked up and the chain pointer updated.

At the end of the routine A1 will contain one of these values:

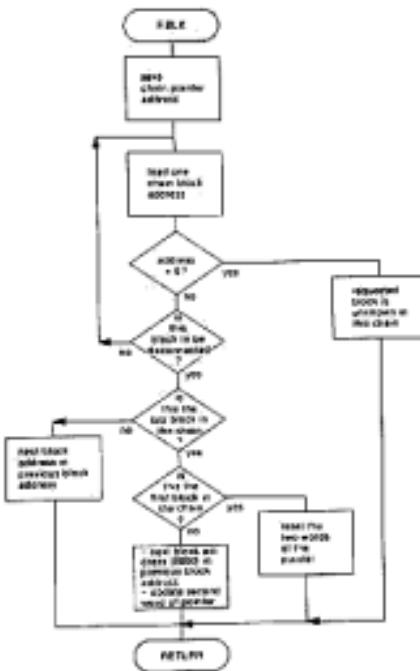
A1 ≠ -1: disconnection performed.

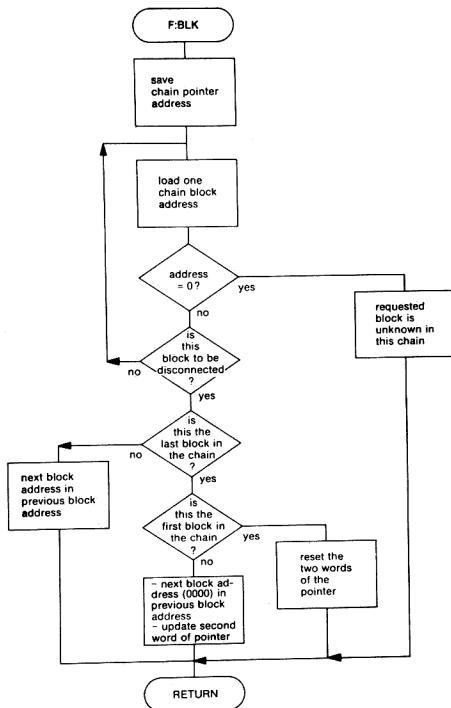
A1 = -1: the block cannot be found in this chain.

A2 remains the same.

A3 and A4 are destroyed.

A6 must contain the return address.





## M:AOO (NON-WIRED INSTRUCTION SIMULATION ROUTINE)

### Calling Sequence

When an interrupt signal is generated as the result of the use of an illegal instruction code, the I:LKM module is started. First, this module checks whether the code specified is an LKM instruction. If not, the M:AOO routine is started and I:LKM transfers the following parameters to M:AOO:

- A1: user operation code
- A2: address of user data
- A3: user data.

### Work Areas and Tables

T:OPC (non-wired operation code table):

MULTIPLY OPERATION CODE	DIVIDE OPERATION CODE
DOUBLE ADD OPERATION CODE	DOUBLE SUBTR. OPERATION CODE
	-

one character per  
operation code

The scanning index is initialized by a parameter L:TOPC:  
L:TOPC EQU 'non-wired op.code number - 1'

T:SRA (table of addresses of non-wired op.code simulation routines):

MPYMOD address
DIVMOD address
ADDMOD address
DSUMOD address

one word for each  
simulation routine  
address

T:SVR (user information safeguard table)

USER P-REGISTER
USER PW
USER A1-REGISTER
USER A2-REGISTER
USER A3-REGISTER
PW SIMULATION ROUTINE
PREGISTER SIMULATION ROUTINE

Each entry (16 words) in this table contains P-register, PW and registers A1 to A14 of the user program, plus the P-register and PW of the simulation routine. The number of entries is defined at system generation time.

The scanning index is initialized by a parameter L:TSVR: L:TSVR EQU (number of entries - 1) x 36.

S:SVR (M:A00 routine register save area)

The length of this area is 8 words:

M:A00 A1-REGISTER
M:A00 A2-REGISTER
M:A00 A3-REGISTER

#### Input/Output Files

None.

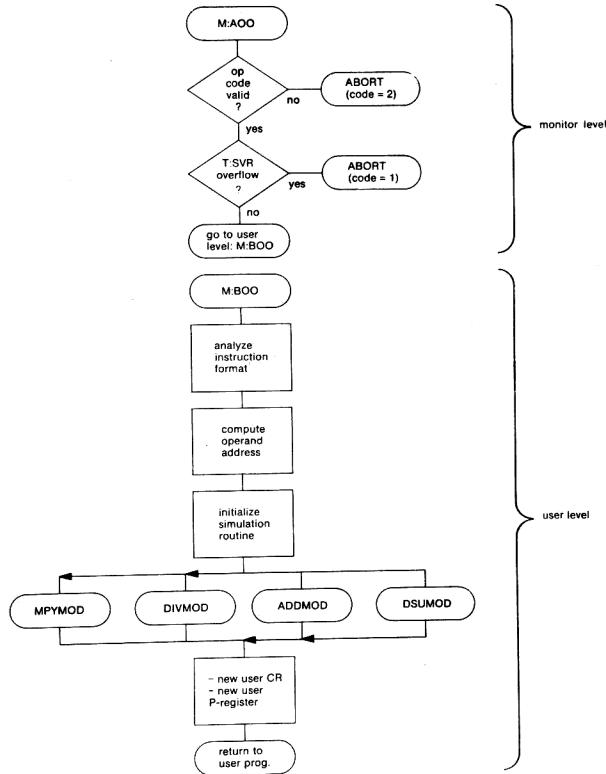
#### Functional Description

First the M:A00 checks the validity of the given operation code. If it is not valid, the ABORT routine is started with code 2. Then it checks the T:SVR table to assign a save area. If overflow occurs on this table, the ABORT routine is started with code 1.

Note: In this case the overflow is not a user error, but a system surrenders (too many non-wired instructions encountered at the same time).

If a T:SVR entry is available, the user program must be restarted.

Then M:ACO gives control to the M:B00 which operates at user level. This routine looks for the user operand and initializes an arithmetical routine to simulate the operation code given up by user. At the end of simulation control is given back to the M:B00 routine which updates the user program's P-register, PSW and condition register before restarting the user program.



I:TRAP (TRAPPING ROUTINE)

Calling Sequence

Upon interrupt caused by use of an illegal instruction, a branch is made to I:TRAP via the trap location: memory address /7E.

Work Areas and Tables

A-15 stack.

Input/Output Files

None.

Functional Description

After saving 8 registers in the A15 stack and a check on overflow, the parameters are prepared by I:TRAP for the MiAGC routine, which will simulate the illegal instruction.

