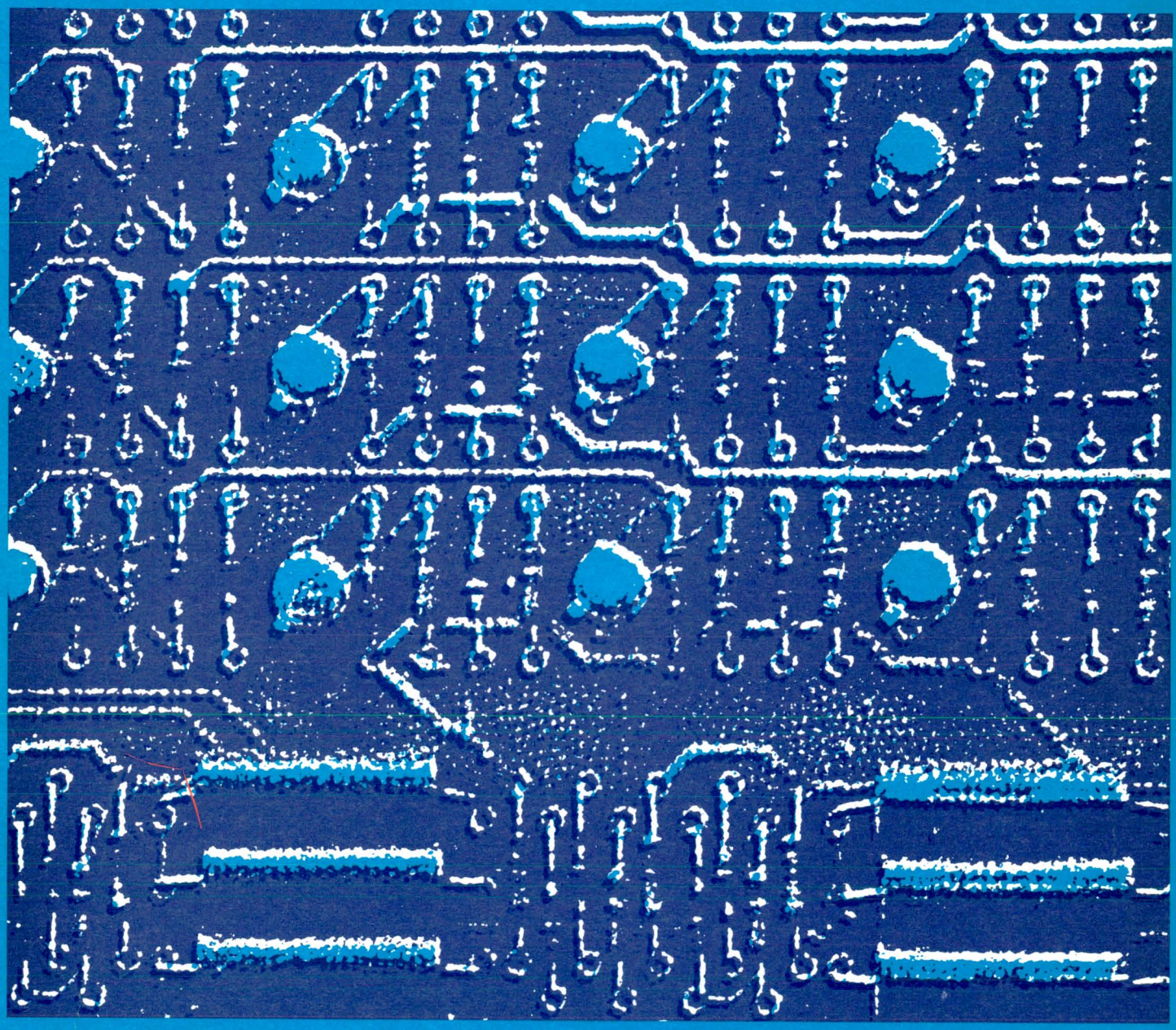
P800M Programmer's Guide 2

Volume IV: Disc Real Time Monitor



PHILIPS Data Systems

PHILIPS

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

A publication of

Philips Data Systems B.V.
Marketing Group Small Computers
Apeldoorn, The Netherlands

Publication number 5122 991 27402

May, 1976

Printed in The Netherlands.

Preface

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 V: Multi-Application Monitor
- Volume VI: Extended Disc File Management

Other books pertaining to the P800M series are:

P852M System Handbook
P856M/P857M System Handbook
P800M Operator's Guide
P800M Interface and Installation Manual
P800M 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:

Manual Writing Small Computers
at the address on the opposite page.

Table of Contents

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 pretested under the DOM.
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 socalled scheduled label feature.
The monitor is of modular structure to allow the user to easily adapt it to his particular application.

1 Principles of Operation

The P852M 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 14 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 DRTM the EDFM package can be used for accessing extended disc files. This is described in Programmer's Guide 2, Volume VI: EDFM

All I/O operations are initialized by monitor requests. Monitor requests can perform various functions. They consist of an LKM 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:

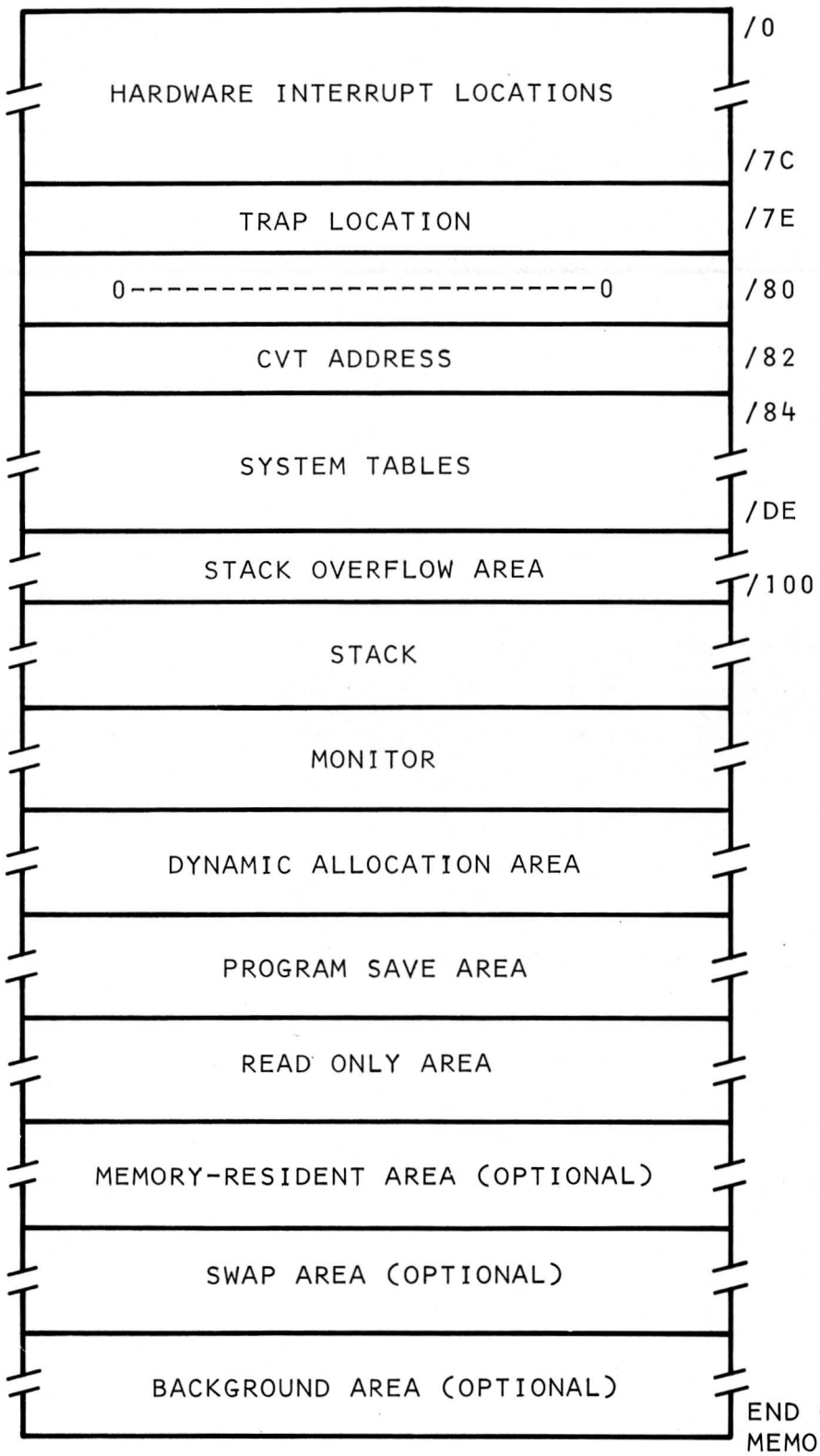
- program activation
- waiting for the occurrence of an event
- program exit
- obtaining and releasing buffer space in the dynamic allocation area
- connecting programs to a priority level and disconnecting them
- switching from one program to another one at the same level to allow time slicing, i.e. 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 unsollicited 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 /7C 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 /980 characters 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 DRTM 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	
50-63: user and system programs:	
50-61: foreground programs	
62 : background programs	
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 /7C. These locations contain the addresses of the interrupt routines which service the internal and external interrupts.
For the interrupts 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 :	LKM/stack overflow	- (interrupt location /02)
/2 :	real time clock	- (interrupt location /04)
/3 :	not used	etc.
/4 :	punched tape reader	
/5 :	tape punch	
/6 :	operator's typewriter	
/7 :	control panel	
/8 - /F:	free	
/10:	X1215 disc	
/11:	disc	
/12:	disc	
/13:	magnetic tape	
/14:	cassette tape	
/15:	card reader	
/16:	free	
/17:	line printer	
/18 - /1F:	free	

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.
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 instuction sequence to use up idle central processor time. See also Chapter 4: Programs.

Dispatcher

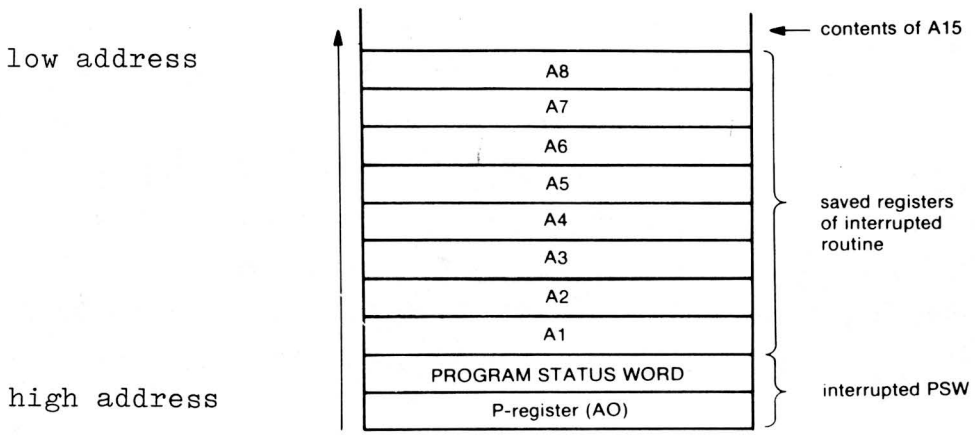
The dispatcher is a monitor module (M:DISP) running on level 48, which divides central processor time by starting programs according to their priority.
The dispatcher can be entered only from an interrupt routine, i.e. from a level below 48, such as the I/O interrupt and monitor request handlers.

System Interrupt Stack

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

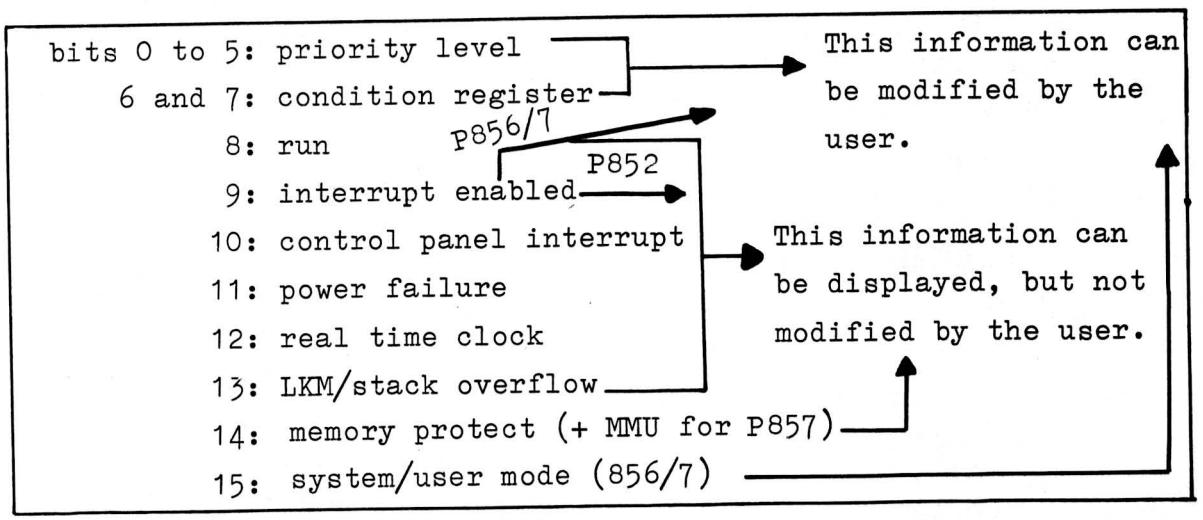
This is done in the system stack, the address of which is held in register A15. The start address of this stack is defined at system generation time and it is built in a downward direction in memory, i.e. towards the lower addresses. The A15 register always points to the first free location in the stack.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

- <u>any number</u> if the interrupt routine runs in inhibit mode and takes care of restoring the registers itself;
- <u>registers A1 to A8</u> 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 occur

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 CF (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 /7C). 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 LD command (see chapter 9).

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 INH 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-ENB-INH). 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 P856/7 and when the simulation rou-
STR     A2,A15          tine for multiple load/store has been se-
 |       | |            lected at sysgen for P852, the sequence is:
STR     A8,A15          MSR 8,A15
     |                   ' '
   coding                coding
     |                   | |
                         MLR 8,A15
LDR     A8,A15          RTN A15    )
LDR     A7,A15
 |       | |
LDR     A1,A15
RTN     A15
```

This is the case of an interrupt routine in inhibit mode with a normal return via the A15 stack. The RTN 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 CVT), he must

take care that before the ENB or ABL instruction is given, the A15 stack contains only P, PSW and registers A1 to A8 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 A8, so any other registers used, must have been restored before making the branch.
- In case of an interrupt routine for internal interrupts (LKM/ 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 (CN 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' (LKM13), 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 /980 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 (LKM4) and Release Buffer (LKM5).
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 program 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   *+2
RB   *-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.

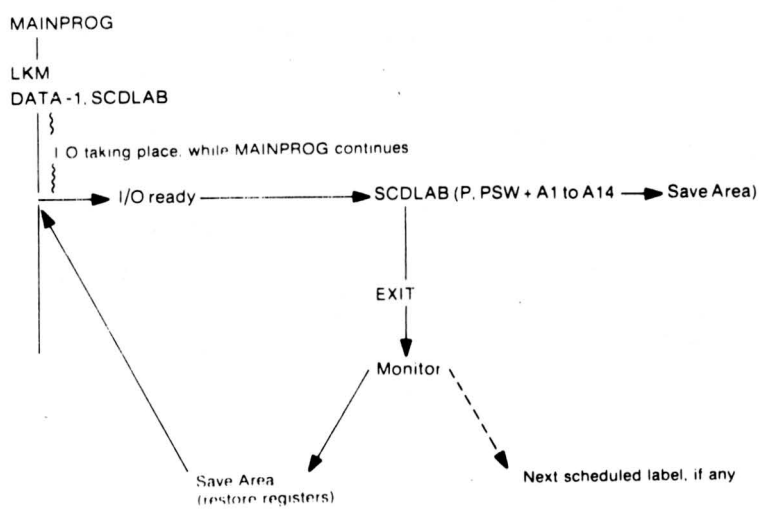
It will be clear from the above subdivision that, although a program may be declared in any class, the main reasons for deciding on program declaration will be response time and calling frequencey.

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:
LDK A7,CODE	LDK A7,CODE
LDKL A8,ECBADR	LDKL A8,ECBADR
LKM	LKM
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 i 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 genera tion 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.

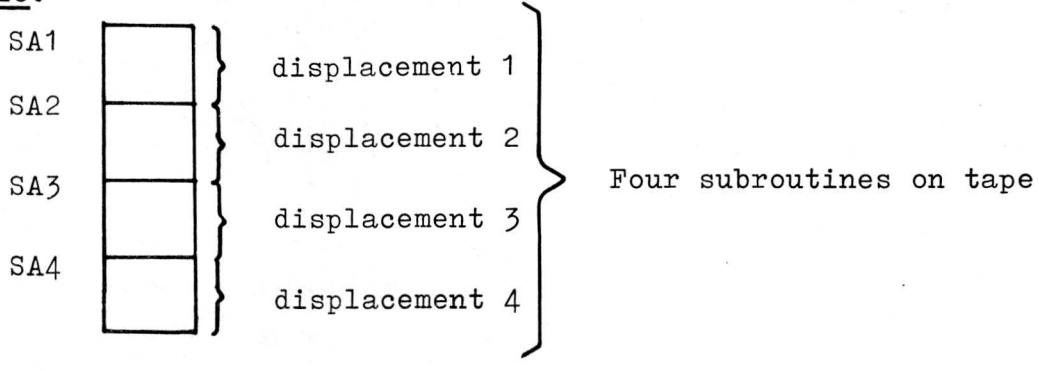
```
        IDENT SLAB
BUF1    DATA '1234567890'       SPECIFYING THE CHARACTERS TO BE
BUF2    DATA 'ABCDEFGHIJ'       OUTPUT
BUF3    DATA 'ZYXWVUTSRQ'
BUF4    DATA '1A2B3C4D5E'
BUF5    DATA 'BUF5'
BUF6    DATA 'BUF6'
DCB1    DATA 5,BUF1,5,0,0,0     DECLARING THE EVENT CONTROL
DCB2    DATA 5,BUF2,40,0,0,0    BLOCKS FOR THE OUTPUT OPERATIONS.
DCB3    DATA 3,BUF3,5,0,0,0     5 IS THE FILE CODE FOR TYPEWRITER,
DCB4    DATA 2,BUF4,12,0,0,0    3 FOR PUNCH, 2 LINE PRINTER.
DCB5    DATA 5,BUF5,6,0,0,0
DCB6    DATA 5,BUF6,6,0,0,0

START   LKD   A7,6              MAIN PROGRAM STARTS AND REQUESTS
        LDKL  A8,DCB1           STANDARD OUTPUT OF BUF1 ON THE
        LKM                     TYPEWRITER. SCHEDULED LABEL SLAB1
        DATA  1,SLAB1           ATTACHED THIS REQUEST.
        LDK   A7,6              MAIN PROGRAM REQUESTS STANDARD
        LDKL  A8,DCB3           OUTPUT OF BUF3 ON THE TAPE PUNCH.
        LKM                     SCHEDULED LABEL SLAB2 ATTACHED
        DATA  1,SLAB2           TO THIS REQUEST.
        LDK   A7,6              MAIN PROGRAM REQUESTS STANDARD
        LDKL  A8,DCB4           OUTPUT OF BUF4 ON THE LINE PRINTER.
        LKM                     SCHEDULED LABEL SLAB3 ATTACHED
        DATA  1,SLAB3           TO THIS REQUEST.
        LKM
        DATA 3                  MAIN PROGRAM EXIT

SLAB1   LDK   A7,6              SLAB1 REQUESTS STANDARD OUTPUT
        LDKL  A8,DCB2           OF BUF2 ON THE TYPEWRITER
        LKM
        DATA 1
        LKM
        DATA 3                  AND EXITS
SLAB2   LDK   A7,6              SLAB2 REQUESTS STANDARD OUTPUT
        LDKL  A8,DCB5           OF BUF5 ON THE TYPEWRITER
        LKM
        DATA 1
        LKM
        DATA 3                  AND EXITS
SLAB3   LDK   A7,6              SLAB3 REQUESTS STANDARD OUTPUT
        LDKL  A8,DCB6           OF BUF6 ON THE TYPEWRITER
        LKM
        DATA 1
        LKM
        DATA 3                  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:



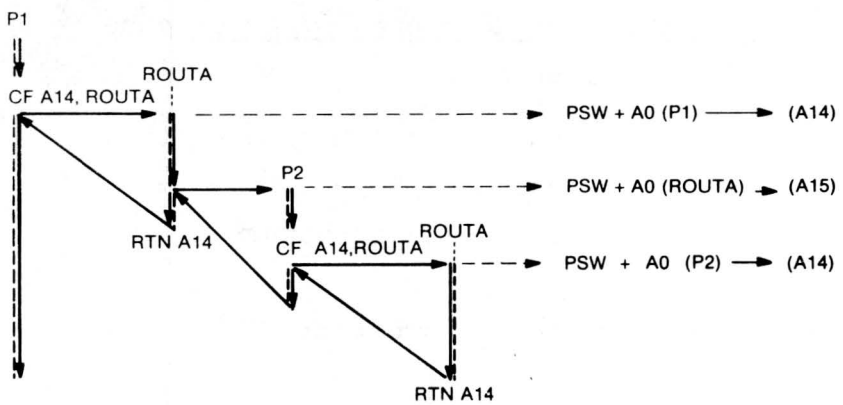
Dummy program:

```
        IDENT   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
  |
EXTRN     SA1
EXTRN     SA4
  |
  CF      A14,SA4
  |
  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 situation:



- 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 → (52) → Switching program:

```
LDK   A7, 55
LKM
DATA  13
```

This calling sequence requests switching on level 55.

```
LKM
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:

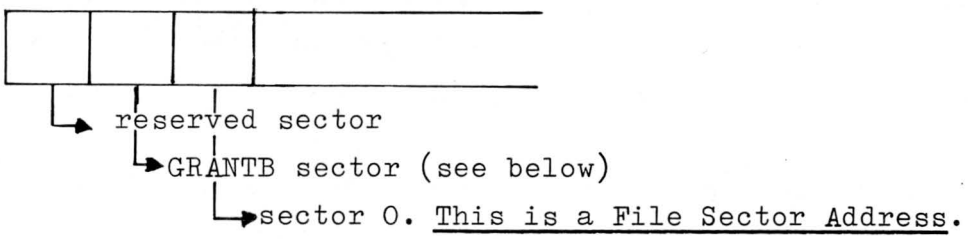
- until an interrupt starts an interrupt routine (level 0 to 47) or until it is required to start level 53 or 54, e.g. because an I/O operation for which one of them was waiting is ready.
- until 20 milliseconds have passed. Then the real time clock gives another interrupt and control is returned to 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 msecs. 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 must 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.

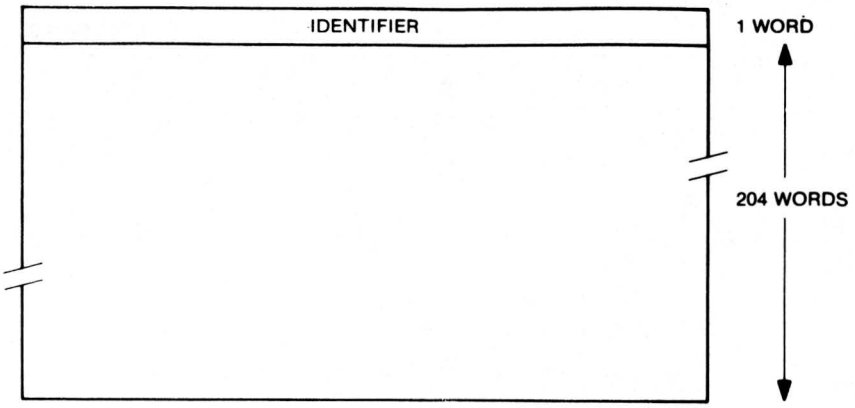
To understand this chapter it is necessary to define some concepts first.
The disc organization is based on sectors, which are sections of a track with a length of 205 words, of which 200 are usable.
On the basis of this sector concept, the following definitions must be kept in mind.

- 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 (DSLA):
  the address of a sector on the disc as calculated by interlacing, which is ordering the sectors in such a way as to make access as efficient as possible (see below).
- File Sector Address (FSA):
  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:

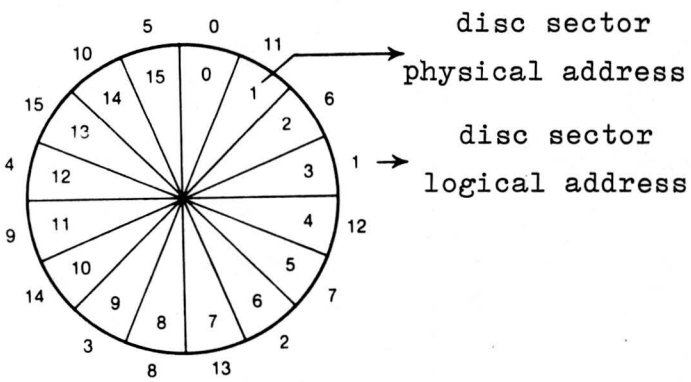


SECTORS

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 for 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 by the DISC PREMARK program. 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 sector 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 interlaced, on a factor -3 basis for discs with 8 or 16 sectors per track.

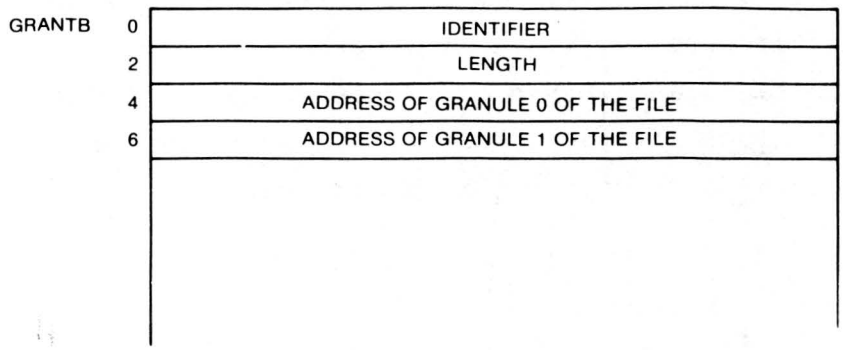


16 sectors

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.

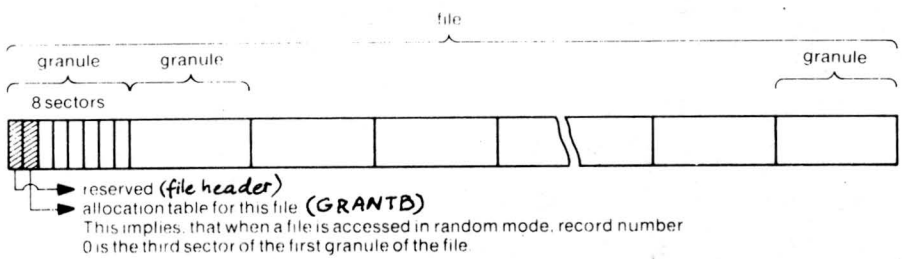
FILES

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 granule GRANTB is initialized when a file code is assigned. If the file is consecutive, all the granules are allocated consecutively for random access, otherwise one granule at a time is allocated for sequential access files. GRANTB is filled with the addresses of these granules, any remaining words being set to zero. A GRANTB thus initialized for random access is not modified. 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. 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.



In this table, location 2 (i+2) is the address of the ith 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. GRANTB is 200 words long, so the file length is limited to 320k words (200x8 sectors).

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 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 GRANTB). 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 DRTM, the requested number of granules is allocated and the file cannot be extended, unless it is completely rewritten.

- 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:

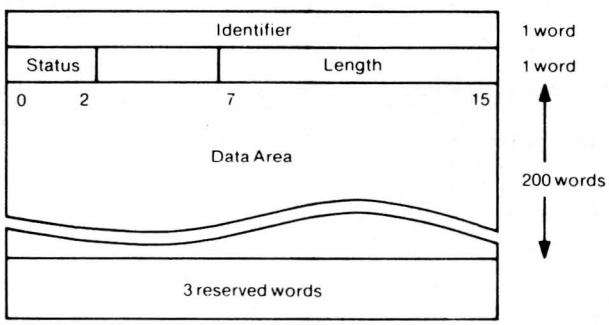
- records may be of any length, up to 16k words. The sector format of such records is described below.
- the only relation between the records in a sequential file is their sequence. The records of such a file must be presented in the order in which they must be written onto the disc, i.e. the

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

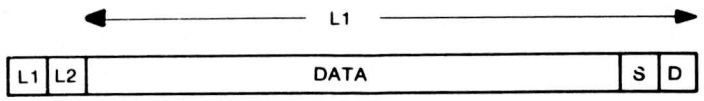
The format of physical record (sector) has already been described (see Physical Organization). This is the record format for random files.
In sequential files a sector may contain a logical record or it may be part of a logical record. Sectors have the following format in these files:



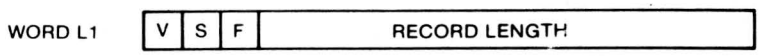
- status consists of 3 bits:
  bit 0: if 1, this sector has been deleted
  bit 1: if 1, an EOS (End-of-Segment) record has been written in this sector. This record is the last one in the sector, for the first record following an EOS always starts at a new sector.
  bit 2: if 1, EOF (End-of-File) record 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
S = 1: this record is an EOS record
F = 1: this record is an EOF record

L2 is the initial record length, in characters. This is the requested length recorded in the ECB 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):

:EOS: 4004 - 0 - S - D
(4004 = L1)

:EOF: 2004 - 0 - S - 4
(2004 = L1, 4 = D)

Blank card: 0004 - 0050 - S - D
(0004 = L1, 0050 = L2) (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 TYPES

Under the DRTM, 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 DOM 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 DRTM 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:

Number of Granules	Maximum Program Size (Words)
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 /980 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 swappable programs in core image format.
The size of this file is declared during the DRMGEN 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 /980 words, each one requires 6 sectors on the D:CI file, i.e. 8 granules for the 10 system programs.

DISC STRUCTURE

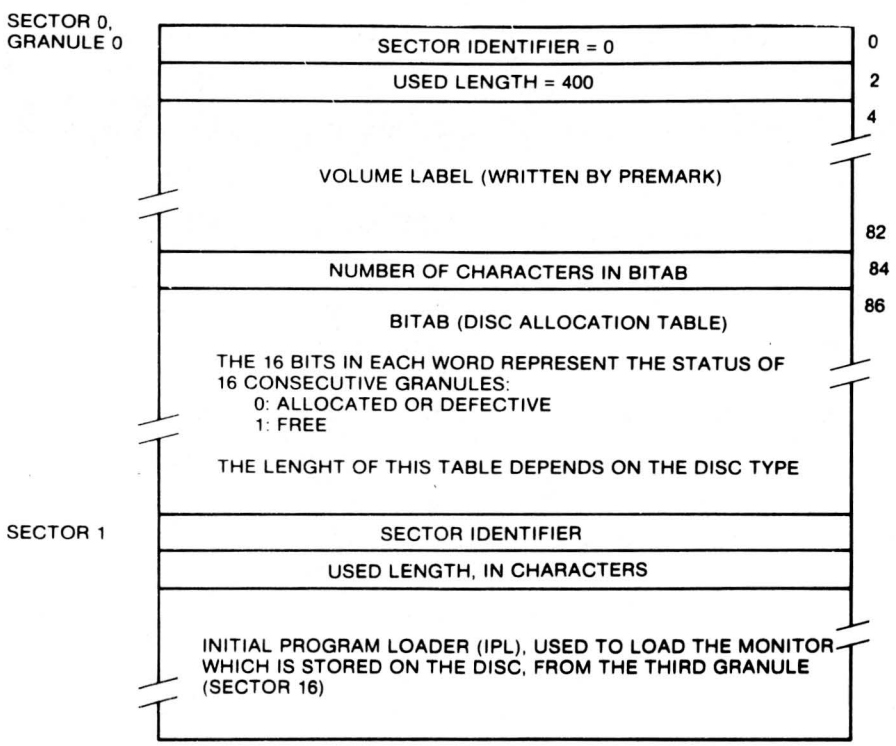
Catalogue and Library Structure

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

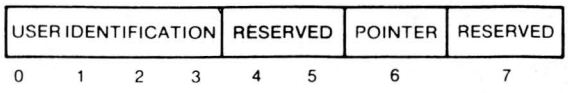
The first granule on a disc contains a catalogue of the users of this disc. Its layout is as follows:

- 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:



- words 0 to 3 contains the user identification

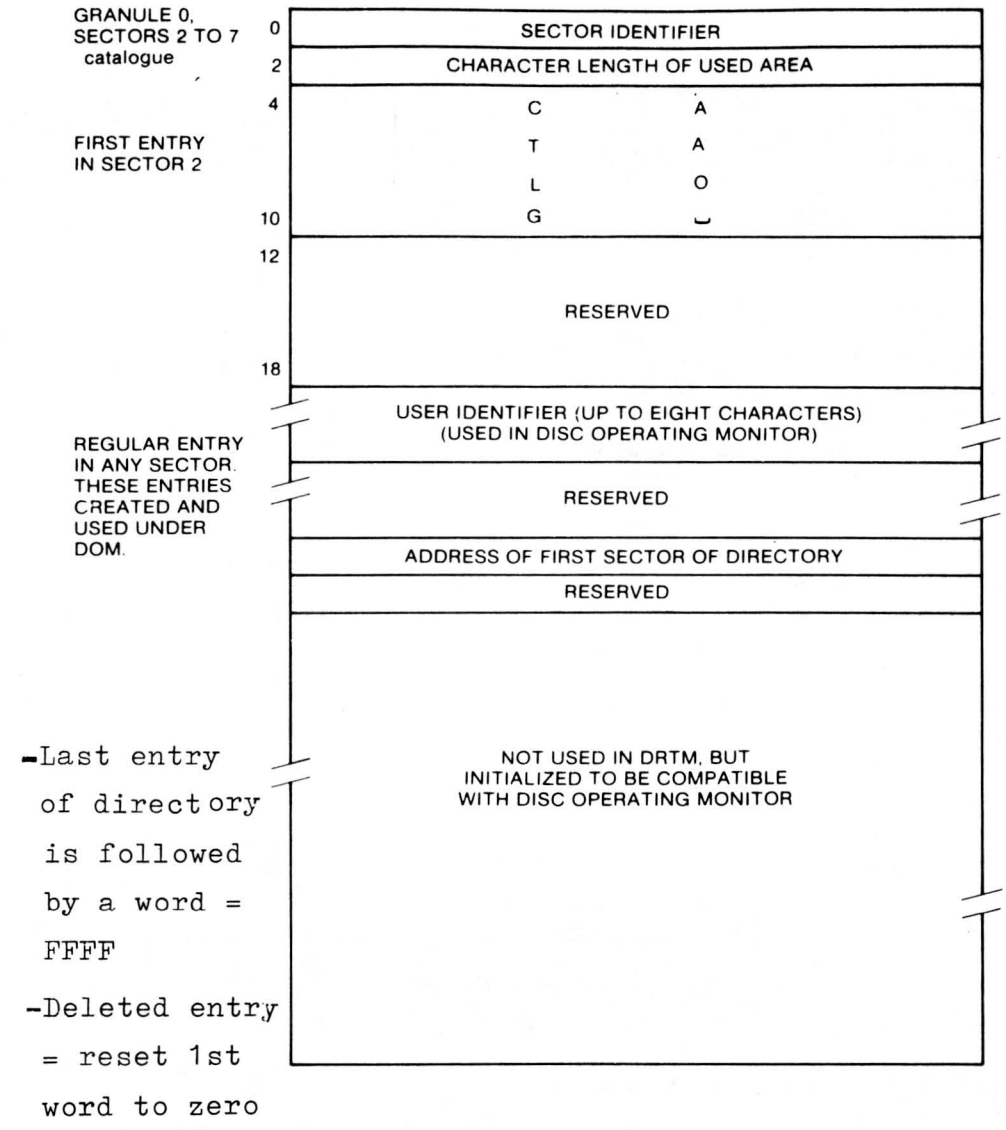
- words 4 and 5 are not used
- words 6 contains a pointer to the user directory; it is the disc sector address of the granule containing the user library directory.
- word 7 is reserved.

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.

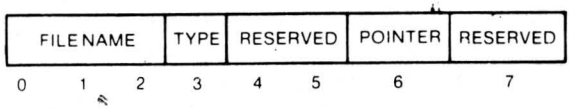
The last entry in the catalogue is followed by a word containing /FFFF.



Format of user catalogue (sectors 2 to 7) -

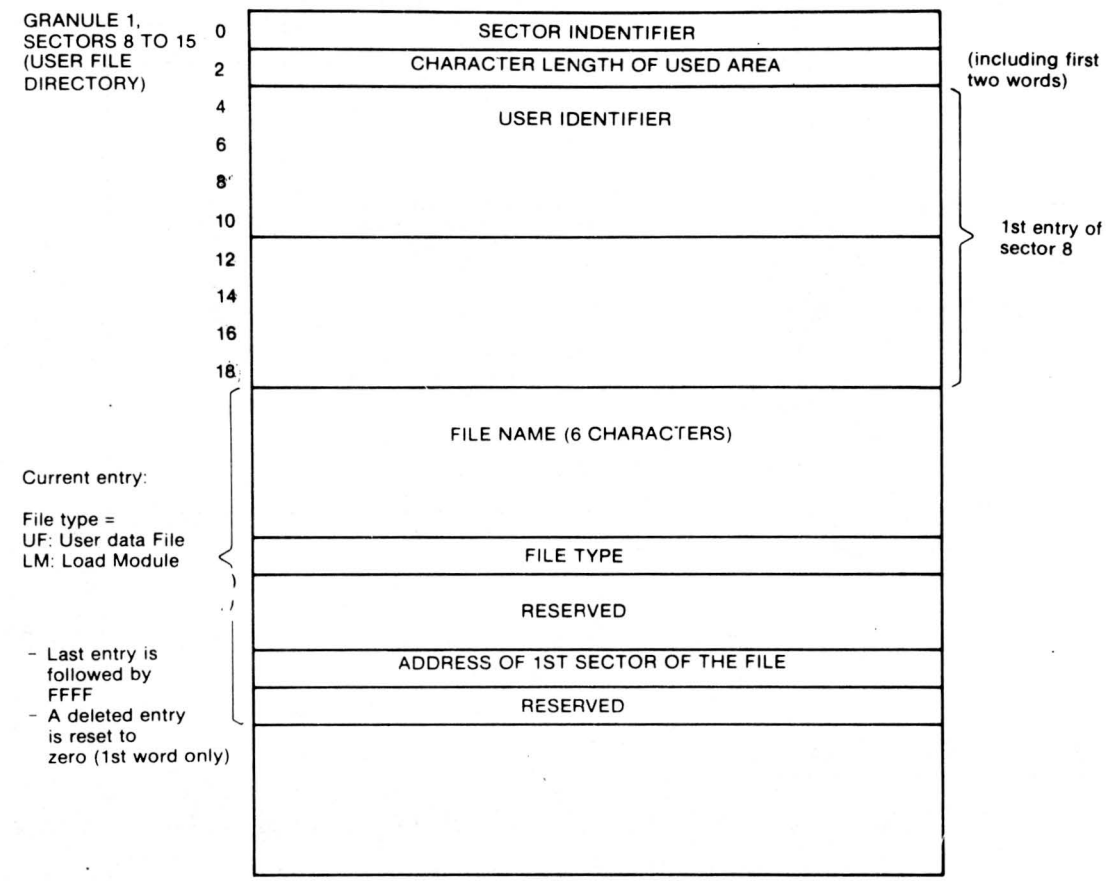
Directory Structure

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:



- 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 disc 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 UF. 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.

DRTM 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 7: Catalogue (only one entry is used under DRTM)
- Sectors 8 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: (DRTM 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: DM, 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.

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 LKM 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, so they are used in case of binary I/O. The monitor handles only the control command initialization 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 EOS or EOF 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-disc devices, I/O operations are done at record level, by I/O drivers running at level 48. No blocking- deblocking is performed. For disc devices, the user can use an I/O driver or he can access the disc 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 DRTM system are:

02:	used by the Dump routine, to output the disc or memory contents.
E0:	used to read the SCL control commands from.
EF:	used for operator communication.
F0 to FF:	used for the physical disc units.

ACCESS MODES

Two access modes are allowed for disc 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.

7 Data Management

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 EOF, 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 (GRANTB) 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

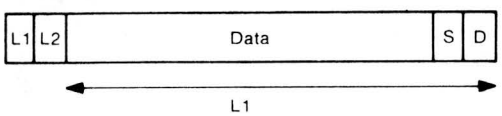
A file is sequential when the only relation between the different records is their sequence. When such a file is created, the records must be presented in the same order in which they must be written onto the disc. To access the file, it must be scanned sequentially until the desired record is found.
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 3200 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.

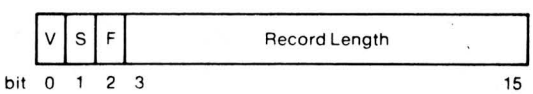
Logical Record Format

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:

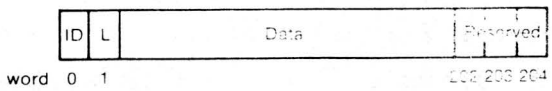
L1 is the record length, including the words S and D, but not including L1 and L2:



V = not used
S = 1, if the current record is a segment mark (EOS)
F = 1, if the current record is a file mark (EOF)
L2 = the initial record length in characters, as specified in the user's ECB (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

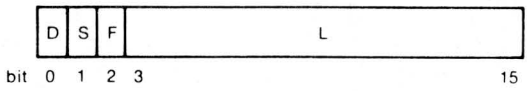
The format of a sector is as follows:



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: 4004 - 0 - S - D
(4004 = L1)

:EOF: 2004 - 0 -S - 4
(2004 = L1; 4 = D)

: Blank card: 0004 - 0050 - S - D
(0004 = L1; 0050 = L2 (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 EOF 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 (LKM 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 EOS 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.

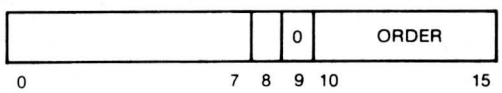
Data Management Requests

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:

```
LDK   A7,CODE
LDKL  A8,ECB
LKM
DATA 1
```

where:

the word CODE is made up as follows:



bit 8 = 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 /C010 will be returned).

ORDER contains the function code:

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

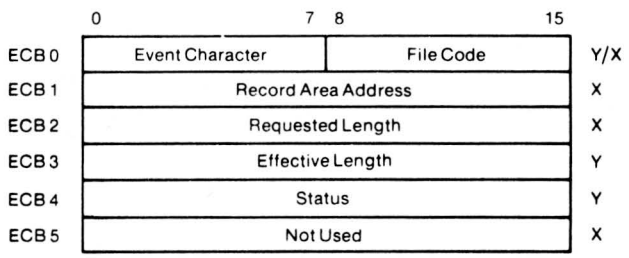
/05: Write a Record (Basic)

/06: Write a Record (Standard)
/07: Write a Record (Object, 4+4+4+4 tape format)
/08: Write a Record (Object, 8+8 tape format)
/22: Write EOF mark (Close a file)
/26: Write EOS 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 F0 to FF) or DL (disc logical files) will be returned in ECB word 1 and the word LFTMOD1 (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:



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).

ECB 4: This word contains the status returned to the user program by Data Management. See page 1-98). In addition, status /10 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 DRTM, 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 KF 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 (required by the physical disc I/O driver).

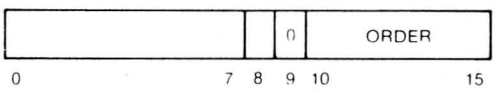
Data Management Requests

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
LDKL    A8,ECB
LKM
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 (LKM 2).

bit 9 is not used and must be reset to zero. (If it is 1, the status /C010 will be returned).

ORDER contains the function code:

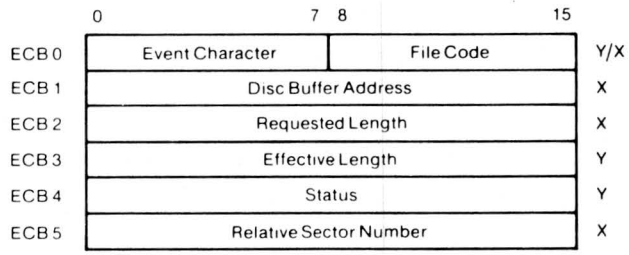
/0A: Read (Random)

/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 ASII characters 'DL' will be returned in ECB word 1 and the word LFTMOD1 (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:



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: This word is always filled with the number 410, being the number of characters in a record, which is always equal to a sector.

ECB4: This word contains the status returned to the user program by Data Management when the sector transfer is terminated. See page 1-98. 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. The Data Management module is not used for this type of access, but special orders are used in the monitor request (LKM1):

/11=physical read
/15=physical write

Moreover, the file code in the ECB must be one of the disc unit file codes /F0 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 absolute sector number 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 Disc'.

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:

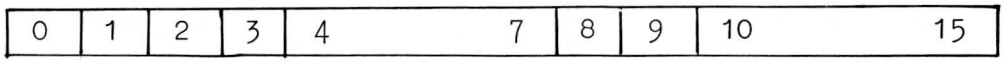
- switch on the CPU
- load the bootstrap into the first 64 memory locations manually, by means of the toggle switches and the Load Memory button on the control panel. The 64 bootstrap values can be found in Appendix E. Then check by reading these locations out.
- set up the device parameters on the toggle 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:



where:

bit 0 = 0: tape format used is 8+8
= 1: tape format used is 4x4

bit 1 = 0: the device used is not a disc
= 1: the device used is a disc

bit 2 is used only if bit 1 = 1:
= 0: the disc used is a fixed-head disc
= 1: the disc used is a moving-head disc

bit 3 = 0: the device is connected to the I/O processor
= 1: the device is connected to the programmed channel

bits 4 to 7 are used to qualify the CIO Start command sent by the bootstrap and are transferred to the addressed control unit on lines BIO 12 to 15 when required

bit 8 = 0: the control unit involved is a single device control unit
= 1: the control unit involved is a multiple device control unit

bit 9 = 1: the disc used is an X1215 disc (P824)

bits 10 = 15 contain the device address.

Initial Program Loader (IPL)

The disc IPL program is written onto the disc when the disc is premarked. It is written in absolute binary, so, to enable it to run anywhere in memory it does not contain any memory direct reference. The user must take care that the IPL, once it has been loaded into memory is not overwritten by any program it loads.
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 (188 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.

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 /980 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

  DKER 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

  BUY 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 granules 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 RSU 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 SW 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 reallocation 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?
  DYN AREA LENGTH:
  READ ONLY LENGTH:
  PROGRAM SAVE AREA:
  CORE RESIDENT AREA LENGTH:
  DKER, if a disc cannot be initialized (see below).
  DISC <address> UNKNOWN, if the disc specified by <address> is unknown by the CPU (wiring-error or non-existant).

  When the system is ready to be started, the following message will br 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.

  DKER␣<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 X1215 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.

DSK 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 (RB *) 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:DML (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 RB * instruction can be found in the SYSLINK MAP output (see System Generation): it is the address of location D:ERDG+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.
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

LOAD A MEMORY RESIDENT PROGRAM

Syntax

LD␣<name>,<disc number>[,<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.
<disc number> is the file code (/F0 to /FF) of the disc from which the program is loaded.

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

DECLARE A READ ONLY PROGRAM

Syntax

RO␣<name>,<disc number>⌈,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).
<disc number>: file code (from /F0 to /FF) of the disc on which the program can be found.
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 D: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 11: too many scheduled labels

ER 12: program already declared previously

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

ER 26: D:CI file overflow.

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

DECLARE A SWAPPABLE PROGRAM

Syntax

SW␣<name>,<disc number> [<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 D:CI file.

<name> is the name of the program.

<disc number> specifies the file code (from /F0 to /FF) of the disc on which the program is stored.

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
ER26: overflow on the D:CI file.
ER 27: 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

CN␣<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 system 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

CT␣<name>,<NTIM>,<PR>,[<NC>|<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 LKM10.
<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.
<NC> indicates the number of timer cycles before the first program activation (a hexadecimal value from 0 to /7FFF).
<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

DN␣<name>,<level>

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

Error Code
ERO1: syntax error
ERO7: level error.

DISCONNECT A PROGRAM FROM A TIMER

Syntax

DT␣<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>

Use

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

ER14: program has not been connected.
ER18: program does not exist on disc(s)
ER19: overflow on dynamic area (no possibility to build an activate block to activate background).

KEEP FILE

Syntax

KF␣<file code>,< file name>

Use

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 UF (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.

SAVE DISC ONTO MAGNETIC TAPE

Sytax

SM␣<disc number>,<file code>

Use

This command is used to have a disc of which the file code is given under <disc number> (from /F0 to /FF), copied onto the magnetic tape indicated by <file code>.
The tape can later be used to restore the disc.

Error Codes

ER01: syntax error
ER11: I/O error

ASSIGN A FILE CODE

Syntax

AS␣<file code 1>,⌊<file code 2>|DN⌈DA⌉|DKFX⌈,<file name>⌉
<number of granules>⌉⌋

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.
DFFX must be used when <file code 1> must be assigned to a file on disc with file code FX, where X is a hexadecimal number from 0 to F. It can be specified on its own or followed by <file name> or <number of granules>.

When <file name> is specified, <file code 1> is assigned to a catalogued file.
When <number of granules> 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.
When DKFX is specified on its own, <file code 1> is assigned to a sequential temporary disc file.

Note: the system file codes must all be assigned at sysgen 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 /EO, SCL assigns this file code to /EF when it finds it unassigned.
The following device names are supported by the system:
MT: magnectic 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
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

DF␣<disc number>,<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.
<disc number> is the file code (from F0 to FF) of the disc on which this file is catalogued.
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.
- BF␣<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␣<file code>[,<number>] is used to skip forward in the defined file over 1 or, if specified, a <number> of records.
- BR␣<file code>[,<number>] is used to skip backwards in the defined file over 1 or, if specified, a <number> of records.
- WF␣<file code> is used to write an end-of-file mark after the specified file.
- WS␣<file code> is used to write an end-of-segment mark.
- WV␣<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 SCL 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 by 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
CC	Request SCL	1-90
CR	Correction	1-90
DD	Dump Disc	1-90
DM	Dump Memory	1-90
HD	Halt Dump	1-91
HT	Stop CPU	1-91
RD	Release Device	1-91
RY	Retry I/O Operation	1-91
WM	Write Memory	1-92

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:

CR␣<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 number>,<sector 1>⌈,<sector 2>⌉
As a result of this command, a hexadecimal dump will be made of the contents of the disc specified by <disc number> (a value from F0 to FF). 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:

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

Stop CPU

HT
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:

```
INH
RB   *-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 HT command and still allow the current write operation to be terminated.

Retry/Release and I/O Operation:

⌊RY|RD⌋␣<device address>
When an I/O error occurs which requires operator intervention, the system prints out the message:

PU␣<device name><device addres>,<status>, RY

where RY denotes that the operator may retry the I/O operation. If the error can be corrected, the operator must give the RY 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 CIO Ŝtart Instruction to retry the last erroneous operation.

Write Memory:

WM␣<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.

11 Monitor Requests

The user program can request the monitor to perform certain functions. A request takes the form of an LKM (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 A8 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 DRTM, 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	LKM		Page
Input/Output	1		1-95
Wait for an Event	2		1-100
Exit	3		1-102
Get Buffer	4		1-103
Release Buffer	5		1-105
Connect Program to Timer	10		1-106
Disconnect Program From Timer	11		1-108
Activate a Program	12		1-109
Switch Inside a Software Level	13		1-111
Attach Device to Program	14		1-112
Detach Device from Program	15		1-114
Get Time	17		1-115
Set an Event	18		1-117
Connect Program to Software Level	20		1-118
Disconnect Program from Level	21		1-119
Wait for a Given Time	22		1-120
Assign a File Code	23		1-122
Delete a File Code	24		1-125
Read Unsolicited Operator Message	25		1-126
Cancel Unsolicited Message	26		1-128

I/O REQUEST

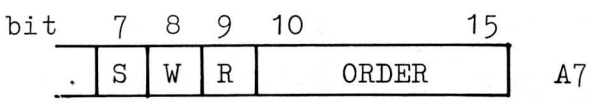
Calling Sequence

```
LDK     A7,CODE
LDKL    A8,ECBADR
LKM
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 CODE specifying the details of the I/O function, as follows:



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 soom as the transfer has been initiated. The program will give a Wait request later on for synchronization.

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

05: Basic Write.

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 8+8 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 (4+4+4+4 tape format)

08: Object Write (8+8 tape format)

Object I/O requests provide, by means of standard conversions, for special features such as error control characters, checksum and data conversion from external 4+4+4+4 or 8+8 tape format to internal 16-bit format.

0A: Random Read

0B: Random Write

14: Skip forward to EOS mark

16: Skip forward to EOF mark

22: Write EOF mark

24: Write EOV mark

26: Write EOS mark

30: Get information about a file code

31: Rewind to load point

33: Backspace one block

34: Space one block forward (not allowed for cassette)

36: Skip backward to EOF mark

38: Unlock.

(Physical disc access on sector level is possible with orders /11 (Read) and /15 (Write). The Disc File Management module is not used in these cases. The file code in ECB0 must be one of the disc file codes F0 to FF. ECB5 must contain the absolute sector number of the sector which is to be accessed. Therefore, when a disc is shared by Disc File Management and user physical disc access, extreme caution must be exercised).

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	
Y/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	TABULATION TABLE ADDR. OR RELATIVE SECT. NBR.		WORD 5

X: these words must be filled by the user
Y: these words are filled by the monitor

where:

WORD 0: event character:
bit 0 = 1: end of operation has occurred for the ECB.
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 EOS mark has been read.
  bit 15 = 1: an EOF 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 DTM 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 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

```
LDKL   A8,ECBADR
LKM
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, so 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 (LKM 18), 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 ECB 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.

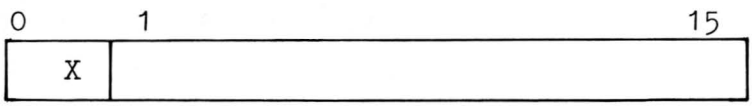
GET BUFFER REQUEST

Calling Sequence

```
LDK    A7, LENGTH
LKM
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.

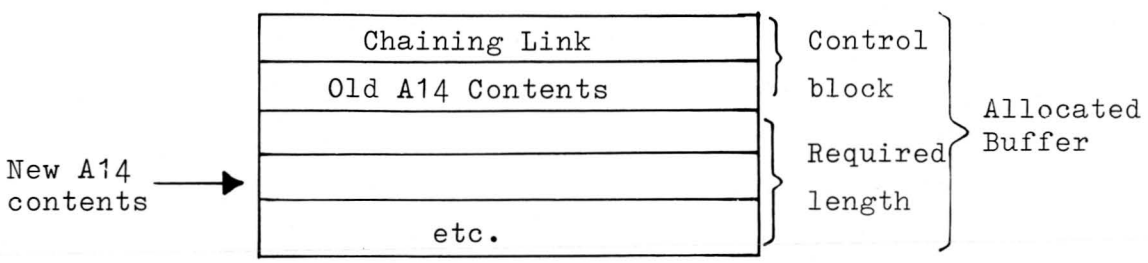
If 0 is loaded into A7, the monitor will return the upper address of memory in A7.

If the memory size is 32k, 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, BUFADR
LKM
DATA 5
```

where:

BUFADR 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.

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

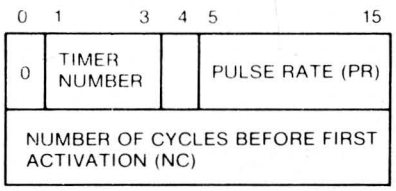
```
LDKL    A8, PARAM
LDKL    A7, PRNAME
LKM
DATA 10
```

where:

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

Two formats are possible for this block:

- format 1 (bit 0 = 0):

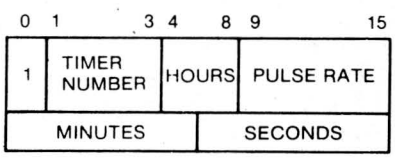


NTIM: the number of the timer (0=real time clock)

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):



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 2-51.
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 = -3;
- 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 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
LDKL  A7,PRNAME
LKM
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
LKM
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 (LKM18) 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 interrrupt 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 ≠ 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.
- disc I/O error: A7 = -7.

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
LKM
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

```
LDK    A7, FLAG
LDKL   A8, DEVBLK
LKM
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 wait request).

DEVBLK 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 <u>not</u> attached. If it is attached, the wait is done on this bit.
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.

Note:
- 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.
- 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

```
LDKL A8, DEVBLK
LKM
DATA 15
```

where:

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

FC

where:

FC 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

```
LDK  A7, FLAG
LDKL A8, TIMBLK
LKM
DATA 17
```

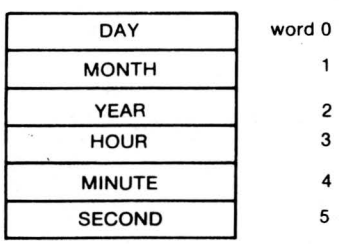
where:

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:



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

HOUR	word 0
MINUTE	1
SECOND	2
Tenth of Second	3
Fiftieth of Second	4
C:TIME	5

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

```
LDKL  A8, ECBADR
LKM
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

```
LDK    A7, NUMBER
LDKL   A8, PRNAME
LKM
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 = -3;
- 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

```
LDK    A7, NUMBER
LDKL   A8, PRNAME
LKM
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.

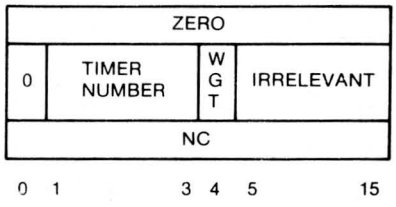
WAIT FOR A GIVEN TIME

Calling Sequence

```
LDKL   A8, PARAM
LKM
DATA 22
```

where:

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



where word 0 contains zero, as the ECB 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 LKM 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.

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.

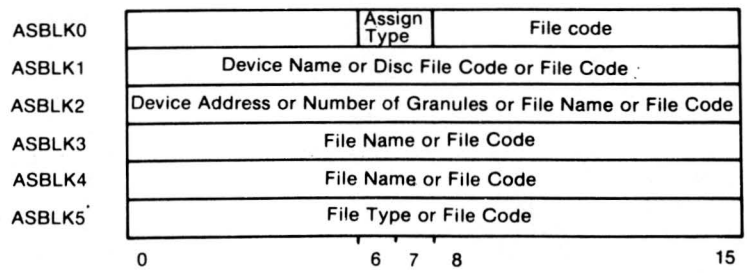
ASSIGN A FILE CODE

Calling Sequence

```
LDKL  A8, ASBLK
LKM
DATA  23
```

where:

ASBLK is a parameter block with the following layout:



ASBLK0: - Assign Type specifies the type of assignment, according to which ASBLK1 to ASBLK5 have different values.

- File Code is the file code which must be assigned. Its value may be in the range from 01 to FF. The user must be careful not to change a system file code.

Assign Type: - 00: a file code must be assigned to a physical device.
ASBLK1 contains the device name, expressed as 2 ASCII characters.

ASBLK2 contains the device address, in binary. If bit 0 of this word is set to 1, the system will assign the file code to the first device of the same name (as specified in ASBLK1) encountered in the monitor tables, regardless of the address

- 01: a file code must be assigned to a disc temporary file.

ASBLK1 contains the file code of the disc (F0 to FF) which will contain the file.

ASBLK2: = 0: the file is sequential

≠ 0: number of consecutive granules required for the direct access file.

ASBLK3 and ASBLK4 are not used.

ASBLK5 must contain the ASCII characters UF.

- 10: a file code must be assigned to a permanent file.

ASBLK1 contains the file code of the disc (from F0 to FF) on which the file is catalogued.

ASBLK2 to ASBLK4 contain the name of the file, in ASCII.

ASBLK5 contains the file type UF.

- 11: a file code must be assigned to another file code.

ASBLK1 contains the file code to which the new file code, specified in ASBLK0, is to be assigned.

In this way an equivalence is established between these two file codes.

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 DRTM 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.

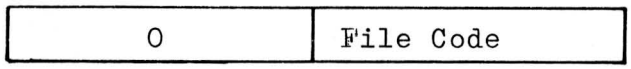
DELETE A FILE CODE

Calling Sequence

```
LDKL  A8,PARAM
LKM
DATA  24
```

where:

PARAM points to a parameter word:



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:

A7 = 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.

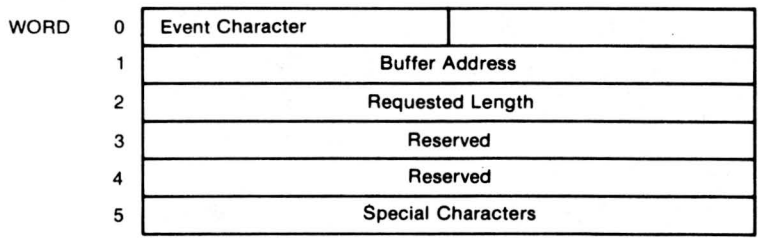
READ UNSOLICITED OPERATOR MESSAGE

Calling Sequence

```
LDKL  A8, PARAM
LKM
DATA  25
```

where:

PARAM points to a parameter block with the following layout:



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 recognize 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 /EF) 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

```
LDKL    A8, PARAM
LKM
DATA    26
```

where:

PARAM points to the same block to which is referred under the Request for an Unsolicited Operator Message (LKM25).

Use

When a monitor request is issued in a program to Read an Unsolicited Operator Message (LKM25), 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 (LKM26) can be issued to have the event count decremented and the LKM25 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.
- DISLOD, 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 sysgen 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 basis as this is the assumed standard for this sysgen 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 sysgen processor must be put on the tape reader and then loaded and started.

In the following paragraphs, the whole set of operations necessary for the generation of a DRTM 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 rema 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
  DRMGEN
  PREMDK
  GENLKE
  side B: DRTMLIB 1
- cassette G2: side A: DRTMLIB 2
  DISLOD
  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:
  - 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	0	1	0	0	0	0	0	(hexa 1020)

(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 sysgen program from the cassette in TK05 into memory:

GENMON

GENMON is a special monitor used only during the sysgen 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 possibilitieswhich 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⌈ES⌉ or N⌈O⌉.
If the user replies Y or YES followed by (LF) (CR), GENMON assumes the following standard configuration definition:

- typewriter : TY10 at level /6
- tape reader : PR20 at level /4
- tape punch : PP30 at level /5
- 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
- X1215 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 in 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 Y[ES] or N[O].
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 E2: TK05 (Disload object input)
- file code EF: 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 NO)
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 (LF) (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: DRMGEN.

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. PTR, 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 P842, because they work in echo mode.

DRMGEN

When GENMON initialization is terminated and the message READY TO LOAD PROGRAMS has been output the user can start the second phase, DRMGEN, the building of the DRTM 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 W1) 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 M: type in
  LD
- now the following sysgen processor, i.e. DRMGEN is loaded from the cassette or paper tape and its identification is output:
  IDENT 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 /580.
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_{10} words during program loading;
- to assign a disc logical file, the system requires 205_{10} words during the assign process;
- when a disc sequential file is used the system requires 205_{10} 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_{10} 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_{10} 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_{10} 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: TG03: 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: TG03: 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: TG03: the reply is not a hexadecimal number.

SWAPPING FILE CODE

Reply: Specify the file code of the disc (from /F0 to /FF) 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: TG03: 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 = program length, then the number of

sectors required for this program is:

$$\frac{L - 1}{200_{10}} + 1$$

- Swappable Programs:
  if S = length of the swap area, the number of sectors required for each swappable program is:

$$\frac{S - 1}{200_{10}} + 1$$

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} + 2m words, where m is the same value which is given in reply to the question MAX NUMBER OF SCHEDULED LABLES

Error Message: TG03: invalid reply.

USER INTERRUPT ROUTINES

Reply: Specify all user interrupt routines (usually drivers for special devices) which must be link-edited 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: TG03: syntax error
TG06: the first character of the name (ENTRYi/j) is not alphabetic
TG09: 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: TG03: 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[O] nor EN[D] . 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: TG03

LKM LEVEL

Reply: First specify the level to which the LKM 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). DRMGEN 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 DFM at DRMGEN, an I/O monitor request (LKM1) on a logical disc file may cause a system hang-up. Therefore users should define exactly the required functions.

Error Message: TG06: 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. DRMGEN 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.
TABGEN prints the following list:

CNTM (Connect a Timer)
DNTM (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)
DEFC (Delete a File Code)
KEYN (Read Unsolicited Key-in)
CNSL (Cancel Read Unsolicited Key-in)

Error Message: TG06: 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:

<N1>,<ENTRY1>
<Ni>,<ENTRYi>
<Nn>,<ENTRYn>

END

where Ni consists of two hexadecimal digits defining the DATA number which follows the LKM instruction, and ENTRY 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: TG03: the first parameter is not hexadecimal
TG07: the first character of the second parameter is not alphabetic
TG08: syntax error

PCT 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 assignations for that program or not.
If he types N, he must also type the level and

assignations 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

PNRLKM (possibly non-resident monitor requests): D:USV1, D:USV2, D:USV3 and D:RMAC modules.

GR ALLOC (granule allocation): D:ALGR and D:CTPN 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, DRMGENprints and implements the following list:

TIMER	: 31	CR
LOADER	: 31	CR
PNRLKM	: 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 18_{10} words. The user need not reserve entries for the system programs.

Error Message: TG03: 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:

⟨DNDA⟩,⟨L⟩

!

END

where:

DN 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: 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: TG01: the device specified is not supported by the system.

TG02: device address error

TG09: level error

TG10: 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:
  LDPA,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 wanted.

No check is made on the device declaration.

DISK UNITS

Reply: Specify all disc units connected to the I/O processor as follows:

DNDA,L,Q ⌈,DAL⌉

END

where:

DN is the device name, giving the type of disc on 2 ASCII characters as:

BM: X1215 moving-head disc (model 1)

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, on 2 hexadecimal digits.
L specifies the interrupt level to which the disc is connected (1 or 2 hexadecimal digits)

Q defines, on 2 hexadecimal digits, the length of the disc queue table, in number of entries
DAL defines the length of the disc allocation table in characters.

Note:

- The disc queue table is used only if DFM 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.
- DAL must be 0 if DFM has not been selected (except for disc units containing the D:CI file). For disc units on which no logical I/O operations will be performed, 0 must be specified also. Where used, the maximum value of DAL depends on the type of disc:
  X1215: /66

END is specified to indicate that all disc units have been declared.

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:

```
<FC>,<DNDA>
    ¦
<FC>;<DNDA>

   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 CODES ASG 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 GENLKE 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 ASG 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 DRMGEN for each user device and it must not be reassigned under the DRTM.

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 FCT

Reply: Specify the number of spare entries reserved in the File Code Table (FCT), 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 LKM 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_{10} 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 FILLAB 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

PREMDK

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.
PREMDK outputs a number of questions on file code /EF (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 PREMDK tape on the reader; make it operable
- push the INT button on the CPU control panel
- on output of M: type in
  LD
- now PREMDK is loaded as the next processor from the cassette G1 in drive TK05 or the tape reader and its identification is output
  IDENT PREMDK
- when loading is terminated,
  :EOS
  :EOF
  is output on the typewriter.
- push the INT button
- on output of M: type in
  ST
- now PREMDK is started and the following messages and questions are output on the typewriter:
  INITIALIZATION OF PREMRK
  NBR OF CYLINDERS = (type in 4 decimal digits giving the number of cylinders on the disc, followed by (LF) (CR) Example: 0203)
  NBR OF TRACKS = (type in 4 decimal digits giving the number of tracks per cylinder, followed by (LF) (CR) Example: 0002)
  NBR OF SECTORS/TRACK = (type in 4 decimal digits giving the number of sectors per track, followed by (LF) (CR) Example: 0016)
  DISK TYPE = (for the X1215, this must be CM followed by (LF) (CR))
  DISK UNIT PHYSICAL ADDRESS = (type in two hexadecimal digits giving the physical address of the disc, followed by (LF) (CR) Example: 32)

LABEL = (type in a volume label, consisting of up to 8 characters, followed by (LF) (CR) Example: TONTO)

DATE = (type in 6 decimal characters, separated by delimiters, e.g. spaces or slashes, followed by (LF) (CR) Example: 01/01/84)

PACK NBR = (type in 3 characters giving the number of the disc pack, followed by (LF) (CR) Example: F32)

SYSTEM USERID = (type in up to 8 ASCII characters, specifying the first user of this disc pack, i.e. the user identification of the system. This is to enable the pack to be loaded by DISLOD. It must be followed by (LF) (CR) Example: SILVER) It is mandatory to answer this question.

When the user has answered these questions, PREMDK starts premarking the disc and outputs the messages:

WRITING THE IDENTIFIERS

CHECKING THE IDENTIFIERS

END OF CHECK

NBR OF BAD GRANULES = XXXX

RUN AGAIN? (to this last question the user must type in NO if he wants no other disc premarked; if he types in OK, the above procedure is repeated; after NO PREMDK types out its last message:)

END OF PREMRK

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.

GENLKE

During this phase the final user monitor is obtained by linking the tables generated under DRMGEN 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 W2 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:
  IDENT GENLKE
- when loading is terminated,
  :EOS
  :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 DRMGEN) 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> [,[8|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

8 or 4 is used if the monitor is punched on paper tape to indicate whether it must be punched in 4-track ot 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

  P

  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 (DRMLIB1) 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 INIMON 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 G2 in drive TK05 with side A up and wait for it to be rewound
- 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

  T

  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 monito 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 G1 from drive TK05 and put it back in with side B up. Wait for it to be rewound.
- put cassette W1 (tables output by DRMGEN) into drive TK25 and wait for it to be rewound
- put the second working cassette W2 in drive TK15
- 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 DRMGEN. See also under GENMON for the file code assignments.
- when this is accepted, GENLKE outputs L: and the user types
  H
  Upon this command, GENLKE scans the input file (i.e. the cassette 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 it 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 IPL 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 DRMGEN is put on the reader and GENLKEBis 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 (+ ASEX) is the last unsolved reference with a U command and having typed L to solve it, the user then types T 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.

DISLOD

DISLOD 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 DRTM.
The standard object input file code for DISLOD 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 DISLOD tape on the tape reader and make it operable
- with cassette, the cassette tape in file code /E2 is positioned after the DRMLIB 2, i.e. before DISLOD)
- push the INT button on the CPU control panel
- on output of M: type in LD
- now DISLOD is loaded from the tape reader or the cassette and its identification is output:
  IDENT DISLOD
- when loading is terminated,
  :EOS
  :EOF is output on the typewriter.
- with paper tape, DISLOD tape must now be taken from the reader and the newly generated DRTM 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 DRTM) which must first be taken from file code /3 (normally TK15).
- push the INT button
- on output of M: type
  ST
  to start the DISLOD processor.
- DISLOD then outputs the message
  SYSLOAD XX P852
  on the typewriter, followed by the question
  NEXT ACTION:
- at this point, the user has three possibilities for action:
  - if he types (LF) (CR) the next program or module on the input file (/E2) will be recorded onto the system disc. DISLOD will output the program name onto the typewriter (PROG.NAME = XXXXXX) and list name, length in sectors,

start address and program length on the listing device (/2).

After this, DISLOD will again output the message NEXT ACTION:.

- if he types PS, DISLOD goes into Pause state, enabling the user to modify an assignment, operate on the cassettes by manual control, etc. To restart DISLOD, 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, DISLOD performs an Exit and the process is stopped.

- so, after the first NEXT ACTION: message output by DISLOD, the user types (LF) (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, DISLOD 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 D:USV1 tape (see list of sysgen 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 (LF) (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 (LF) (CR)
- when the last one (IPLGEN) has been recorded onto the system disc and DISLOD again types
  NEXT ACTION:
  the user must type in
  HT
  to terminate the DISLOD operation.

The user now has a Disc Real Time Monitor on disc.

DISK PREMARK

PREMRK 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 and writes identifiers in them and it checks for bad tracks.

PREMRK 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 typewriter and the user can type in his answers:

NBR OF CYLINDERS
Type in 4 decimal characters, specifying the number of cylinders on the disc, followed by LF CR

NBR OF TRACKS
Type in 4 decimal characters, giving the number of tracks per cylinder, followed by LF CR

NBR OF SECTORS PACK =
Type in 4 decimal characters, specifying the number of sectors per track, followed by LF CR

DISK TYPE =
Type in 2 characters, specifying the type of disc unit, and the type of channel used, as follows:

CM X1215 on I/O processor

then type in LF CR

DISK UNIT PHYSICAL ADDRESS
Type in two hexadecimal characters, followed by LF CR

LABEL
Type in 8 characters, giving the volume label, followed by LF CR

DATE
Type in 6 decimal characters, specifying the date, followed by LF CR

PACK NBR
Type in 3 characters to specify the pack number, followed by LF CR

SYSTEM USERID =
Type in the identification of the system generated, followed by LF CR.

RUN AGAIN?:
Type in NO if no other discs are to be formatted, or OK if another disc must be done. In the second case, the above procedure is repeated.

Example:

```
INITIALISATION OF PREMRK
NBR. OF CYLINDERS = 0203
NBR. OF TRACKS = 0002
NBR. OF SECTORS/TRACK = 0016
DISK TYPE: CM
DISK UNIT PHYSICAL ADDRESS = 02
LABEL = EXAMPLE
DATE = 12/02/74
PACK NBR. = 001
SYSTEM USERID = SAG
- WRITING THE IDENTIFIERS
- CHECKING THE IDENTIFIERS
- END OF CHECK
- NBR. OF BAD GRANULES = 0000

RUN AGAIN?: NO
END OF PREMRK
```

The user replies are given in boldface.

Error Messages:
BAD GRANULE ZERO (granule zero of the pack is bad and therefore this pack is not usable)
NO SYSTEM USER POSSIBLE (granule one of the pack is bad and therefore no system catalogue can be opened).

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.

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 (2 dummy characters must be reserved behind the buffer).
- 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 (4 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, /1 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+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).

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 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.

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 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 EOV 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 EOV status must be taken into account.

Note:
In case the EOT is detected while reading an EOV, only the EOV status is returned.

RETURN INFORMATION ABOUT A FILE CODE (/30)

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
- 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.

Note: (physical files; F0-FF)

If this order is used for a disc file code, the system will return the device name DK or DL (logical files) in ECB word 1 and fill the other words of the ECB with zeros.

OFF LINE (/38)

Magnetic tape:
This order switches the machine off.

Cassette Tape:
This order unlocks the cassette from the drive unit.

Appendix D Control Unit Status Word Configuration

Bit	Description	Control unit							
		ASR	CR	DM	LP	PTP	PTR	CASS Tape	MT
0	–								
1	has become ready			x				x	x
2	rewinding								x
3	tape mark has been read							x	x
4	no data							x	
5	on cylinder			x					
	beginning of tape							x	
6	seek error			x					
	write unable							x	x
7	A or B side							x	
8	Device Address							x	x
9	Device Address			x				x	x
10	EOT						x	x	x
	tape low					x			
11	Program error			x				x	x
12	incorrect length		x	x				x	x
13	Parity error							x	x
	data fault		x	x					
14	throughput error	x	x	x			x	x	x
15	not operable (only significant bit for TST)	x	x	x	x	x	x	x	x

DM = moving-head disc

```
00000                    IDENT    BEBOOT
00001          *
00002          *
00003          *         DISPLAY THE KEYS AS FOLLOWS:
00004          *
00005          *         BITS     MEANING
00006          *         0=1      IPL LOADED FROM ASR , FORMAT 4*4
00007          *         1=1      DISK,THEN
00008          *         2=1      MOVING HEADS
00009          *         2=0      FIXED HEADS
00010          *         3=1      PROGRAMMED CHANNEL
00011          *         3=0      I/O PROCESSOR
00012          *         4 TO 7   BOU LINES ( 4 RIGHTMOST BITS )
00013          *         8=1      MULTI DEVICE CONTROLLER
00014          *         8=0      SINGLE DEVICE CONTROLLER
00015          *         9=1      X1215 DISK
00016          *         10 TO 15 DEVICE ADDRESS
00017          *
00018          *
00019          *
00020          *         DESCRIPTION
00021          *
00022          *         BEBOOT LOADS ONE RECORD ONTO LOCATION /80 THEN START AT /84
00023          *         THE RECORD IS THE SECTOR # 1 IF DISK,OR 254 CHARACTERS OF THE
00024          *         INPUT DEVICE,LEADING NULL CHARACTERS IGNORED
00025          *
00026          *
00027          *
00028          *         USED REGISTERS :
00029          *
00030          *         A1 BOU LINES,NOT TO BE DESTROYED IF BOOT IS CALLED AGAIN
00031          *         A2 ADDR OF INR INSTRUCTION
00032          *         A3 ADDR OF CIO INSTRUCTION(WHICH IS DESTROYED AND NEEDS TO BE
00033          *                              RESTORED IF BOOT IS CALLED AGAIN)
00034          *         A4 ADDR OF SST INSTRUCTION
00035          *         A5 MULTIPLEX : CONTENTS OF 1ST WORD TO BE SENT TO EXT REGISTER
00036          *            IO BUS : CHARACTER COUNT,INITIALIZED AT 254 AND DECREMENTED
00037          *         A6 MULTIPLEX : CONTENTS OF 2ND WORD TO BE SENT TO EXT REGISTER
00038          *                              (LOADING ADDR)
00039          *            IO BUS : ADDR OF NEXT CHAR TO BE LOADED,INIT AT /80 AND
00040          *                              INCREMENTED
00041          *         A7,A8 WORK REGISTERS
00042          *         A9 WORK REGISTER
00043          *         A10 TO A14 NOT USED
00044          *         A15 CONTAINS THE KEYS'VALUE
00045          *
00046          *
00047          *
00048          *
00049          *
```

```
00050
00051
```

* *

```
00052                                   EJECT
00053                                   ADRG    0
00054                           *
00055                           *
00056                           BOOT    EQU     *
00057                           *       INITIALIZE REGISTERS
00058  0000  0200  F                    LDK     A2,INR          ADDR OF INR INSTRUCTION
00059  0002  0300  F                    LDK     A3,CIO          ADDR OF CIO INSTRUCTION
00060  0004  0400  F                    LDK     A4,SST          ADDR OF SST INSTRUCTION
00061                           *                               EXTRACT DEVICE ADDR AND INIT I/O COMMANDS
00062  0006  861E                       LDR     A6,A15
00063  0008  263F                       ANK     A6,/3F          DEVICE ADDR
00064  000A  9629                       ADRS    A6,A2           INITIALIZE I/O INSTRUCTIONS
00065  000C  962D                       ADRS    A6,A3
00066  000E  9641                       ADS     A6,MIO
       0010  0000  F
00067                           *                               EXTRACT CONTROLLER ADDR AND INIT WER INST
00068  0012  871E                       LDR     A7,A15
00069  0014  3FC6                       SLC     A7,6            MULTI OR SINGLE DEVICE CONTROLLER
00070  0016  5600  F                    RF(6)   INIT20          SINGLE ONE
00071  0018  260F                       ANK     A6,/F           MULTIPLE ONE
00072                           INIT20  EQU     *               INITIALIZE MULTIPLEX DBLE WORDS
00073  001A  9631                       ADRS    A6,A4           SST INSTRUCTION
00074  001C  3E41                       SLL     A6,1
00075  001E  9641                       ADS     A6,WER1         SET UP WER INSTRUCTIONS
       0020  0000  F
00076  0022  9641                       ADS     A6,WER2
       0024  0000  F
00077                           *                               LOAD A1 WITH BOU CONTENTS
00078  0026  811C                       LDR     A1,A7
00079  0028  0550                       LDK     A5,80           MULTIPLEX DOUBLEWORDS LOAD 80 CHAR INTO
00080                           *                               LOCATION /80
00081  002A  0680                       LDK     A6,/80
00082                           *                               CHECK IF DISK
00083  002C  3FE7                       SRC     A7,7
00084  002E  5600  F                    RF(6)   NODISK          NO
00085                           *                               FIXED HEADS ?
00086  0030  3FC1                       SLC     A7,1
00087  0032  5600  F                    RF(6)   NOSEEK          YES
00088  0034  0103                       LDK     A1,3            SEEK ZERO
00089  0036  41C0               CIO     CIO     A1,1,0          CIO SEEK ZERO
00090                           NOSEEK  EQU     *
00091  0038  811E                       LDR     A1,A15
00092  003A  3966                       SRL     A1,6            SECTOR NUMBER
00093  003C  213C                       ANK     A1,/3C
00094  003E  8520                       LDKL    A5,/80CD        1ST WORD OF MULTIPLEX FOR DISK DEVICE
       0040  80CD
00095                           NODISK  EQU     *
00096                           *                               EXECUTE WER,WHATEVER THE CHANNEL IS
00097                           WER1    EQU     *
```

```
00098  0042  7500          WER     A5,0
00099                WER2    EQU     *
00100  0044  7601          WER     A6,1
00101  0046  F031          EXR*    A4          SST
00102  0048  F02D          EXR*    A3
00103  004A  5C06          RB(4)   *-4
00104  004C  871E          LDR     A7,A15
00105  004E  3F43          SLL     A7,3
00106  0050  5600  F       RF(6)   SST         MULTIPLEX
00107                *
00108                *               IO BUS
00109                *
00110  0052  8194          LDR     A9,A5       IF A9=A5 IGNORE LEADING CHAR.
00111                INR     EQU     *
00112  0054  4F00          INR     A7,0,0      READ ONE CHAR
00113  0056  5C04          RB(4)   *-2
00114  0058  E994          CWR     A9,A5       LEADING CHAR?
00115  005A  5400  F       RF(4)   INR10       NO
00116  005C  27FF          ANK     A7,/FF      CHECK IF NULL
00117  005E  580C          RB(0)   INR         YES,IGNORE
00118                *               NO,CHECK IF 4*4
00119                *
00120                INR10   EQU     *
00121  0060  879E          LDR     A15,A15     4*4
00122  0062  5600  F       RF(6)   STORE       NO 8*8
00123  0064  3F44          SLL     A7,4        YES,4*4
00124  0066  809C          LDR     A8,A7       SAVE LEFT BITS
00125  0068  F029          EXR*    A2          READ NEXT CHARACTER
00126  006A  5C04          RB(4)   *-2
00127  006C  270F          ANK     A7,/F       GET 4 RIGHT MOST BITS
00128  006E  9702          ADR     A7,A8
00129                STORE   EQU     *
00130  0070  E739          SCR     A7,A6       STORE CHAR
00131  0072  1601          ADK     A6,1        NEXT CHAR ADDR
00132  0074  1D01          SUK     A5,1        COUNT DONE ?
00133  0076  5924          RB(1)   INR         NO
00134                *               YES
00135                *
00136  0078  4180          HIO     CIO     A1,0,0
00137                *
00138                STATUS  EQU     *
00139  007A  4FC0          SST     SST     A7,0
00140  007C  5C04          RB(4)   *-2
00141  007E  0F84          AB      /84
00142                *
00143                *
00144                *
00145                        END     BOOT
```

```
          SYMBOL TABLE

  BOOT    0000  A   INR     0054  A   CIO     0036  A   SST     007A  A
  HIO     0078  A   INIT20  001A  A   WER1    0042  A   WER2    0044  A
  NODISK  0042  A   NOSEEK  0038  A   INR10   0060  A   STORE   0070  A
  STATUS  007A  A

          ASS,ERR,    00000
:EOF
PROG ELAPSED TIME:  00H-00M-00S-000MS-
BEA IPL52S
DATE     /   /     TIME 24H-60M-60S-
```

Command	Meaning	Page
AS	Assign a File Code	1-84
BF	Space File Backwards	1-87
BR	Space Record Backwards	1-87
CN	Connect a Program to a Software Level	1-79
CT	Connect a Prgram to a Timer	1-80
DF	Delete a File	1-86
DL	Delete a File Code	1-86
DN	Disconnect a Program from a Level	1-81
DT	Disconnect a Program from a Timer	1-81
EN	End of Commands	1-88
FF	Space File Forward	1-87
FR	Space Record Forward	1-87
KF	Keep File	1-83
LD	Load a Memory Resident Program	1-76
Magnetic Tape Control Commands		1-87
RO	Declare a Read Only Program	1-77
RW	Rewind Tape	1-87
SC	Set Clock	1-87
SD	Set Date	1-82
SM	Save Disc onto Magnetic Tape	1-84
ST	Start a Program	1-83
SW	Declare a Swappable Program	1-78
TS	Define Time Slice	1-79
UN	Uload Tape	1-87
WF	Write End-Of-File Mark	1-87
WS	Write End-Of-Segment Mark	1-87
WV	Write End-Of-Volume Mark	1-87

Appendix G Operator Control Messages

Message	Meaning	Page
CC	Request SCL	1-90
CR	Correction	1-90
DD	Dump Disc	1-90
DM	Dump Memory	1-90
HD	Halt Dump	1-91
HT	Stop CPU	1-91
RD	Release Device	1-91
RY	Retry I/O Operation	1-91
WM	Write Memory	1-92

Request	LKM	Page
Input/Output	1	1-95
Wait for an Event	2	1-100
Exit	3	1-102
Get Buffer	4	1-103
Release Buffer	5	1-105
Connect Program to Timer	10	1-106
Disconnect Program from Timer	11	1-108
Activate a Program	12	1-109
Switch Inside a Software Level	13	1-111
Attach Device to Program	14	1-112
Detach Device from Program	15	1-114
Get Time	17	1-115
Set an Event	18	1-117
Connect Program to Software Level	20	1-118
Disconnect Program from Level	21	1-119
Wait for a Given Time	22	1-120
Assign a File Code	23	1-122
Delete a File Code	24	1-125
Read Unsolicited Operator Message	25	1-126
Cancel Unsolicited Message	26	1-128

PART 2 DRTM CONFIGURATION

The Disc 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 DRTM 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 (LKM1)
- Activate (LKM12)
- Exit (LKM3)

- Set Event (LKM18)
- Get Buffer/Release Buffer (LKM 4/5)
- Wait for an Event (LKM2)
- Switch inside a Program (LKM13)

They all run at level 48 and thus make use of the A15 stack just as interrupt routines.

System Tables

The following tables are part of the DRTM 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 18 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.
- 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.

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:PFAR)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 /E0 (control command input) is assigned to the typewriter. When /E0 is assigned to another device, the SCL routines must be connected to a level between that of D:0COM +1 (see below) and 61. Output from the SCL takes place on file code /EF, 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:CTPN reads the operator commands and activates the routine required to process a specific command.
  D:CTPN 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:0COM processes various commands, especially RY 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:0COM 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:0COM may not be loaded.
  Therefore, D:0COM 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:0COM 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 48 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 48, control will be returned to the interrupted routine. For software levels (>48), 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.

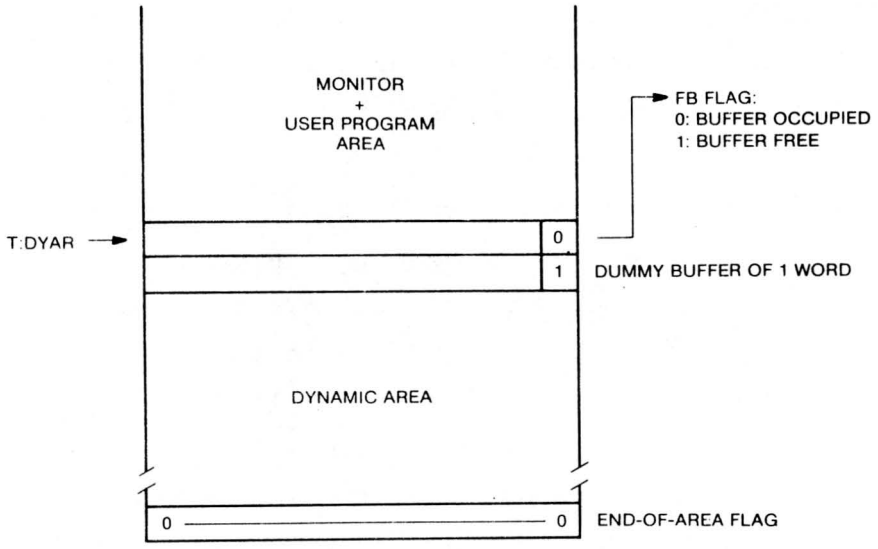
MEMORY ORGANIZATION

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.

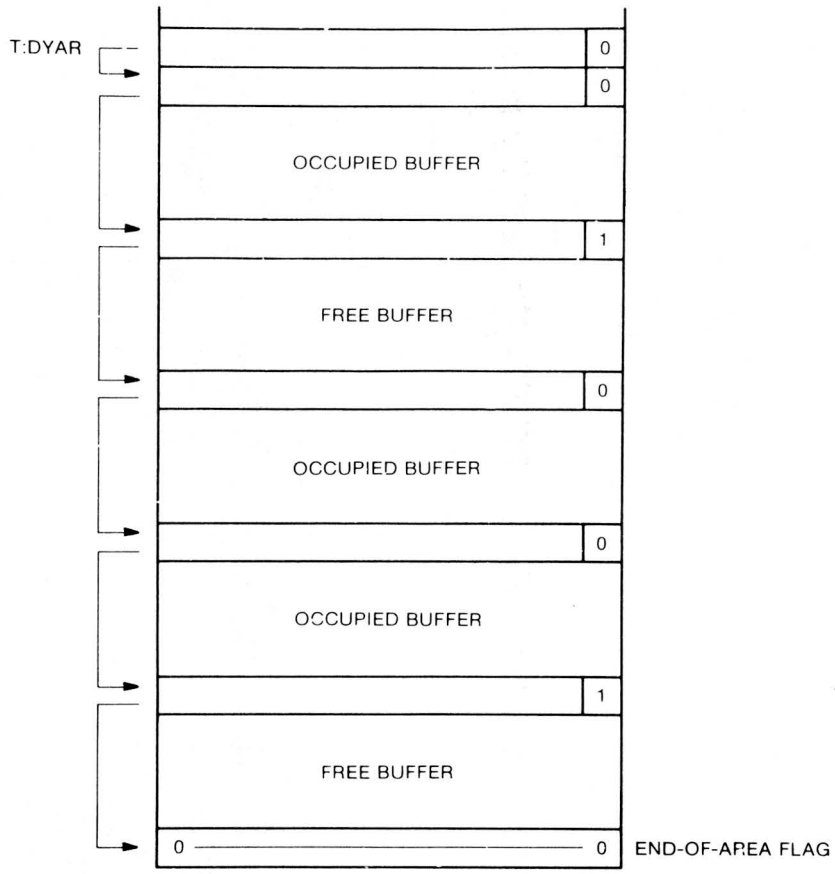
Dynamic Allocation Area

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 paritions. The minimum size of the area,

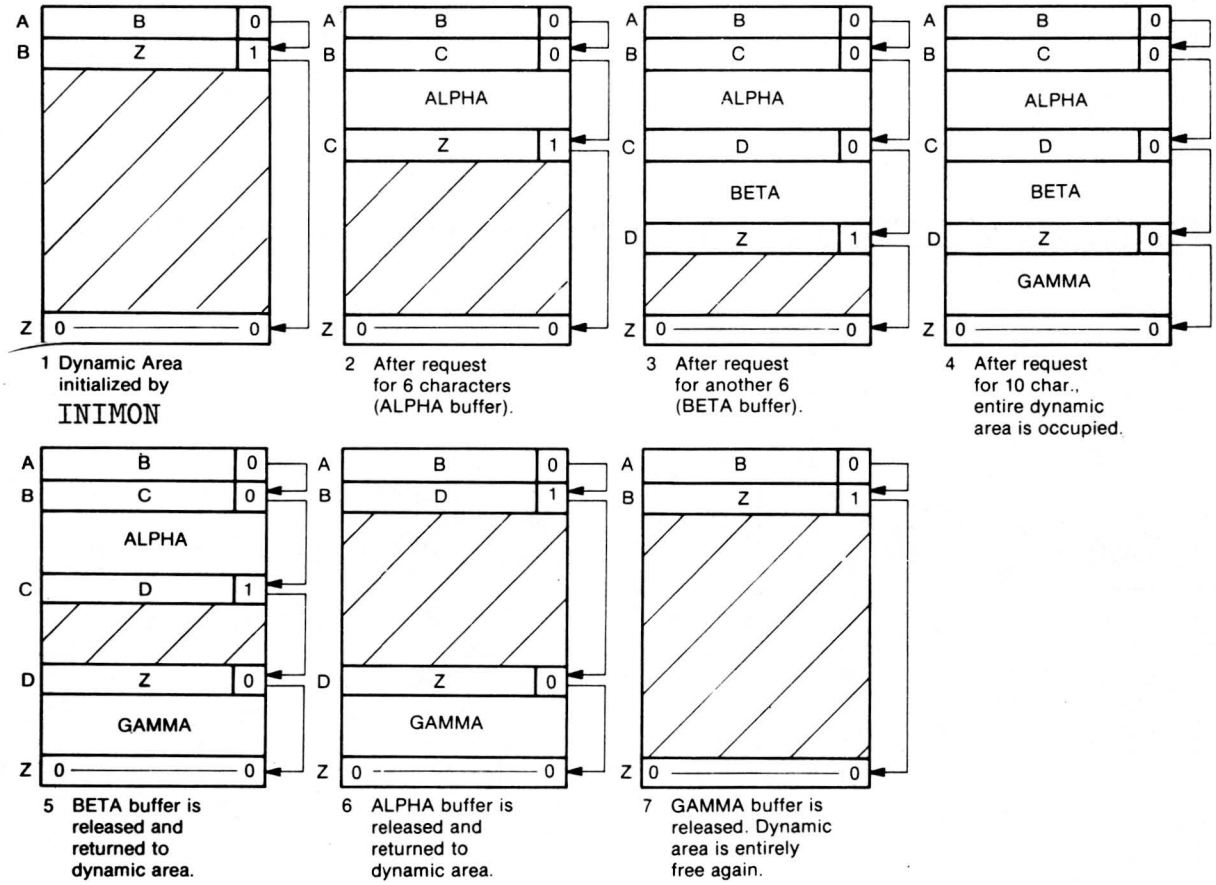
to be specified at system generation time, is 704 words. INIMON gives the following layout to this area, which is to be used for dynamically allocating buffer space to monitor or user programs:



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.

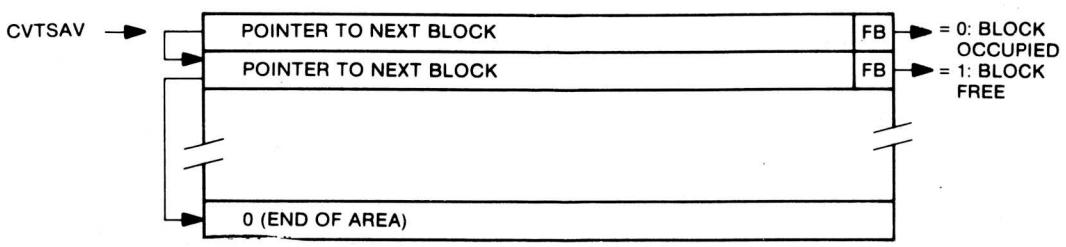

A
B
Z
B
0
1
0
0
1 Dynamic Area initialized by INIMON
A
B
C
Z
B
C
0
0
ALPHA
1
0
0
2 After request for 6 characters (ALPHA buffer).
A
B
C
D
Z
B
C
D
0
0
0
ALPHA
BETA
1
0
0
3 After request for another 6 (BETA buffer).
A
B
C
D
Z
B
C
D
0
0
0
ALPHA
BETA
GAMMA
0
0
4 After request for 10 char., entire dynamic area is occupied.
A
B
C
D
Z
B
C
D
0
0
0
ALPHA
GAMMA
1
0
0
5 BETA buffer is released and returned to dynamic area.
A
B
D
Z
B
D
0
1
0
GAMMA
0
0
6 ALPHA buffer is released and returned to dynamic area.
A
B
Z
B
0
1
0
0
7 GAMMA buffer is released. Dynamic area is entirely free again.

Swap Area

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 CVTSWP 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.

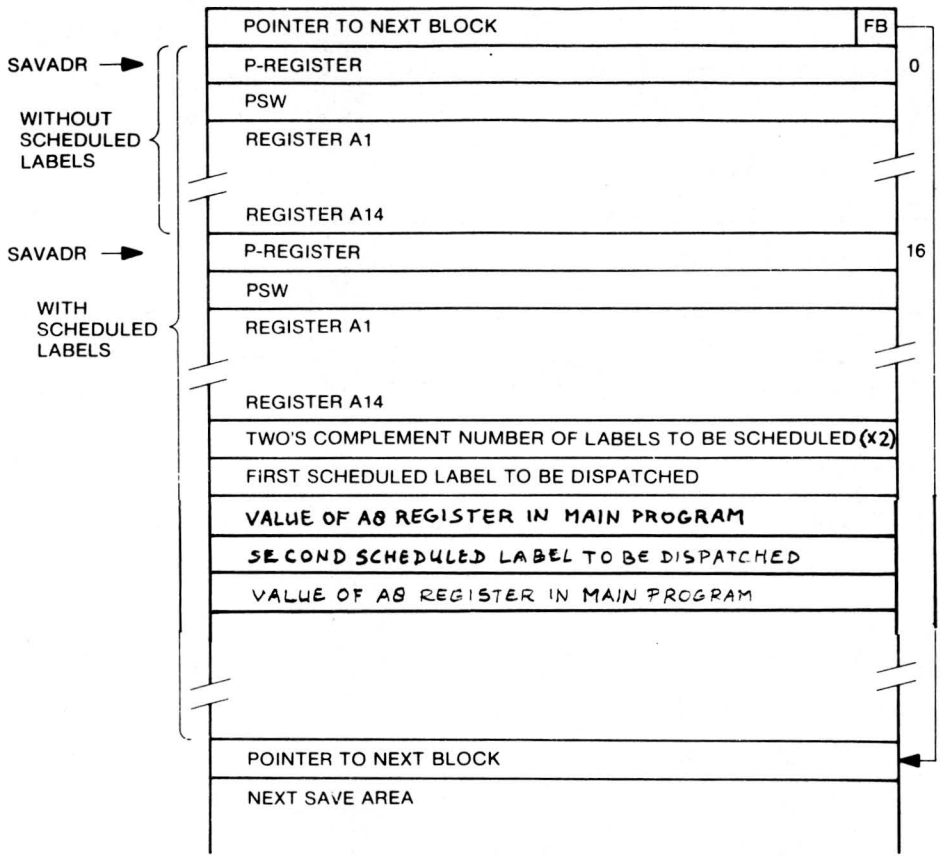
Program Save Area

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 INIMON 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 lables allowed in queue (FILLAB, 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.

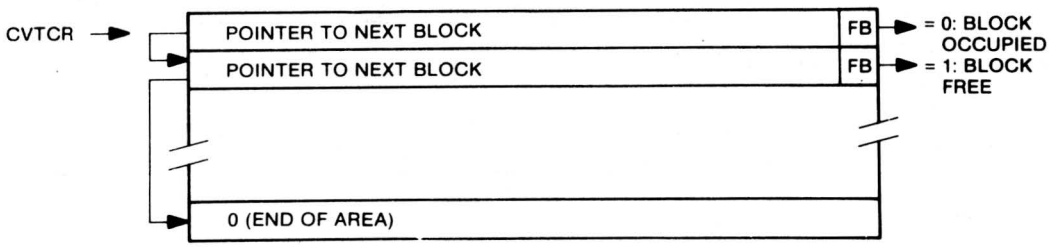
Read Only Area

This area is located behind the Program Save Area in memory. Its length is defined at system generation and must be at least 1088 words. The word CVTRDO 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 CVTRDO.

Memory Resident Area

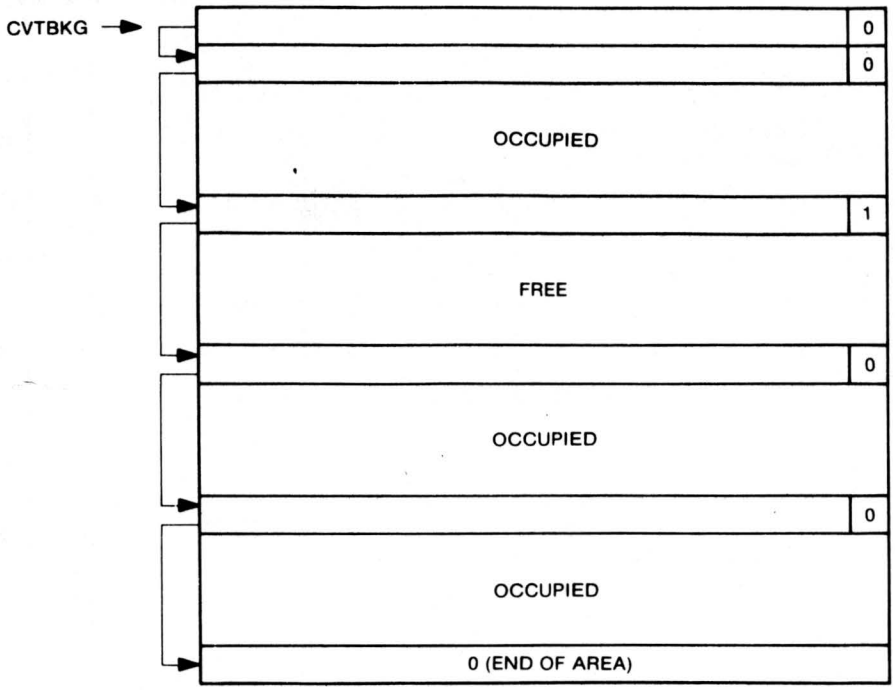
This area is used for memory resident programs.
At system generation time, the location CVTCR in the Communication Vector Table contains the length of this area.
After loading, it points to the first word of this area, which is initialized by the INIMON program as follows:



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 organized 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
- SEC

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 E 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 SEG 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:FCT) the monitor will find the address of a Device Work Table (DWT) or Logical File Table (T: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:RMAC and T:LKM 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.

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:CMSZ
T:CSTB
T:CDSP
T:CIDP
T:CIDA
T:CSLM
T:CTIM
T:CPLS
T:CRST
T:DYAR
CVTFCT
RESERVED
CVTDWT
CVTDCT
CVTLFT
CVTPCT
CVTSLT
CVTSAV
CVTRDO
CVTCR

CVTBKG
CVTOCM
CVTKEY
CVTLAB
CVTCST
CVTDMA
CVTDML
CVTDYE
CVTBKE
CVTCRE
CVTSAE
CVTSSY
CVTSSY+2
CVTSSY+4
CVTSSY+6
CVTSSY+8
CVTSSY+10
CVTSSY+12
CVTSSY+14
CVTPWF
CVTSWP
CVTTS
CVTCID
CVTCIF
CVTCIL
CVTTSC
CVTFDC
CVTEFT
CVTWAT
CVTSET

T:CMSZ 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:CSTB contains the base address of the system stack, as defined at system generation time.

T:CDSP contains the address of the dispatcher.

T:CIDP contains the status word of the Idle Task.

T:CIDA 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.

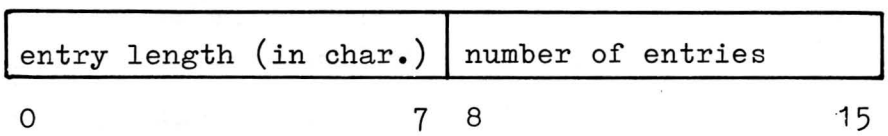
T:CPLS contains the number of pulses of the timer during one 20 msec period (PR): 1 for the 50Hz standard clock.

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.

CVTFCT contains the address of the File Code Table (T:FCT), 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:



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.

CVTRDO 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.

CVTCR 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.

CVTKEY is set to zero at system generation time and is used by the 'Read an Unsollicited Key-in' monitor request to build the chain of requests from user programs for reading the unsollicited key-in.

CVTLAB contains the address of a dispatcher subroutine which manages the scheduled label queue in the Save Area.

CVTDYE, CVTBKE, CVTCRE and CVTSAE 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 DRTM 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 D: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:XNAM routine (search PCT).

CVTSSY + 4 contains the address of the BTIMER 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 (R:POIN)

CVTSSY + 10 contains the address the release timer block routine (F:BLK)

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.

CVTCIF 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.

CVTTSC 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
CVTEFT } 2 words used by EDFM

CVTWAT contains the entry point in the Wait routine

CVTSET contains the entry point in the Reset routine

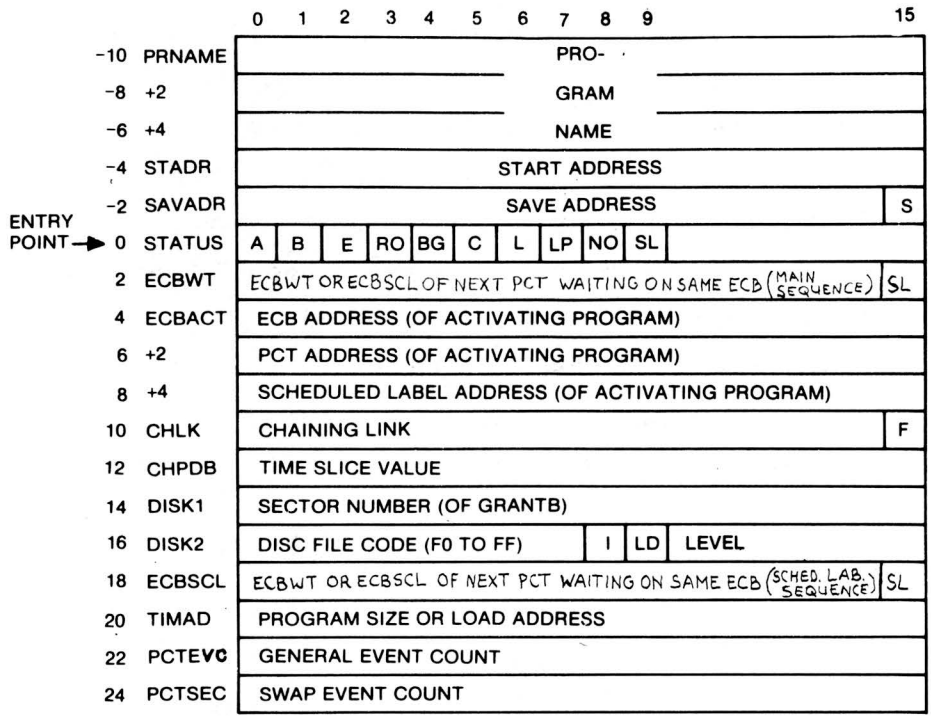
Program Control Table (T:PCT)

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:



where:

PRNAME: is a 3-word block, containing the program name in ASCII left justified and filled with blanks if less than 6 characters long.

STADR: is the start address of the program.

SAVADR: Address of the save area of the program:

If S=0, it is the address of the main program save area.

S=1, it is the address of the scheduled label save area.

STATUS: gives the status of the program. This word is the entry point of the PCT block.

If A=0: the program is in active state (set by Activate (M:ACT)).

A=1: the program is in inactive state (set by Exit (M:EXIT), when Event Count is zero).

B=0: this PCT entry is free for allocation to a program.

B=1: this PCT entry has already been allocated to a program.

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 (PCTEVC) has become zero again (see bit E)

ECBWT: ECBWT or ECBSCL (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).

CHPDB: The left character of this word contains the time slice value for this program. It is set by the control command SW (Declare a Swappable 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 GRANTB 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 swappable 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.

PCTSEC: 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

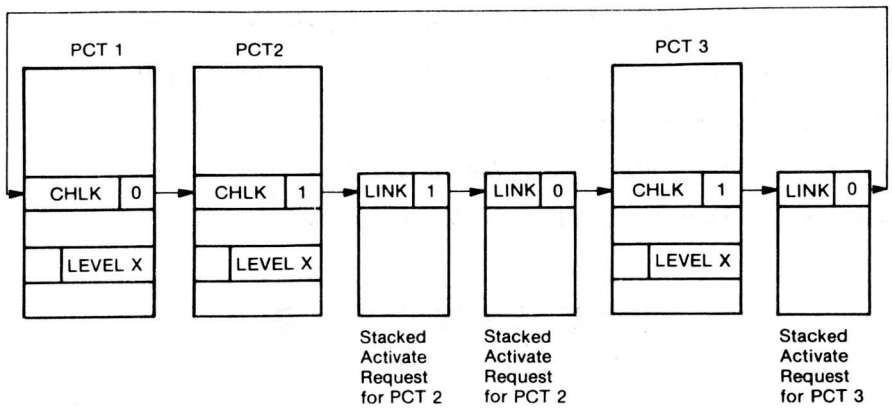
Generally speaking, these words are incremented whenever an action such as an I/O operation, a scheduled label, execution of a non-resident monitor request, is requested by the user and decremented at completion of the action.
For the event counts PCTEVC and PCTSEC, these are the following actions:

- PCTEVC: - incremented by IORM
  decremented by ENDIO
  - incremented by I:LKM if scheduled label requested
  decremented by M:LAB (M:DISP) when scheduling occurs
  - incremented by I:LKM 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:LKM 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, F = 0, it indicates a link to another PCT. If F = 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 hav the following layout:
  - Level 49 stacked entry:

LINK	F	
ADDRESS OF CALLED ROUTINE		(A3)
PCT ADDRESS OF ACTIVATING PROG.		(A5)
SCHEDULED LABEL ADDRESS		(A6)
PROGRAM NAME ADDRESS		(A7)
ECB ADDRESS		(A8)

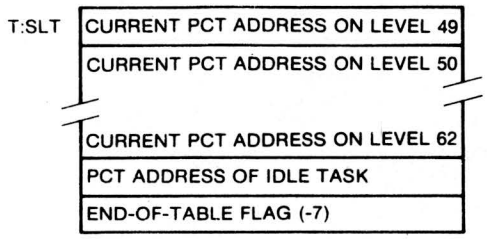
These blocks are created by the Activate monitor request handler (M:ACT) and erased by the Exit monitor request handler (M:EXIT).

Software Level Table (T:SLT)

This table, for which the entry points and the format are defined in the monitor module T:SYS, is used by the dispatcher to find the current PCT for a given level.

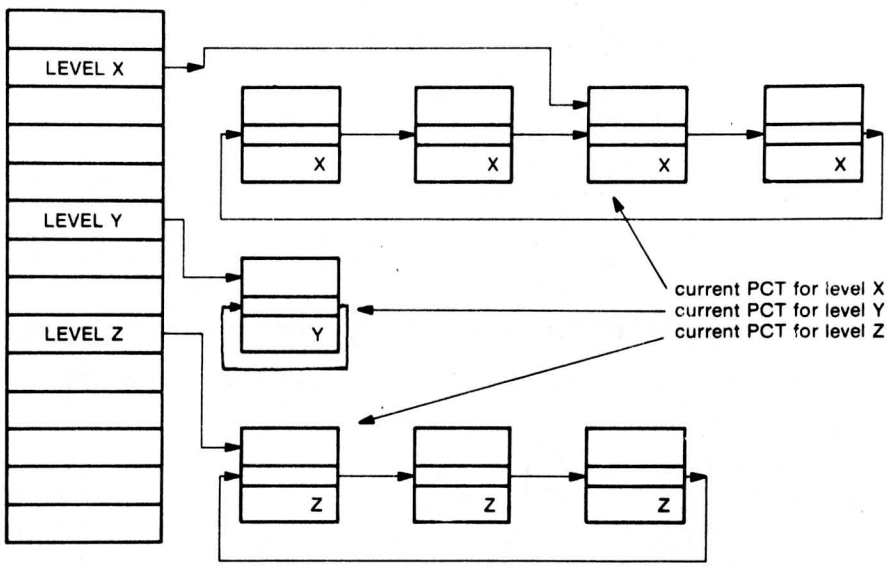
The priority level of a Software Level Program is specified by the position of its PCT-address word in the Software Level Table. The SLT contains 15 entries:

- one for each of the levels 49 through 62
- one for level 63 (level for the Idle Task).



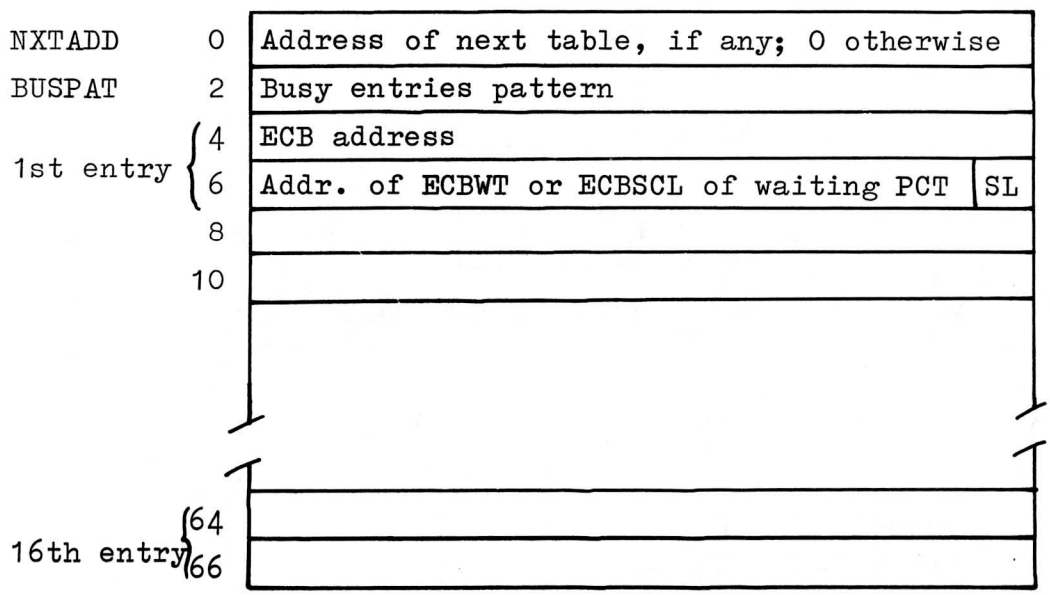
Example of PCT-SLT Organization

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:SWTC).



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:RSEV modules.
The entry point T:EVT must correspond to the word NXTADD of the first table:



NXTADD : Address of the word NXTADD of the next event table, if there is any; otherwise, this word contains 0.

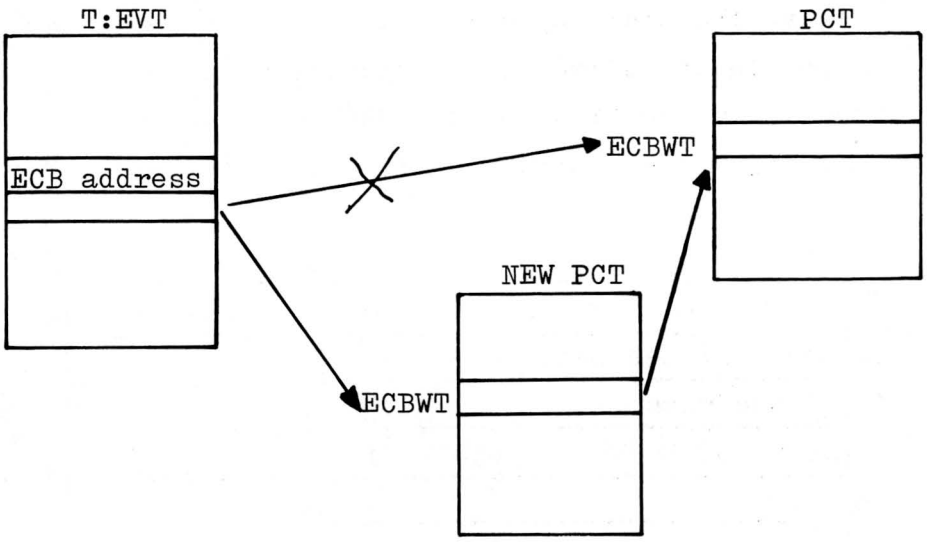
BUSPAT : Pattern of the busy entries in the table, each bit corresponding to one entry in the table (bit 0 = 16th entry, bit 15 = 1st entry). A set bit indicates that the corresponding entry is free.

Each entry is 2 words long:

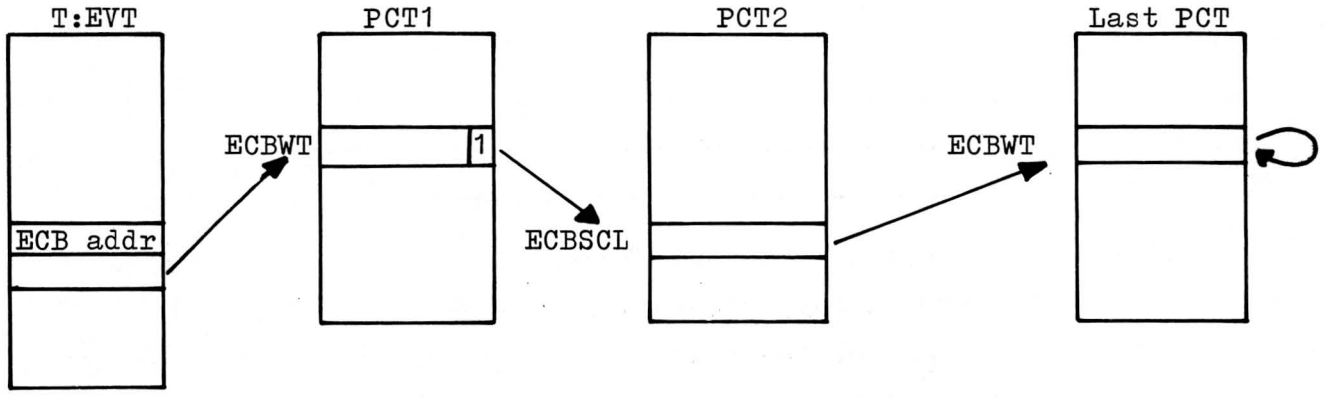
- the first word contains the address of the ECB on which the program is waiting
- the contents of the second word depends on the sequence type of the

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, SL, 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 chain:



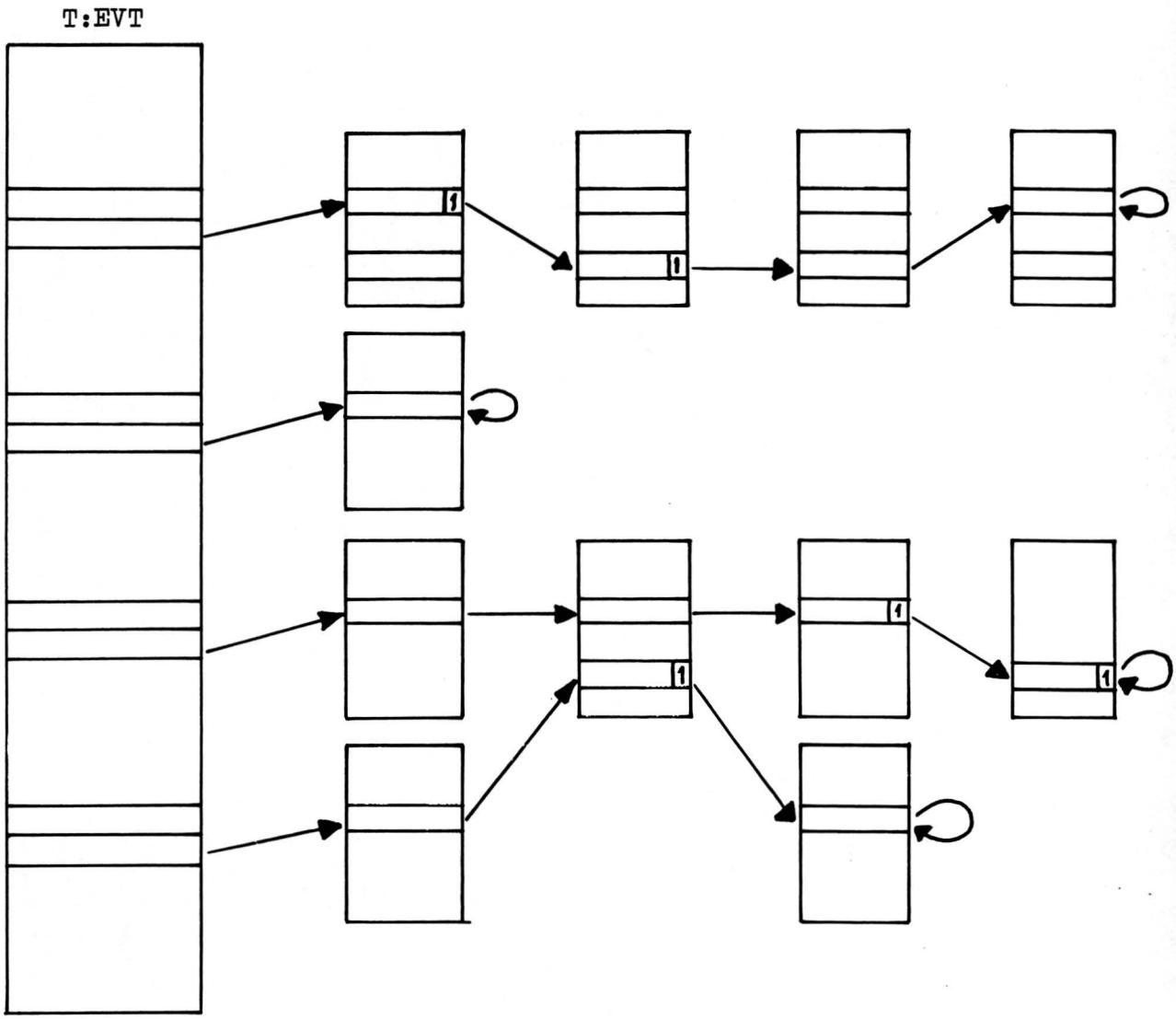
All PCTs waiting on the same ECB are chained as follows:



Event Table Overflow: when no more free entry is left in the Event Table or Pool of Event Tables, a branch is made to SYSAB, with register A2 = 6.

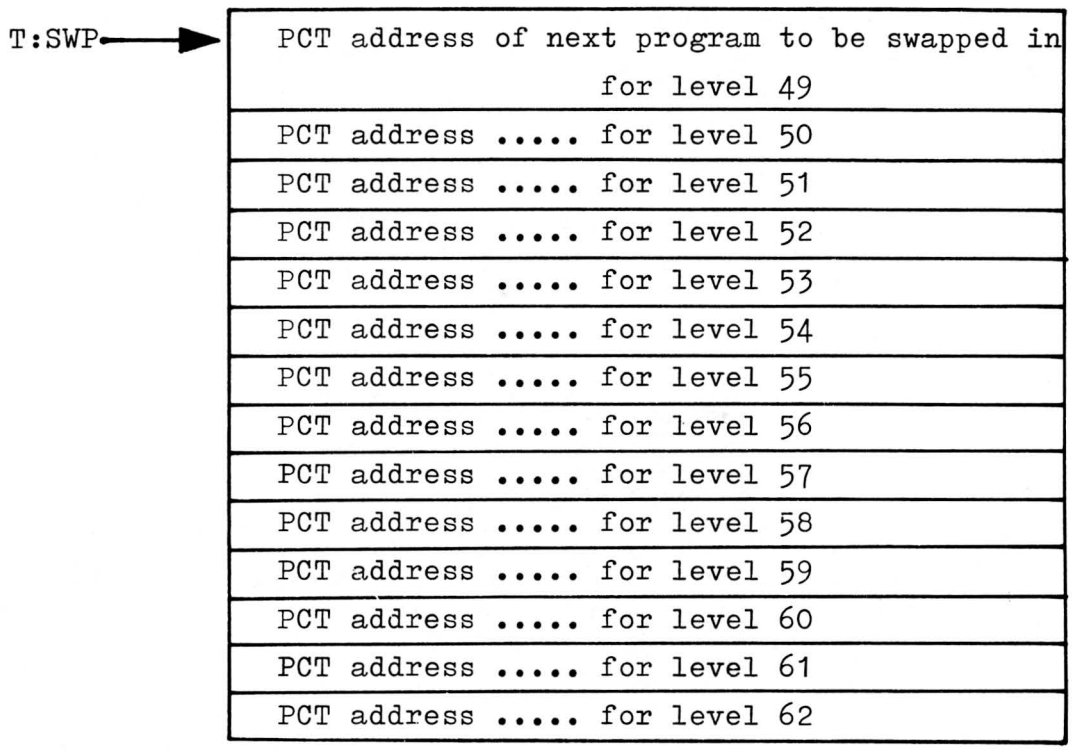
Note: Due to the response time of the requests Wait for an Event and Set Event, it is not recommended to extend the pool of event tables excessively.

Example of T:EVT - PCT organization:



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.



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 ASR 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 DWT has the following layout:

ENTRY POINT →			
		ENTRY LENGTH	NUMBER OF ENTRIES
DWTDN	0	DEVICE NAME	*
DWTDA	2	DEVICE ADDRESS	*
DWTBLG	4	BEST RECORD LENGTH	*
DWTDRV	6	DRIVER ADDRESS	*
DWTSTS	8	SOFTWARE STATUS	
DWTECB	10	ECB ADDRESS	*
DWTBUF	12	CHARACTER OR BUFFER ADDRESS AT INITIALIZATION	
DWTRLG	14	REQUESTED LENGTH	
DWTELG	16	EFFECTIVE LENGTH	
DWTORD	18	ORDER	
DWTRY	20	RETRY BIT (BASIC ORDER)	
DWTTAB	22	WORD TO BE OUTPUT (I/O BUS)/TABULATION ADDR./DCT ADDRESS	
DWTCSM	24	CHECKSUM FOR OBJECT OUTPUT/CONTROL CHAR. FOR LINE PRINTER	
DWTCTL	26	CHAR. INDICATION FOR OBJECT WRITE/CONTR. CHAR. FOR LINE PRINTER	
DWTA5	28	SAVED A5 (PCT ADDRESS OF REQUESTING PROGRAM)	
DWTA6	30	SAVED A6: SCHEDULED LABEL ADDRESS (0, IF NONE)	
DWTC:N	32	CONTROLLER STATUS ADDRESS/C:NXX OR DCTHD ADDRESS	
DWTATT	34	ATTACH LOCATION (PCT ADDRESS OF ATTACHED PROGRAM)	*
DWTSST	36	SST SEQUENCE ADDRESS	*

(*: 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: 255. magnetic tape: 4094. 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 11)

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.

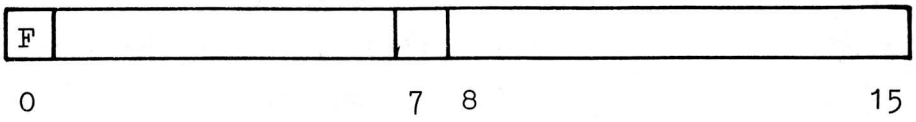
WORD 26 specifies, in case of 4 x 4 object input for devices on the I/O bus, whether the right of left character must be entered. In case of line printer output it is used to save the control code (first two characters of a buffer).

WORD 28 is used to save the program PCT 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 DCTHD 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 wether 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:



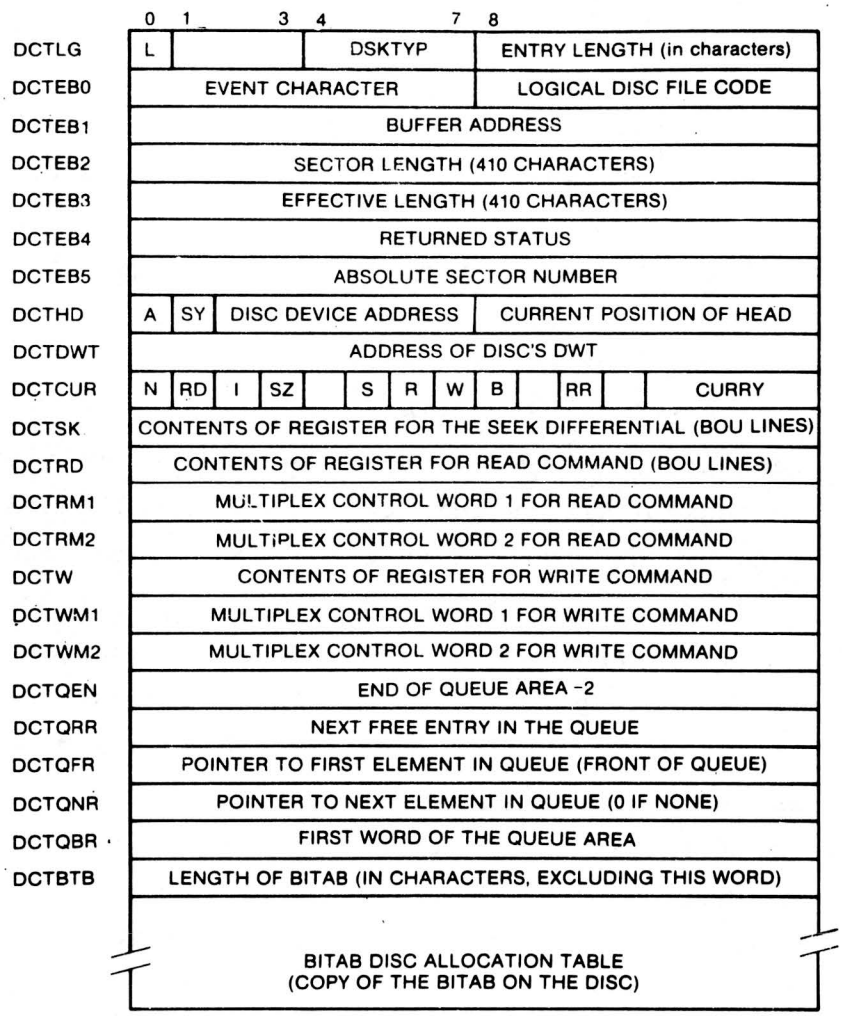
F = 1 if the control unit is free; 0, if busy.
If busy, bits 7 to 15 are used as follows:
bit 7 = 1, set by M:RETR to indicate that the device is waiting for an RY or RD operator message. If RY is typed it is reset to zero; if RD is typed, it remains 1, to indicate to M:RETR not to output the RY flag 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 36 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.

Disc Control Table (T:DCT)

This table complements the Device Table (DWT) for the disc drivers and facilitates communication between Data Management, disc driver and monitor.
At system generation time, one entry must be created for each disc in the configuration:



DCTLG: L=1: last entry in DCT
L=0: current entry
DSKTYP indicates the type of disc:
0: CDD disc model 1

6: CDD disc model 2

The right character contains the length of the entry, in characters.

DCTEB0 to DCTEB5 is the ECB used only by the Data Management routine to make a request to the physical disc driver:

DCTEB0: bits 0 to 7: event character
bits 8 to 15: disc file code (from /F0 to /FF), identifying the disc on which the file is stored.

DCTEB1: address of the buffer (one sector long) which is used by the physical disc driver to read or write a sector. If sequential access is used, this is the blocking buffer address and if random access is used, this is the user buffer address.

DCTEB2: length of the buffer used to read or write a sector.

DCTEB3: effective length, returned by the physical disc driver.

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: the absolute address, on disc /FX, of the sector which is to be read or written.

DCTHD: A=1: device free
A=0: device active, a request has been received
DEVICE ADDRESS is the physical device address
SY=1, when the system disc is concerned.

POSITION OF HEAD is a value from 0 to 202

DCTCUR: gives a list of the operations which must be performed to satisfy the request. It is initialized either when the request is processed or in case of error recovery.
N=1: the disc is not operable. Set to 0 at system generation time, set by the INIMON module when a disc is not mounted.
RD=1: disc is ready, a new pack has been mounted but not yet initialized.
When one of the bits N or RD is set, the physical disc driver will refuse the request, even if it came from a system program. Bit 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 = 1, the program DISKINIT (belonging to D:USV1) will be called to reinitialize the disc by:

- suppressing the file codes assigned to the files on the old disc
- resetting N and RD to 0
- resetting DCTHD
- reloading DCTBTB.

The following bits indicate the operations which must be performed:
I=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 zero when SST instruction is executed.

SZ=1: A seek to zero must be performed. Set only after detection of a seek error, either by hardware status or by software identifier check.
S=1: A differential (seek must be performed. The contents of register A3 will be given in word DCTSK.
R=1: A read command must be sent to the disc.
W=1: A write command must be sent to the disc.

B=1: The head position recorded in DCTHD does not correspond to the actual position of the head, i.e. the next operation for this disc will be a seek to zero.
RR=1: A request is being processed on the other part of the X1215 drive; do not initialize an operation on this part of the drive.
CURRY contains the remaining retry count for the operation. It is initialized by the driver when the request is processed and decremented by one each time the system tries to recover an error. When it reaches zero, no more operations will be allowed for the current request and a status will be returned to the ECB of the requesting program. (Up to 5 retries may be made).
In the following cases SZ, S, I and W are reset to zero:
- After an SST command without an error indication in the returned hardware status, the driver resets the leftmost of these bits to zero and attempts to execute the next operation, as specified by the first following bit being 1.
- If during the previous command, a hardware error was detected (seek or throughput error, for example), branch to the error recovery routine at level 48 and do not reset any bit.

Processing an I/O request

When the disc driver receives a Read or Write request various bits have to be set as follows in word DCTUR:

- Write: W=1, and -if a seek must be executed:
  S=R=1
  SZ=0

  -if no seek is required:
  S=R=SZ=0
- Read: R=1, SZ=RD=0, and if a seek is required S=1,
  if not S=0.

Processing a recoverable error

- If the erroneous operation is a Read or Write command, try again (retry is possible up to 5 times).
- If it was the wrong sector identifer, initialize the two words to execute the
  seek to zero
  seek differential
  read (identifier)

  i.e. set SZ=S=R=1; W remains unchanged.

DCTQEN: is the address of the last entry reserved for the queue on the disc.

DCTQRR: is the address of the next free entry in the queue (end of queue+1).

DCTQFR: is the address of the first element of the queue (front of queue).

DCTBTB: this part of the DCT is used in connection with the dynamic allocation of granules to temporary user files. The length depends on the model of the disc to which this DCT is assigned.
When the system is loaded, the monitor initialization program INIMON will load the BITAB table from the disc.

If BITAB is longer than the last section of the DCT, it will be truncated in memory. If it is shorter, the remaining words in the DCT will be filled with zeroes.
When the I/O management routine has allocated a granule, the corresponding bits of DCTBTB are reset to zero.
When a temporary file is deleted by the Command Language program or the LKM processor, all the bits corresponding to the granules of this file will be set to 1 and reset to zero after a file code has been assigned to a temporary file.
Whan a temporary file is made permanent by keeping it in a library, all the corresponding bits in BITAB are set to one.
To delete a permanent file, all corresponding bits in BITAB must be set to 1, but <u>not</u> the bits in DCTBTB.

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	X	LENGTH							ORDER					
LFTEAD	USER ECB ADDRESS													
LFTREC	USER RECORD AREA ADDRESS													
LFTLGT	USER REQUESTED LENGTH													
LFTPCT	A5 (PCT ADDRESS)													
LFTLAB	A6 (SCHEDULED LABEL ADDRESS)													
LFTMD1	A	P	S	O	U	C	T	R	SE	FM	E	W	CO	
LFTMD2	ASCNT													
LFTDCT	D:CT ADDRESS													
LFTBOT	ABSOLUTE ADDRESS OF SECTOR GRANTB OF THIS FILE													
LFTSRC	RELATIVE NUMBER (0 ton) OF CURRENT SECTOR													
LFTSAC	ABSOLUTE ADDRESS OF CURRENT SECTOR													
LFTBAD	BLOCKING BUFFER ADDRESS								NBR. OF SECTORS OF DIR. ACC. FILE					
LFTBDS	DISPLACEMENT OF NEXT RECORD IN BLOCKING BUFFER													
LFTBUF	CURRENT BUFFER ADDRESS (FOR CURRENT OPERATION)													
LFTSEC	CURRENT SECTOR TO BE READ OR WRITTEN													
LFTORC	CURRENT ORDER TO BE PERFORMED													
LFTSTC	CURRENT STATUS													
LFTSVD														
LFTSVS														
LFTSLU														
LFTSLB														
LFTSLC														
LFTSLT														
LFTSLR														
LFTLK1														
LFTLK2														
LFTATT														

LFTORD: 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.

LFTEAD: contains the address of the user ECB.

LFTREC: is the address of the user record area, where the Data Management routine must read or write the logical record.

LFTLGT: 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.

LFTORD: is used to save the I/O order given in the monitor request.

LFTPCT: contains the address of PCT of requested program.

LFTLAB: contains the scheduled label address. If no scheduled label is used, it contains 0.

LFTMD1 and LFTMD2 are two mode words defining the status of the LFT:

LFTMD1:

A=1: LFT inactive, free for processing a request.
=0: LFT busy. A user request has already been recorded and is not yet terminated.

P=1: file is write-protected.
S=1: source file.
O=1: object file.
C=1: load module.

} These bits are always 0 under DRTM

U=1: undefined file; contains user data.

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.

FM=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.

CO=1: consecutive file.

0: non-consecutive file.

LFTMD2:
Bits 0 to 3 contain the number of file codes assigned to this LFT. At system generation time, it is initialized with the value 0. It is incremented each time a file code is assigned to this LFT and decremented when the assignment of a file code changes. Thus, when ASCNT becomes zero again, LFT 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.

LFTDCT: contains the address of the D:CT of the disc on which this file is stored.

LFTBOT: 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.

LFTSRC: is the relative sector number within the file.

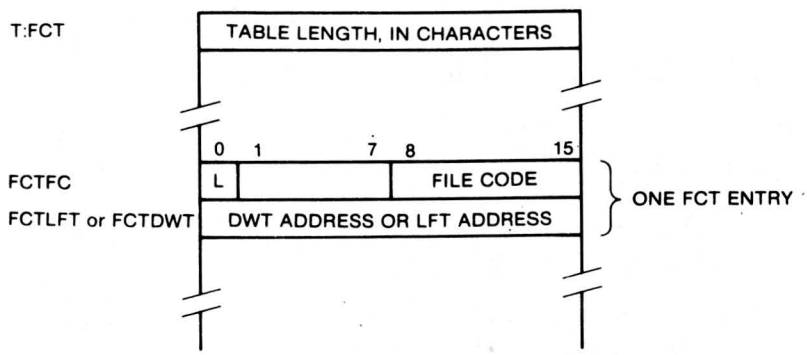
LFTBAD: contains, for sequential files, the current blocking buffer address. For random access files, it contains the number of sectors in the file.

The words LFTBUF to LFTLK2 are used as work area by the logical I/O routines for performing their operations. LFTBUF, LFTSEC, LFTORC and LFTSRC 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.

LFTATT: 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:FCT)

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 /FX. 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:

FCTFC: 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.

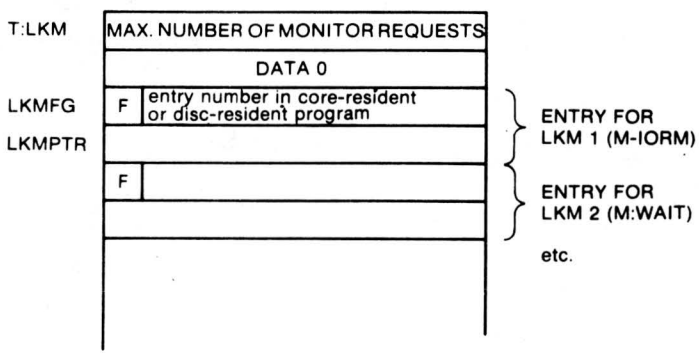
If L=1, the second word of the entry contains the address of the file description table LFT and is called FCTLFT.

If L=0, the second word contains the address of the Device Work Table (DWT) and is called LFTDWT.

- 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:LKM)

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:LKM 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.



There must be as many 2-word entries as the number in the location T:LKM specifies (not including the first 2 words).

The words in the entries have the following meaning:

-LKMFG: F=0: the monitor request is processed by a possibly disc-resident program running at level 49 to 62.
The contents of LKMFG 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.

LKMPTR: 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, LKMPTR 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, LKMPTR 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:

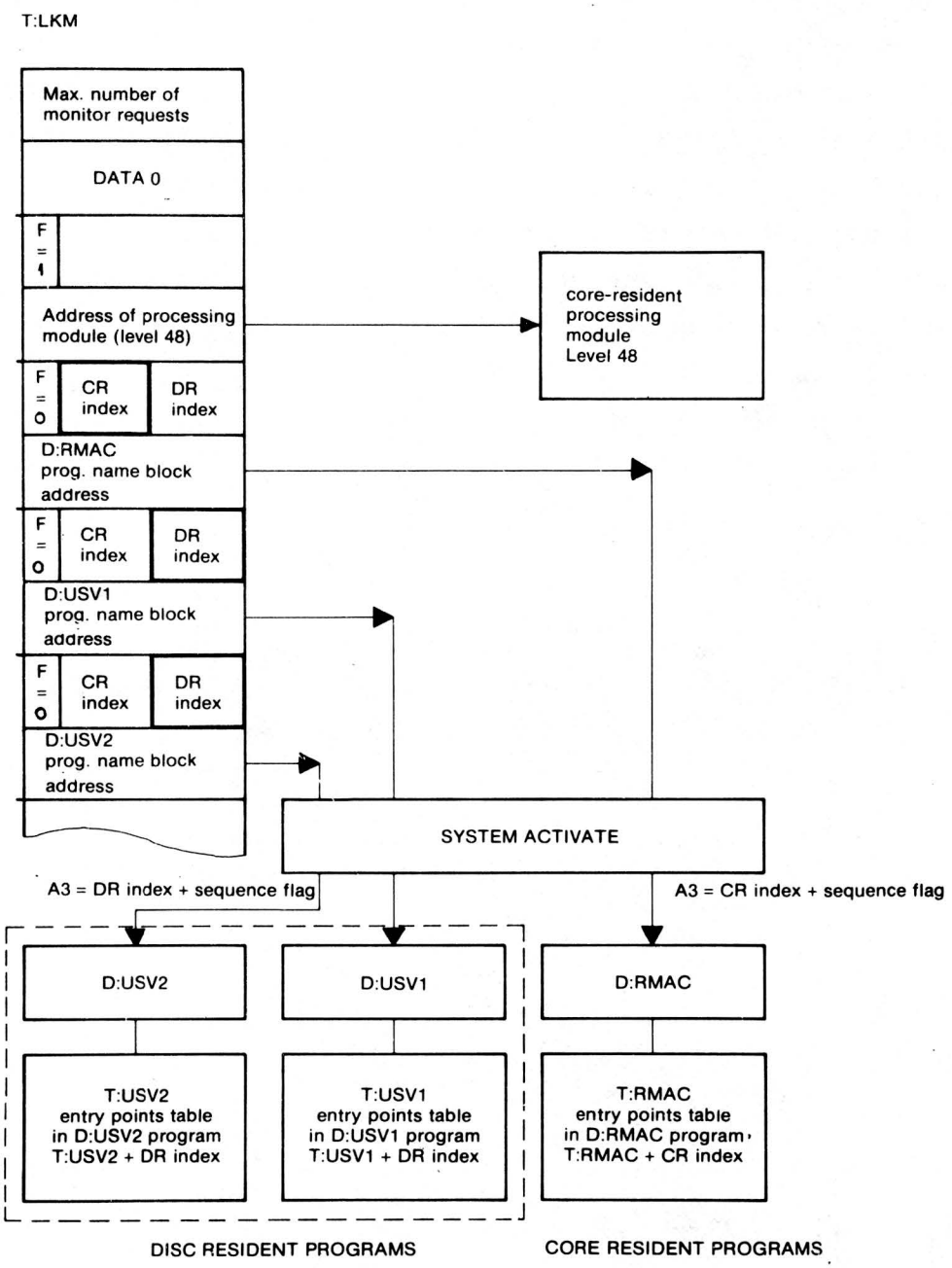
- I/O
- Wait for an Event
- Activate
- Exit
- Get Buffer
- Release Buffer
- Set Event
- Switch inside a Level

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:

T:RMAC		
	0	DATA M:DFM (Data Management)
	2	DATA M:NDLG (Data Management)
	4	DATA D:CNTM (Connect Timer)
	6	DATA D:DNTM (Disconnect Timer)
	8	DATA D:ATDV (Attach Device)
	10	DATA D:DTDV (Detach Device)
	12	DATA D:GTIM (Get Time)
	14	DATA D:CNLV (Connect Level)
	16	DATA D:DNLV (Disconnect Level)
	18	DATA D:WGT (Wait for given Time
	20	DATA ASGPRO (Assign File Code)
	22	DATA DELPRO (Delete File Code)
	24	DATA KEYPRO (Read Key-In)
	26	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
Max. number of monitor requests
DATA 0
F = 1
Address of processing module (level 48)
F = 0
CR index
DR index
D:RMAC prog. name block address
F = 0
CR index
DR index
D:USV1 prog. name block address
F = 0
CR index
DR index
D:USV2 prog. name block address
core-resident processing module Level 48
SYSTEM ACTIVATE
A3 = DR index + sequence flag
A3 = CR index + sequence flag
D:USV2
D:USV1
D:RMAC
T:USV2 entry points table in D:USV2 program T:USV2 + DR index
T:USV1 entry points table in D:USV1 program T:USV1 + DR index
T:RMAC entry points table in D:RMAC program T:RMAC + CR index
DISC RESIDENT PROGRAMS
CORE RESIDENT PROGRAMS

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:MONT	MONTH (ASCII VALUE)
W:YEAR	YEAR (ASCII VALUE)
H:TIME	HOURS (BINARY VALUE -24)
M:TIME	MINUTES (BINARY VALUE -60)
S:TIME	SECONDS (BINARY VALUE -60)
B:TIME	TENTHS OF A SECOND (BINARY VALUE -10)
A:TIME	FIFTIETH OF A SECOND (BINARY VALUE -5)
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 abs. time HH MM SS
	Address of last block connected to abs. 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:DCK2 and reset by it.
If one of the values in the table ≠ 0, the chain connected to that timer must be scanned:

V:FLAG	If not zero, scan chain for H:TIME
	M:TIME
	S:TIME
	A:TIME
	B:TIME
	C:TIME

V:RSET

This block contains the values necessary to reset the real time block values.

V:RSET	-24
	-60
	-60
	-10
	-5
	Negative non-standard clock pulse

Blocks built by Connect Program to Timer request (D:CNTM module)

Standard Block

Format before first activation of the connected program

CHAINING LINK
-NC
+PR
PROGRAM PCT ADDRESS

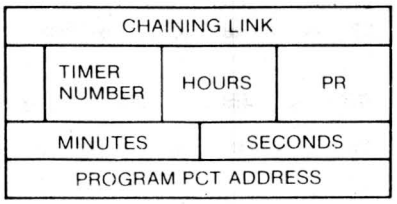
Format after the first activation

CHAINING LINK
+PR
-PR
PROGRAM PCT ADDRESS

NC: number of timer cycles before the first program activation

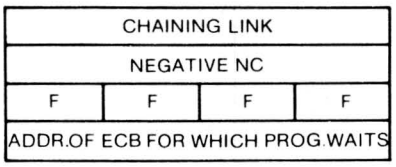
PR: number of timer cycles between two activations of the same program.

Block for programs connected to absolute time



T.N. is the timer number.

Block for programs waiting for a given time (D:WGT module)



Timer Numbering

- With declaration of the 20 msec. clock:
  - 0 - clock timer (20 msecs)
  - 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.

3 Input/Output

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 DWT+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 (1=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 (ENDIO)
- Service routines (COMIO and M:RETR)

IORM

When the user has given an I/O monitor request (LKM 1), this module will receive control, via the I:LKM module and T:LKM 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+32). 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)

  A3: Hardware order (contents of register with which CIO Start will be sent).

Note: The I/O processor control parameters must have been reinitialized.

ENDIO

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, ENDIO 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 ENDIO 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

COMIO

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 these sequences the instructions INH - MSR - ABL must be given in the order specified:

- CIO Start: A6: DWT address
  A2: hardware order
  A3: return address
  INH
  STR A1,A15
  STR A8,A15
  ABL S:TIO
  Return: - condition register set
  - inhibit mode.
- CIO Stop: A6: DWT address
  A3: return address
  INH
  STR A1,A15
  STR A8,A15
  ABL H:LTIO
  Return: - condition register set
  - inhibit mode.
- OTR: A6: DWT address
  A3: Return address
  A1: word to be output
  INH
  STR A1,A15
  STR A8,A15
  ABL O:TRIO
  Return: - condition register set
  - inhibit mode.

- INR: A6: DWT address
A3: return address
INH
STR A1,A15
STR A8,A15
ABL I:NRIO
Return: - condition register set
- inhibit mode
- word or character to be input: in A1.

- SST: A6: DWT address
A3: return address
INH
STR A1,A15
STR A8,A15
ABL S:SST
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
INH
STR A1,A15
STR A8,A15
ABL M:TEX

The requesting program is put in Wait, if this is necessary, as follows:

A7: User I/O order
A8: 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)
A6: 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 operator:
(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 <u>not</u> reset bit 7 of the control unit status, but EXIT
- for RY messages: reset bit 7 of the control unit status, then EXIT.

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

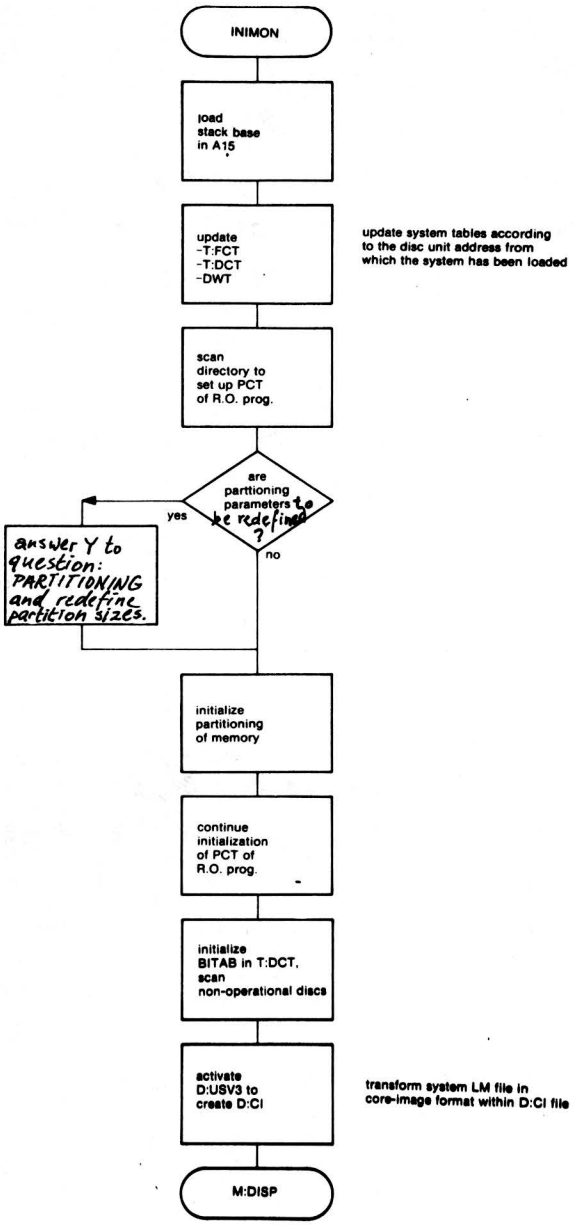
INIMON initializes:

- the disc tables: T:FCT, DWT, T:DCT
- memory partitioning: CVT
- the program control tables: T:PCT.

Besides, it uses one buffer (CATBUF, 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.


INIMON
load
stack base
in A15
update
-T:FCT
-T:DCT
-DWT
update system tables according to the disc unit address from which the system has been loaded
scan
directory to
set up PCT
of R.O. prog.
are
parttioning
parameters to
be redefined
?
yes
no
answer Y to
question:
PARTITIONING
and redefine
partition sizes.
initialize
partitioning
of memory
continue
initialization
of PCT of
R.O. prog.
initialize
BITAB in T:DCT,
scan
non-operational discs
activate
D:USV3 to
create D:CI
transform system LM file in core-image format within D:CI file
M:DISP

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 /EF).

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

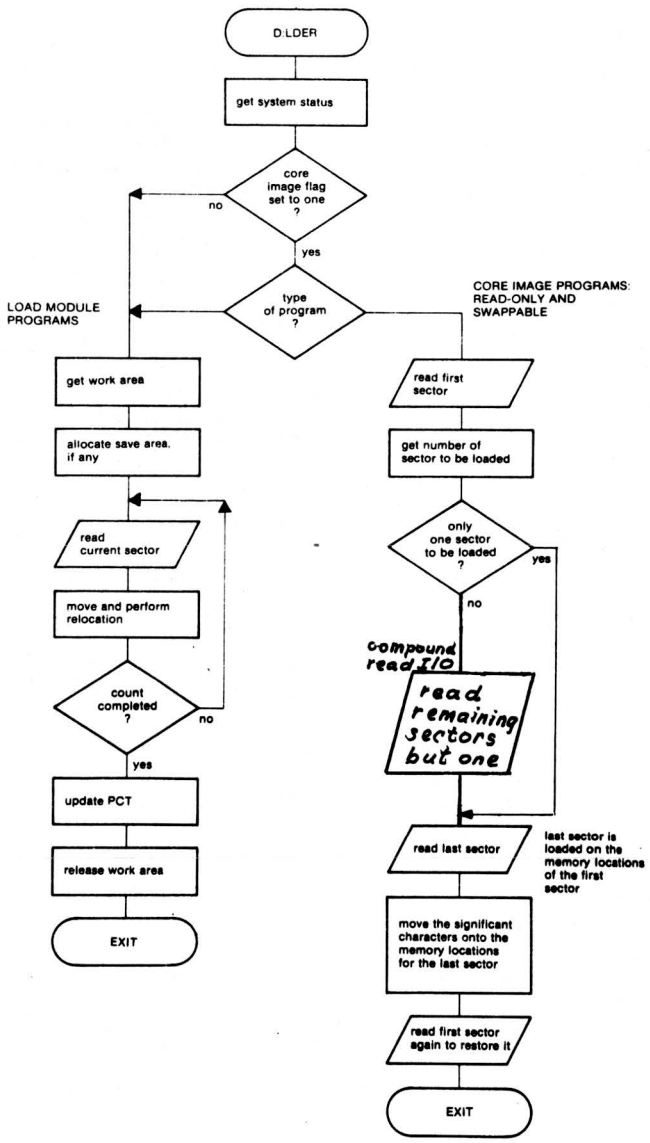
The principal function of the loader is to load a program from disc into memory. The program must be stored on the disc in the standard load module format.
In memory, the program can be stored in any area: Memory Resident, Read Only, Swap or Background Area. The program's PCT must have been initialized 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 PCT entry to which the scheduled label applies. (Irrelevant if A6 = 0).

ABL 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-register, 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:PCT (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:PCT , which points to the first PCT of the PCT Pool;

  P:CUR , which contains the PCT address of the current program;

  P:ROPC , which contains the PCT address of the read only program currently occupying the read only area;

  P:SWAP , which contains the PCT 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 PCT address of the next swappable program to be initialized 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:PCTB , which contains the PCT address of a background program waiting to be loaded.

  F: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 Level: only if the request is fective, i.e. if another PCT waiting for CPU time is found waiting on that level.

Input/Output Files

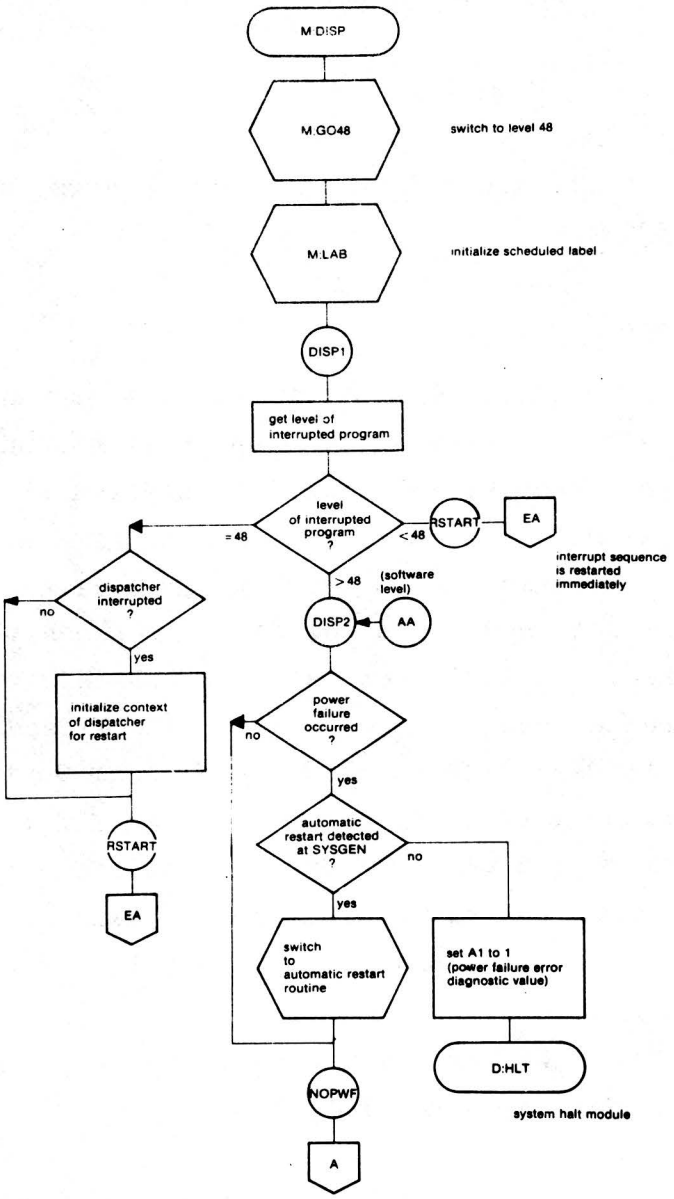
None.

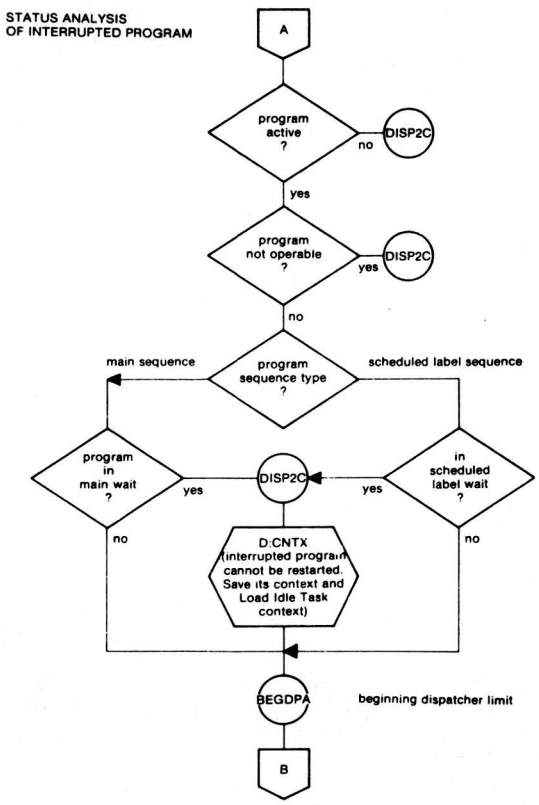
Memory Layout

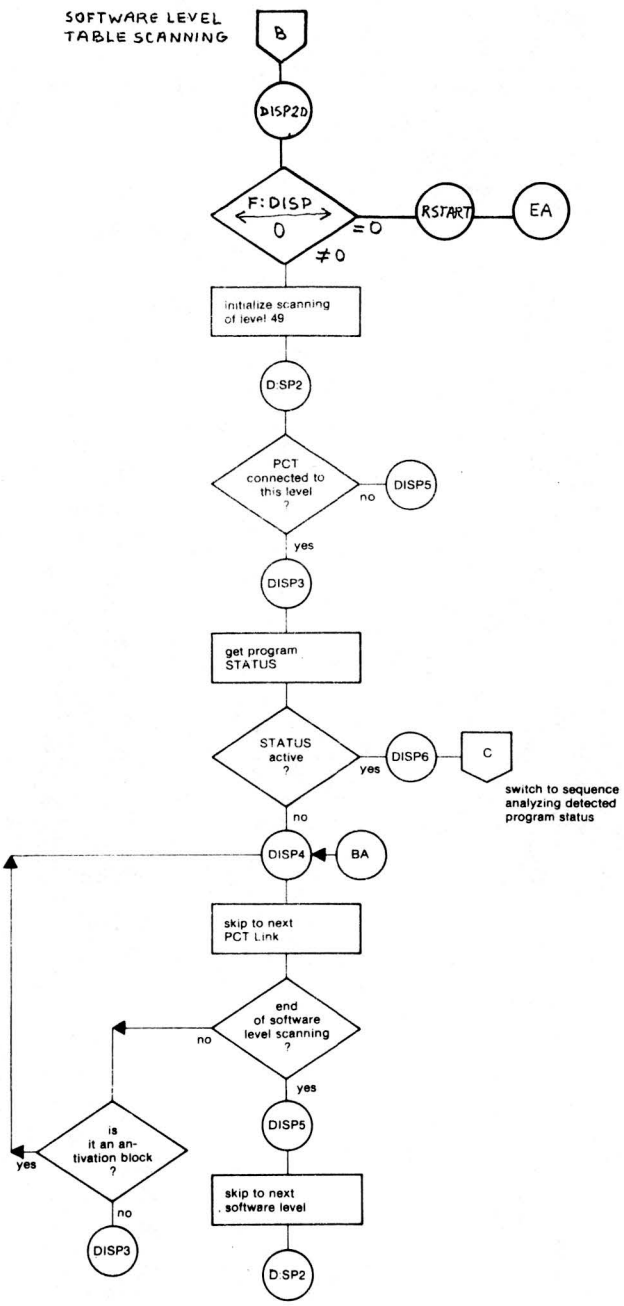
M:DISP is always resident in memory and belongs to the supervisor part of the DRTM.

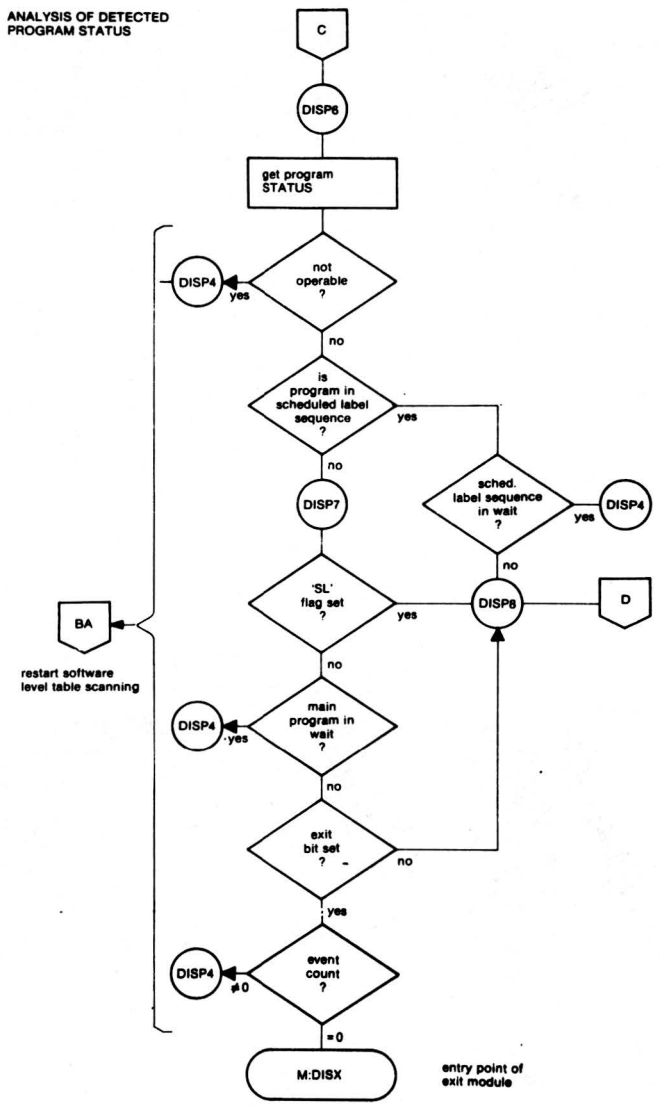
Functional Description

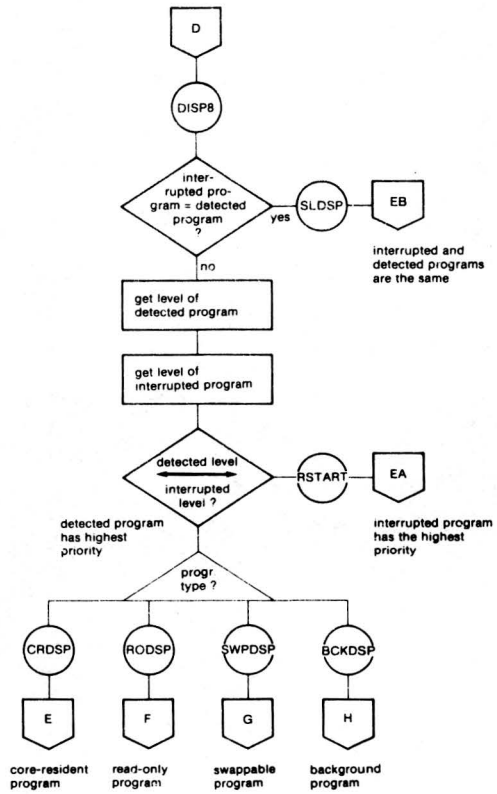
The dispatcher is called by any system routine running at a level equal to or below 48, especially when an event, such as I/O, has occurred and a scheduled label must be started.
Its main function is to determine which program must get CPU time. If the interrupted program was running at a level equal to or below 48, control is returned to it immediately. Then the F: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.


M:DISP
M:GO48
switch to level 48
M:LAB
initialize scheduled label
DISP1
get level of interrupted program
level of interrupted program ?
= 48
< 48
RSTART
EA
interrupt sequence is restarted immediately
> 48
(software level)
dispatcher interrupted ?
no
yes
DISP2
AA
initialize context of dispatcher for restart
power failure occurred ?
no
yes
automatic restart detected at SYSGEN ?
no
yes
RSTART
EA
switch to automatic restart routine
set A1 to 1 (power failure error diagnostic value)
D:HLT
system halt module
NOPWF
A

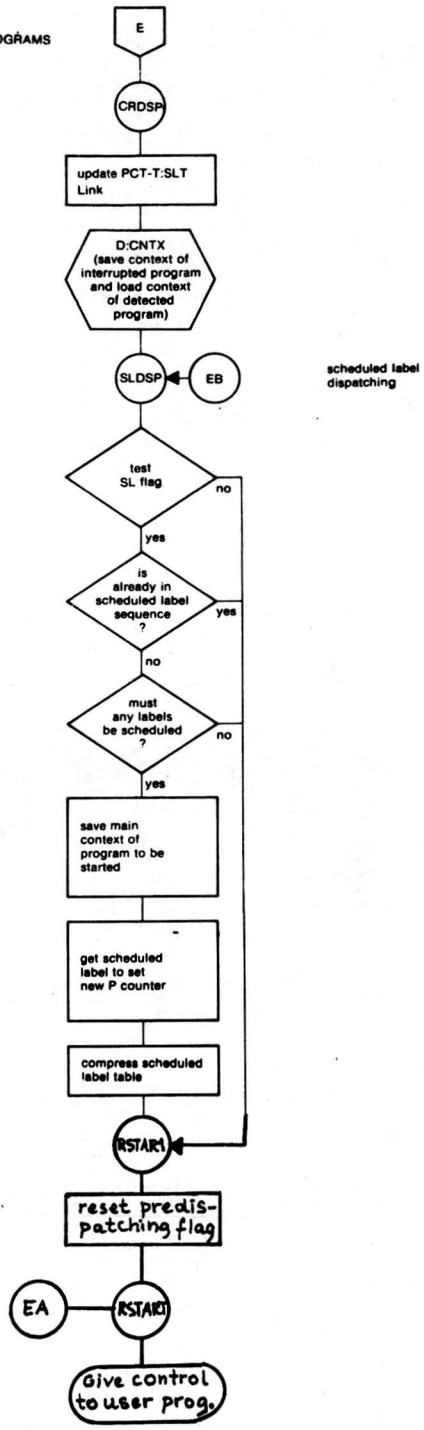

STATUS ANALYSIS
OF INTERRUPTED PROGRAM
A
program active ?
no
DISP2C
yes
program not operable ?
yes
DISP2C
no
main sequence
program sequence type ?
scheduled label sequence
program in main wait ?
yes
DISP2C
yes
in scheduled label wait ?
no
no
D:CNTX (interrupted program cannot be restarted. Save its context and Load Idle Task context)
BEGDPA
beginning dispatcher limit
B

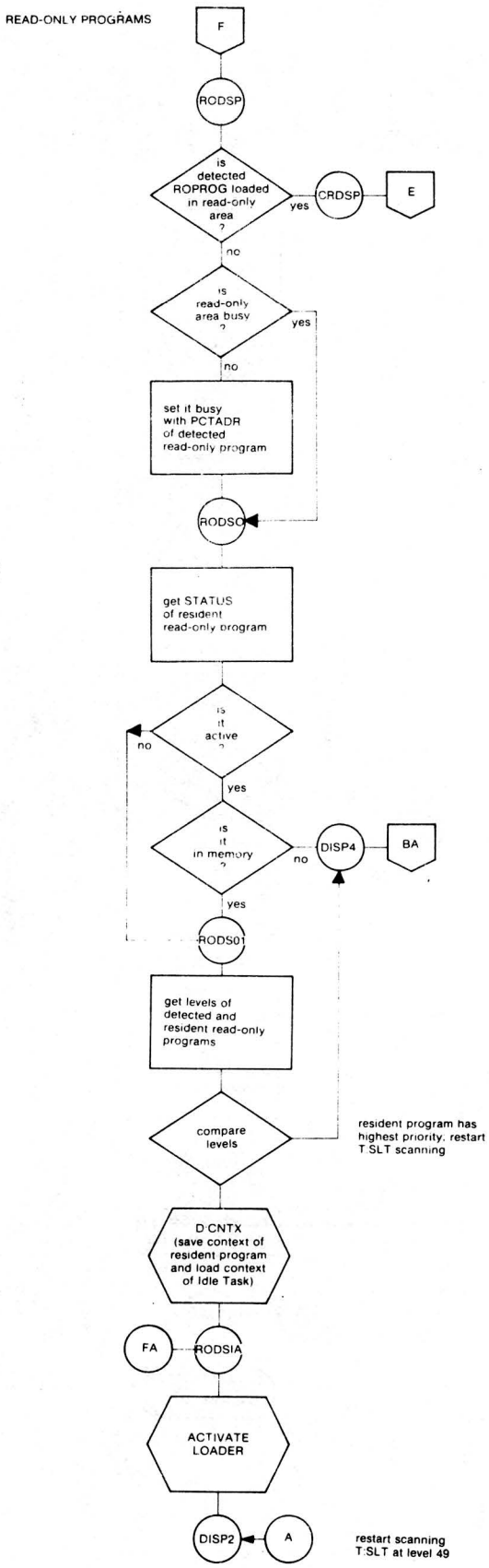

SOFTWARE LEVEL
TABLE SCANNING
B
DISP2D
F:DISP
0
=0
≠0
RSTART
EA
initialize scanning
of level 49
DISP2
PCT
connected to
this level
?
no
DISP5
yes
DISP3
get program
STATUS
STATUS
active
?
yes
DISP6
C
switch to sequence
analyzing detected
program status
no
DISP4
BA
skip to next
PCT Link
end
of software
level scanning
?
no
yes
DISP5
is
it an an-
tivation block
?
yes
no
skip to next
software level
DISP3
DISP2


ANALYSIS OF DETECTED PROGRAM STATUS
C
DISP6
get program STATUS
not operable ?
DISP4
yes
no
is program in scheduled label sequence ?
yes
no
DISP7
sched. label sequence in wait ?
DISP4
yes
no
'SL' flag set ?
yes
DISP6
D
BA
restart software level table scanning
no
main program in wait ?
DISP4
yes
no
exit bit set ?
no
yes
event count ?
DISP4
≠0
=0
M:DISX
entry point of exit module

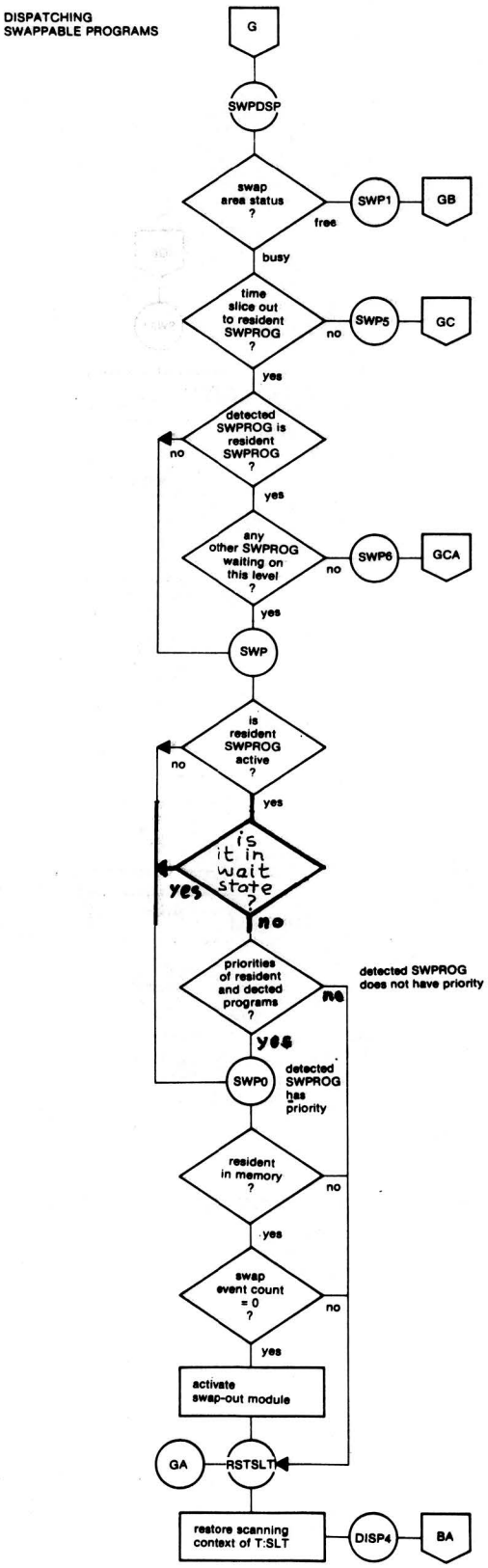

D
DISP8
inter-
rupted pro-
gram = detected
program
?
yes
SLDSP
EB
interrupted and
detected programs
are the same
no
get level of
detected program
get level of
interrupted program
detected level
interrupted
level ?
RSTART
EA
detected program
has highest
priority
interrupted program
has the highest
priority
progr
type ?
CRDSP
RODSP
SWPDSP
BCKDSP
E
F
G
H
core-resident
program
read-only
program
swappable
program
background
program

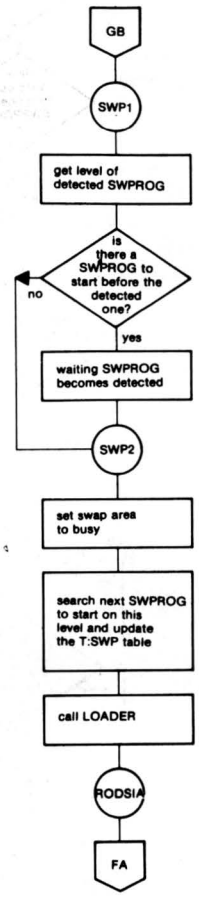
DISPATCHING
CORE-RESIDET PROGRAMS

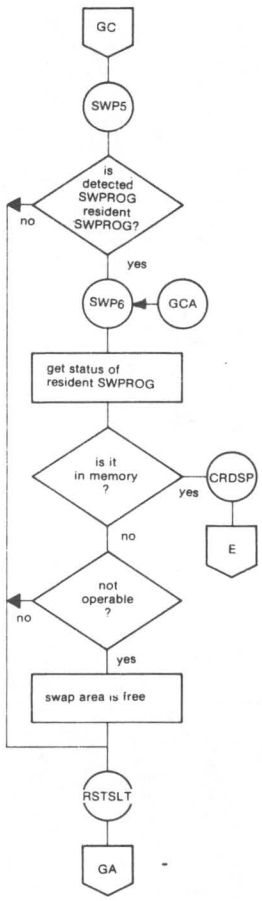



READ-ONLY PROGRAMS
F
RODSP
is detected ROPROG loaded in read-only area ?
yes
CRDSP
E
no
is read-only area busy ?
yes
no
set it busy with PCTADR of detected read-only program
RODSO
get STATUS of resident read-only program
is it active ?
no
yes
is it in memory ?
no
DISP4
BA
yes
RODS01
get levels of detected and resident read-only programs
compare levels
resident program has highest priority, restart T:SLT scanning
D:CNTX (save context of resident program and load context of Idle Task)
FA
RODSIA
ACTIVATE LOADER
DISP2
A
restart scanning T:SLT at level 49

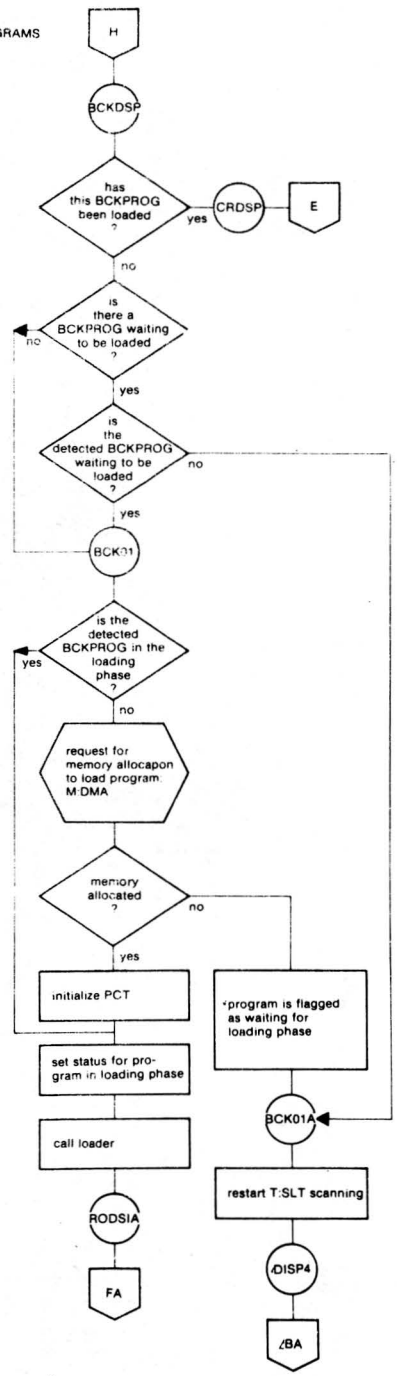
DISPATCHING
SWAPPABLE PROGRAMS

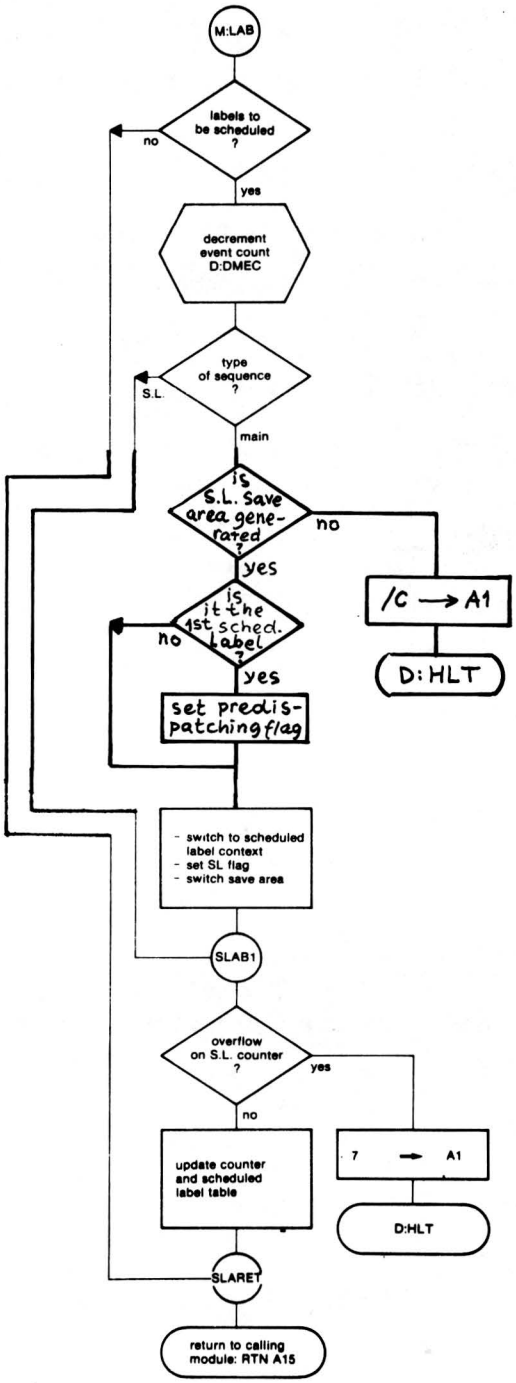



GB
SWP1
get level of detected SWPROG
is there a SWPROG to start before the detected one?
no
yes
waiting SWPROG becomes detected
SWP2
set swap area to busy
search next SWPROG to start on this level and update the T:SWP table
call LOADER
RODSIA
FA


GC
SWP5
is detected SWPROG resident SWPROG?
no
yes
SWP6
GCA
get status of resident SWPROG
is it in memory ?
yes
CRDSP
E
no
not operable ?
no
yes
swap area is free
RSTSLT
GA

DISPATCHING
BACKGROUND PROGRAMS




M:LAB
labels to be scheduled ?
no
yes
decrement event count D:DMEC
type of sequence ?
S.L.
main
is S.L. Save area generated ?
no
yes
/C → A1
D:HLT
is it the 1st sched. Label ?
no
yes
set predispatching flag
- switch to scheduled label context
- set SL flag
- switch save area
SLAB1
overflow on S.L. counter ?
yes
no
update counter and scheduled label table
7 → A1
D:HLT
SLARET
return to calling module: RTN A15

I:PFAR (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:PFAR (from interrupt)
I:ARES (from dispatcher)

Work Areas and Tables

This module uses A15 Stack and CVT 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

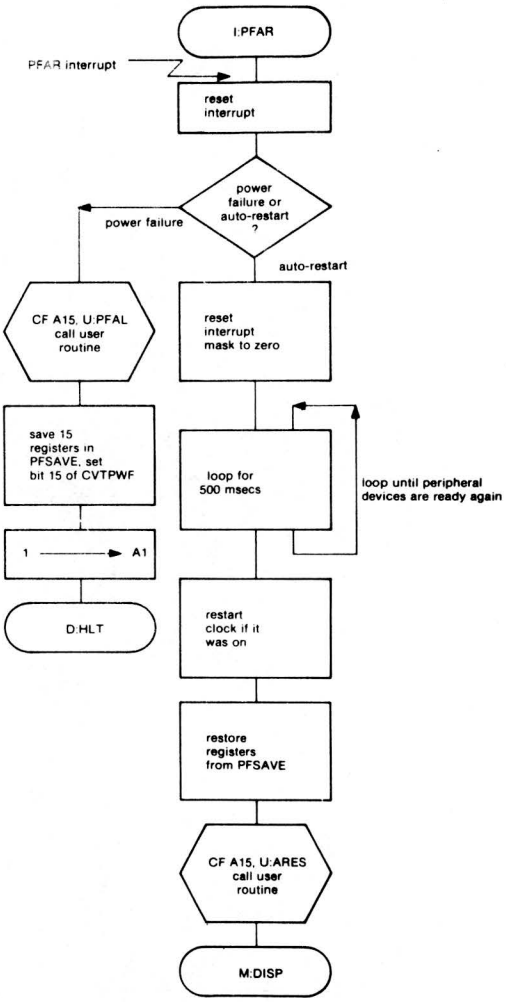
The three optional routines mentioned in the flowchart for this module:
U:PFAL
U:ARES
U:RST
must be provided by the user. If they are not used, they must be simulated with an RTN A15 instruction.

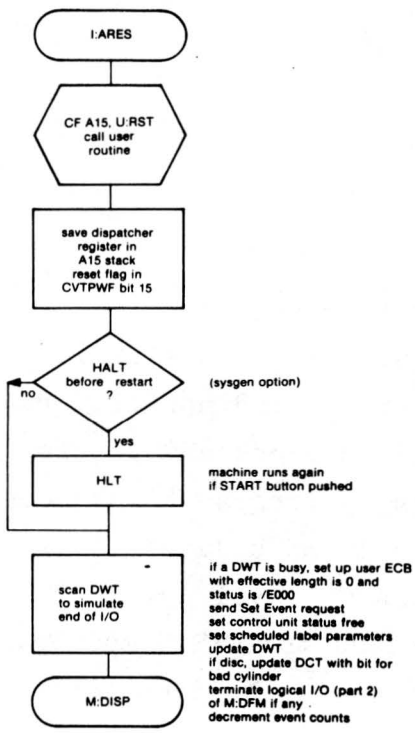
- When a power failure interrupt has been recognized and the interrupt has been reset, the system calls the user power failure routine, before saving the 15 registers:
  CF A15, U:PFAL
- When the automatic restart interrupt has been recognized, the system restores the 15 registers and the user automatic restart interrupt routine:
  CF A15, U:ARES
- The routines U:PFAL 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:

CF A15, U:RST

This restart routine is called before the system simulates the end of all pending I/O operations. These will al receive the status /E000 (power failure occurred), so that they must be requested again, regardless of the device.


I:PFAR
PFAR interrupt
reset interrupt
power failure or auto-restart ?
power failure
auto-restart
CF A15, U:PFAL call user routine
reset interrupt mask to zero
save 15 registers in PFSAVE, set bit 15 of CVTPWF
loop for 500 msecs
loop until peripheral devices are ready again
1 → A1
D:HLT
restart clock if it was on
restore registers from PFSAVE
CF A15, U:ARES call user routine
M:DISP


I:ARES
CF A15, U:RST
call user
routine
save dispatcher
register in
A15 stack
reset flag in
CVTPWF bit 15
HALT
before restart
?
(sysgen option)
no
yes
HLT
machine runs again
if START button pushed
scan DWT
to simulate
end of I/O
if a DWT is busy, set up user ECB
with effective length is 0 and
status is /E000
send Set Event request
set control unit status free
set scheduled label parameters
update DWT
if disc, update DCT with bit for
bad cylinder
terminate logical I/O (part 2)
of M:DFM if any
decrement event counts
M:DISP

I:RTC (REAL TIME CLOCK INTERRRUPT ROUTINE)

Calling Sequence

This routine is called by hardware interrupt.

Work Areas and Tables

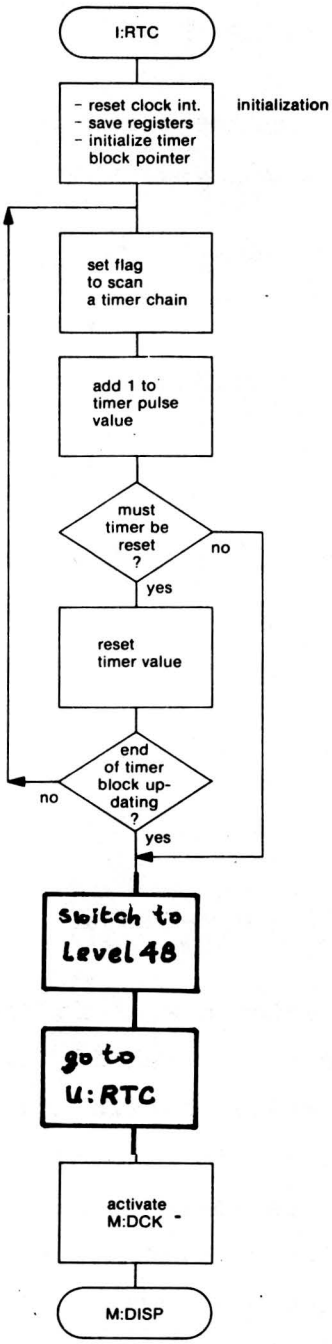
I:CPLS: a word in the Communication Vector Table giving the pulse rate.
V:FLAG: a flag vector indicating the timer to be scanned.
V:REST: a value vector to reset the timers.
H:TIME: start address of the timer block.

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.


I:RTC
- reset clock int.
- save registers
- initialize timer block pointer
initialization
set flag to scan a timer chain
add 1 to timer pulse value
must timer be reset ?
no
yes
reset timer value
end of timer block up-dating ?
no
yes
Switch to Level 48
go to U:RTC
activate M:DCK
M:DISP

USER REAL TIME CLOCK ROUTINE (U:RTC)

Calling Sequence

This module is called through the instruction CF 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 RTN A15 instruction.

The module is selected as follows:
A standard U:RTC module, which executes only an RTN 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:LKM)

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:LKM 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 48, the LKM handler branches directly to the processing routine and then the processing routine will switch to level 48 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

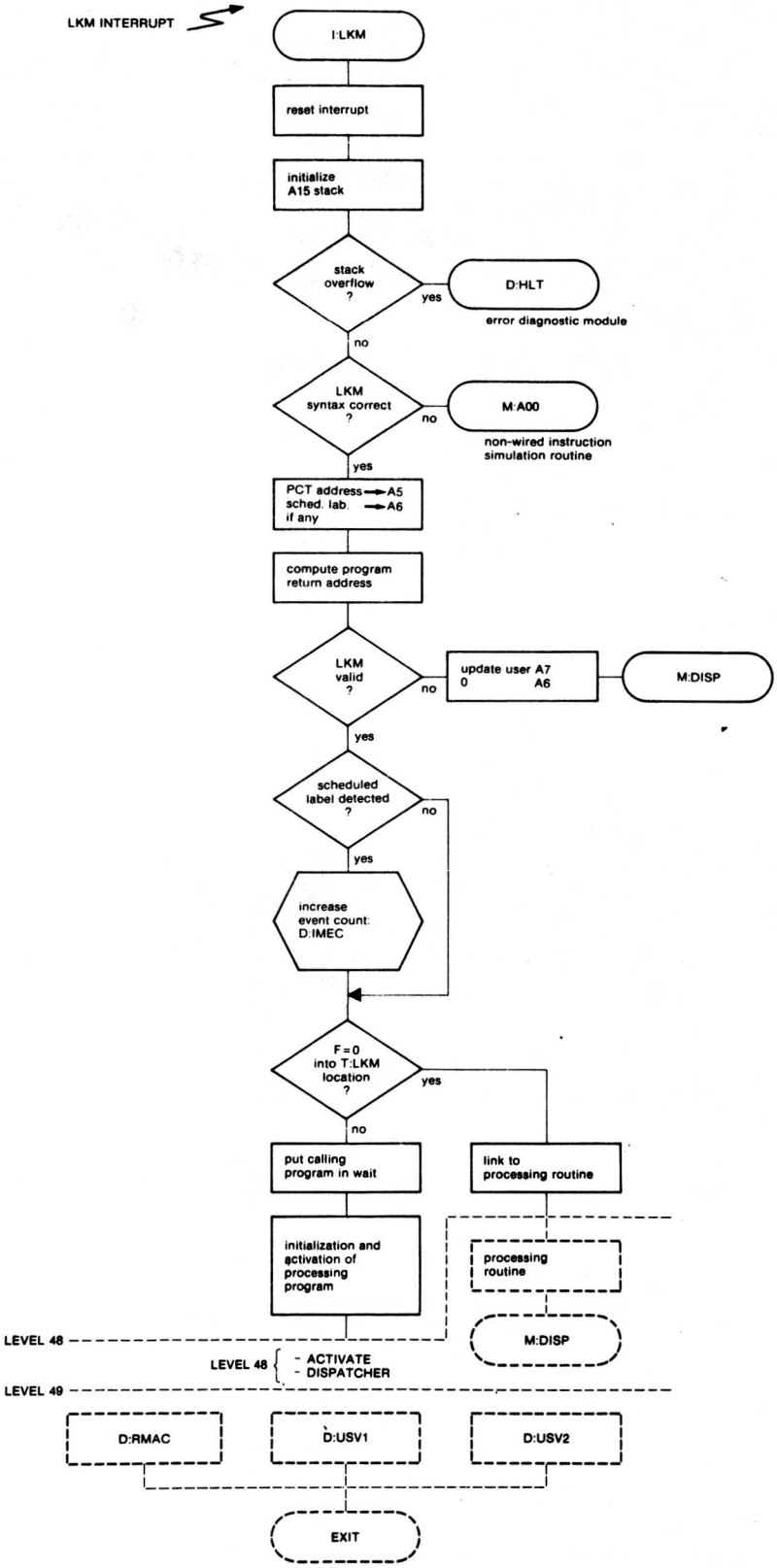
A7: User A7 parameter

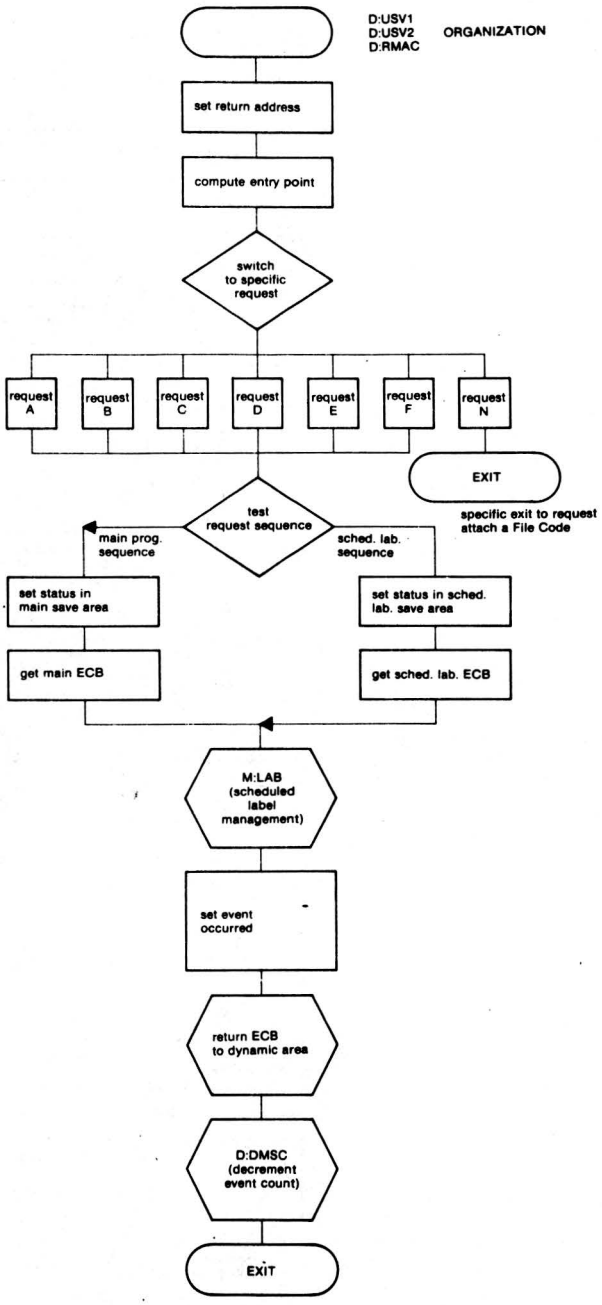
A8: User A8 paramerer.

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

A3 contains the second word of the corresponding entry in T:LKM. If the function is requested by a scheduled label sequence, the sign bit of A3 is set to 1, and if it is requested from a main program sequence it is reset to zero.


LKM INTERRUPT
I:LKM
reset interrupt
initialize A15 stack
stack overflow ?
yes
D:HLT
error diagnostic module
no
LKM syntax correct ?
no
M:A00
non-wired instruction simulation routine
yes
PCT address → A5
sched. lab. → A6
if any
compute program return address
LKM valid ?
no
update user A7
0 A6
M:DISP
yes
scheduled label detected ?
no
yes
increase event count: D:IMEC
F = 0 into T:LKM location ?
yes
no
put calling program in wait
link to processing routine
initialization and activation of processing program
processing routine
M:DISP
LEVEL 48
LEVEL 48
- ACTIVATE
- DISPATCHER
LEVEL 49
D:RMAC
D:USV1
D:USV2
EXIT


D:USV1
D:USV2
D:RMAC
ORGANIZATION
set return address
compute entry point
switch to specific request
request A
request B
request C
request D
request E
request F
request N
EXIT
specific exit to request attach a File Code
test request sequence
main prog. sequence
sched. lab. sequence
set status in main save area
set status in sched. lab. save area
get main ECB
get sched. lab. ECB
M:LAB (scheduled label management)
set event occurred
return ECB to dynamic area
D:DMSC (decrement event count)
EXIT

INPUT/OUTPUT HANDLER (M:IORM)

Calling Sequence

M:IORM is called by the LKM handler (I:LKM).

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:FCT (File Code Table)
- T:DWT (Device Work Table)
- T:LFT (Disc Logical File Table)
- T:DCT (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.

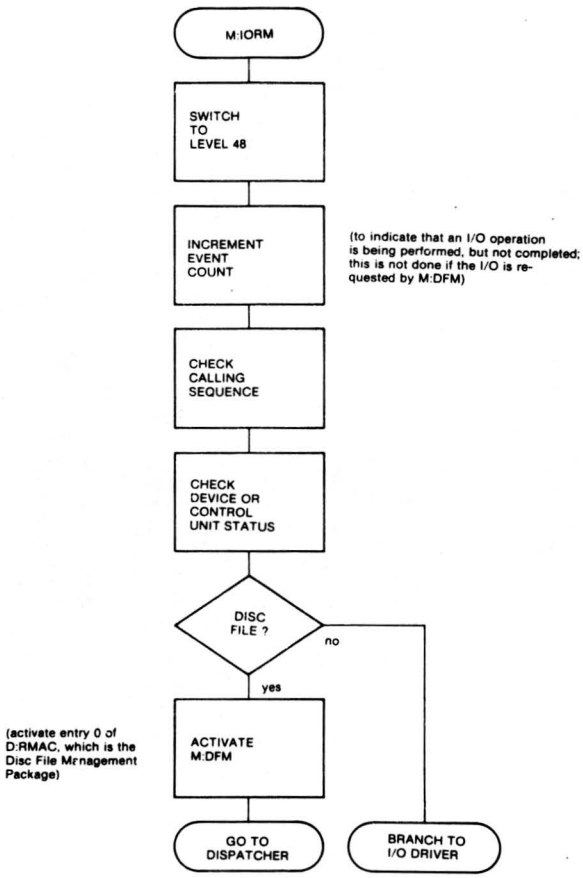
Functional Description

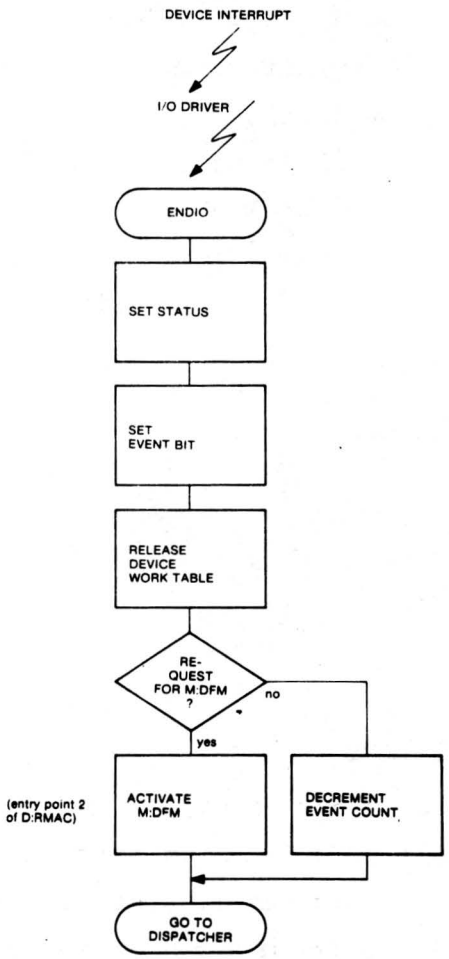
The I/O Supervisor can be regarded as combining three functions:

- pre-processing
- processing
- post-processing

Pre-processing is done by the M:IORM 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:IORM
SWITCH TO LEVEL 48
INCREMENT EVENT COUNT
(to indicate that an I/O operation is being performed, but not completed; this is not done if the I/O is requested by M:DFM)
CHECK CALLING SEQUENCE
CHECK DEVICE OR CONTROL UNIT STATUS
DISC FILE ?
no
yes
(activate entry 0 of D:RMAC, which is the Disc File Management Package)
ACTIVATE M:DFM
GO TO DISPATCHER
BRANCH TO I/O DRIVER

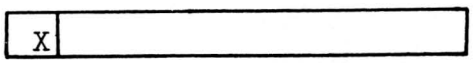

DEVICE INTERRUPT
I/O DRIVER
ENDIO
SET STATUS
SET
EVENT BIT
RELEASE
DEVICE
WORK TABLE
RE-
QUEST
FOR M:DFM
?
no
yes
(entry point 2
of D:RMAC)
ACTIVATE
M:DFM
DECREMENT
EVENT COUNT
GO TO
DISPATCHER

M:WAIT (WAIT FOR AN EVENT - LKM2)

Calling Sequence

A5: PCT address
A6: Scheduled Label
A8: Event Control Block address

ECB format:



X = 1: event has occurred.

Entry Points:

- M:WAIT; M:PUTW; M:WAWI
  (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 CVTWAT order to be used by the read-only system programs)

Work Areas and Tables

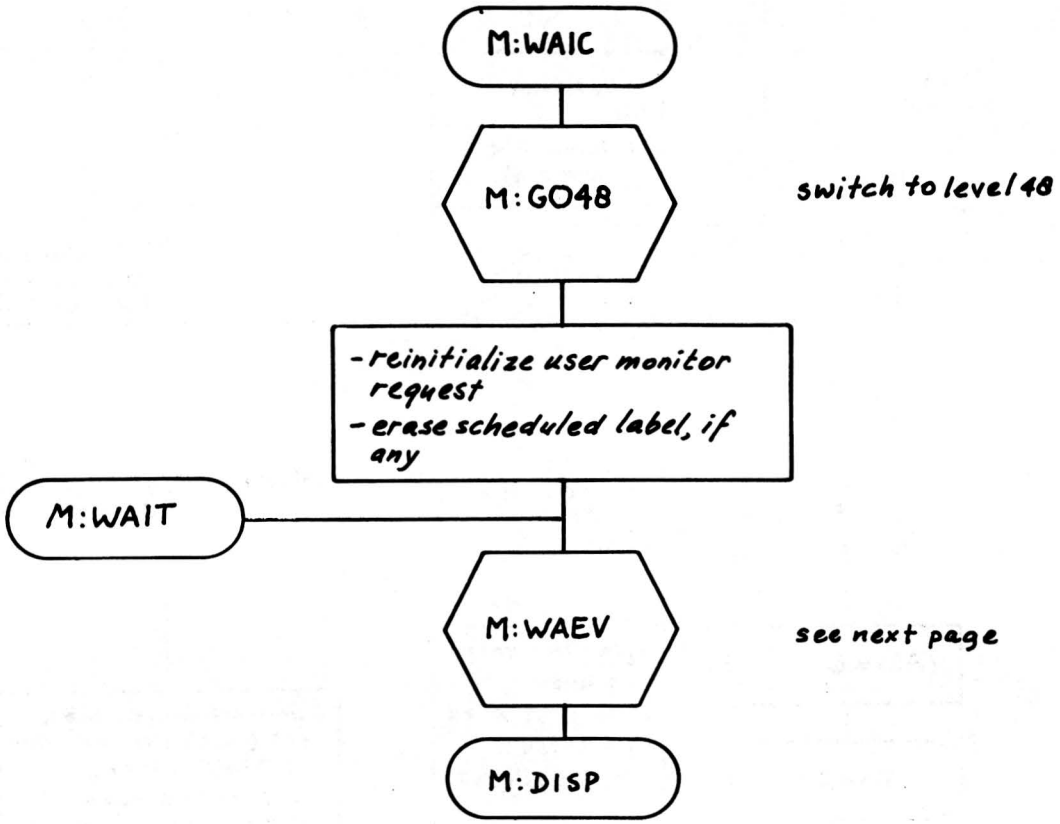
None.

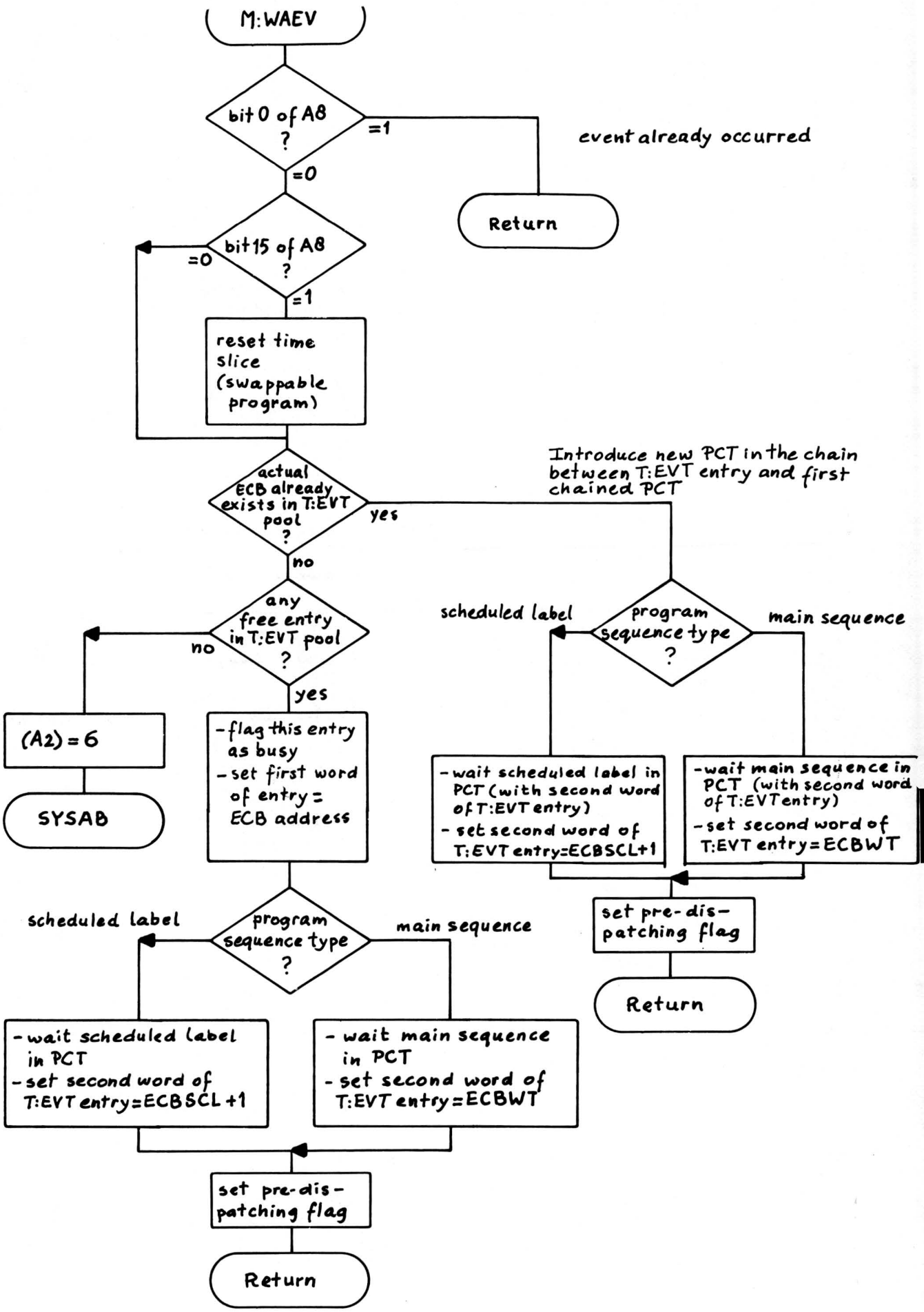
Input/Output Files

None.

Functional Description

This module processes the Wait for an Event monitor request (LKM2). 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:WAIC
M:GO48
switch to level 48
- reinitialize user monitor request
- erase scheduled label, if any
M:WAIT
M:WAEV
see next page
M:DISP


M:WAEV
bit 0 of A8 ?
=1
event already occurred
Return
=0
bit 15 of A8 ?
=0
=1
reset time slice (swappable program)
actual ECB already exists in T:EVT pool ?
yes
Introduce new PCT in the chain between T:EVT entry and first chained PCT
no
any free entry in T:EVT pool ?
no
yes
(A2) = 6
SYSAB
- flag this entry as busy
- set first word of entry = ECB address
program sequence type ?
scheduled label
main sequence
- wait scheduled label in PCT (with second word of T:EVT entry)
- set second word of T:EVT entry=ECBSCL+1
- wait main sequence in PCT (with second word of T:EVT entry)
- set second word of T:EVT entry=ECBWT
set pre-dis-patching flag
Return
program sequence type ?
scheduled label
main sequence
- wait scheduled label in PCT
- set second word of T:EVT entry=ECBSCL+1
- wait main sequence in PCT
- set second word of T:EVT entry=ECBWT
set pre-dis-patching flag
Return

M:EXIT (EXIT-LKM3)

Calling Sequence

A5: PCT address of program requesting exit.

Entry Points: - M:EXIT (entry point from I:LKM: user request)
- M:DISX (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

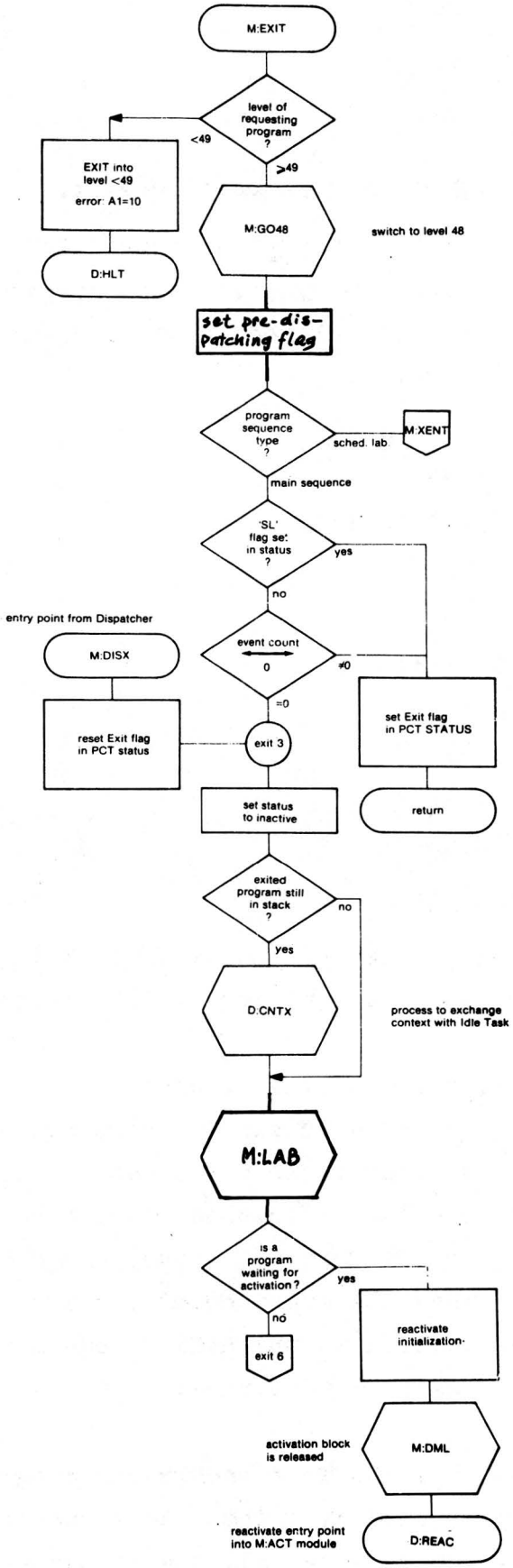
None.

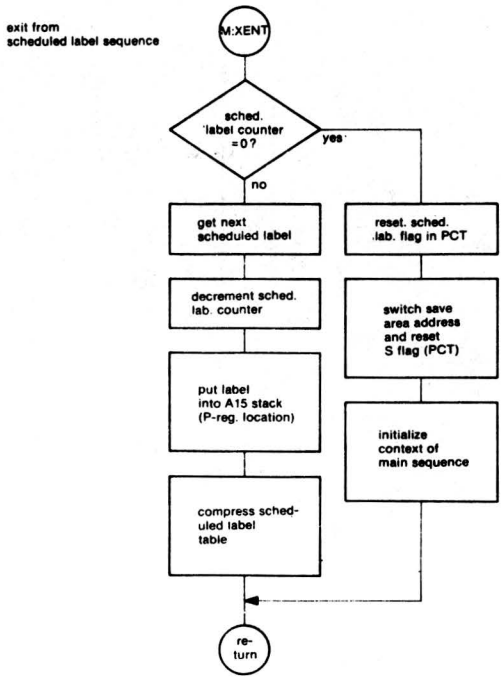
Functional Description

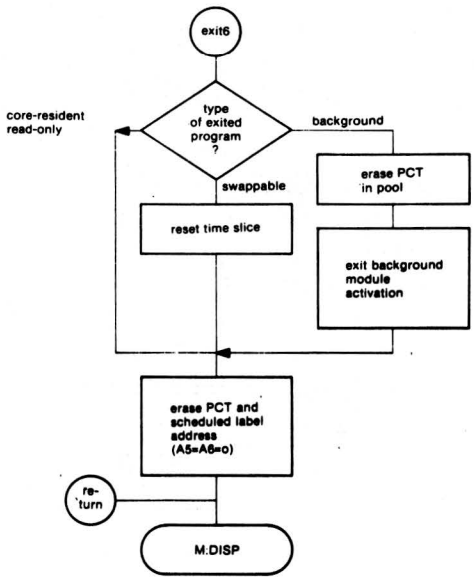
This module processes the Exit monitor request (LKM3). If all I/O is terminated and there are no more labels to be scheduled, the main program exits:

- 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:DISX) 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:EBCK) 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:EXIT
level of requesting program ?
<49
>49
EXIT into level <49
error: A1=10
D:HLT
M:GO48
switch to level 48
set pre-dis-patching flag
program sequence type ?
sched. lab
M:XENT
main sequence
'SL' flag set in status ?
yes
no
entry point from Dispatcher
M:DISX
event count
0
≠0
=0
reset Exit flag in PCT status
exit 3
set Exit flag in PCT STATUS
return
set status to inactive
exited program still in stack ?
no
yes
D:CNTX
process to exchange context with Idle Task
M:LAB
is a program waiting for activation ?
yes
no
exit 6
reactivate initialization
activation block is released
M:DML
reactivate entry point into M:ACT module
D:REAC


exit from
scheduled label sequence
M:XENT
sched.
label counter
=0?
yes
no
get next
scheduled label
decrement sched.
lab. counter
put label
into A15 stack
(P-reg. location)
compress sched-
uled label
table
reset. sched.
lab. flag in PCT
switch save
area address
and reset
S flag (PCT)
initialize
context of
main sequence
re-
turn


exit6
core-resident
read-only
type
of exited
program
?
background
swappable
erase PCT
in pool
reset time slice
exit background
module
activation
erase PCT and
scheduled label
address
(A5=A6=0)
re-
turn
M:DISP

M:GBUF, M:FBUF (GET BUFFER - LKM4/RELEASE BUFFER - LKM5)

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:FBUF:

  A14: address of the buffer previously reserved for the calling program by a Get Buffer request.

Entry Point: M:FBUF

Work Areas and Tables

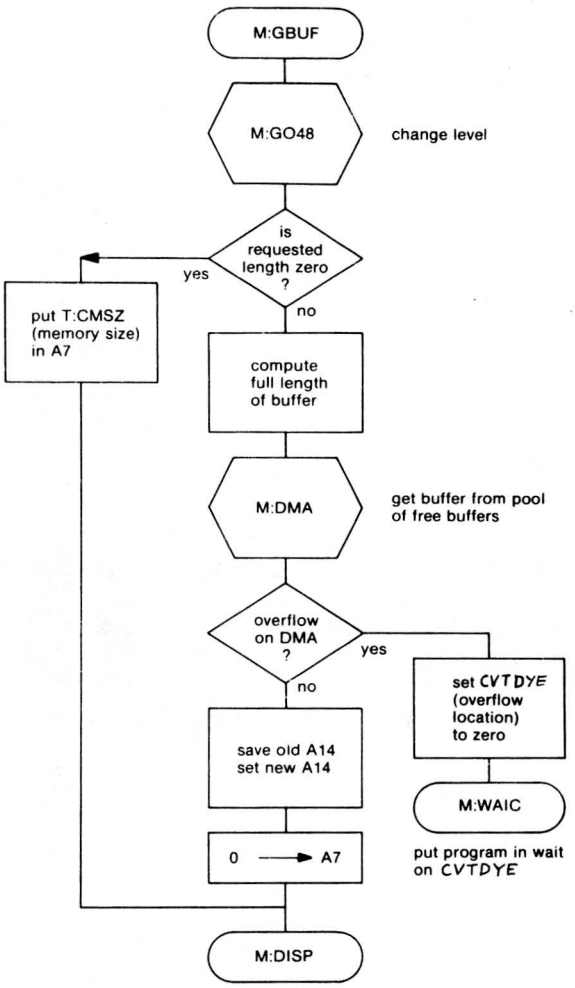
T:CVT: Communication Vector Table.
PCT entry of calling program.
Dynamically Allocated Memory Area (buffer).

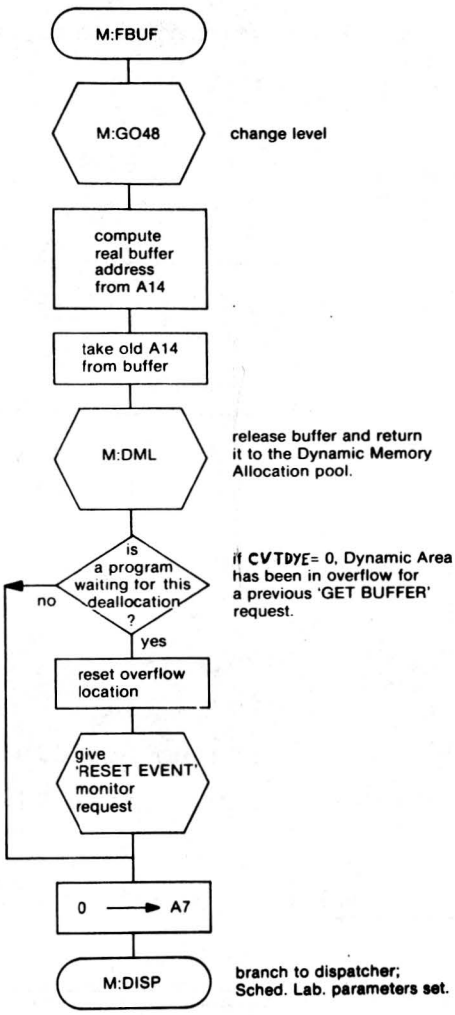
Input/Output Files

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 its request for a buffer.


M:GBUF
M:GO48
change level
is requested length zero ?
yes
no
put T:CMSZ (memory size) in A7
compute full length of buffer
M:DMA
get buffer from pool of free buffers
overflow on DMA ?
yes
no
set CVTDYE (overflow location) to zero
M:WAIC
put program in wait on CVTDYE
save old A14 set new A14
0 → A7
M:DISP


M:FBUF
M:GO48
change level
compute real buffer address from A14
take old A14 from buffer
M:DML
release buffer and return it to the Dynamic Memory Allocation pool.
is a program waiting for this deallocation ?
no
yes
if CVTDYE= 0, Dynamic Area has been in overflow for a previous 'GET BUFFER' request.
reset overflow location
give 'RESET EVENT' monitor request
0 → A7
M:DISP
branch to dispatcher; Sched. Lab. parameters set.

D:CNTM (CONNECT A PROGRAM TO A TIMER - LKM10)

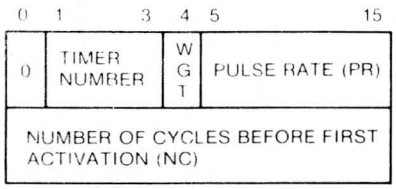
Calling Sequence

Entry Point: D:CNTM

A7: address of program name block.

A8: address of 2-word parameter block, which may be of one of the following two formats:

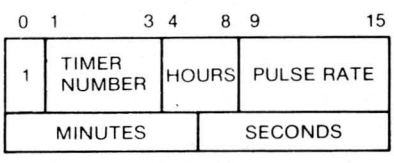
Standard Connection:



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:



where

PR is a value from 0 to 127.

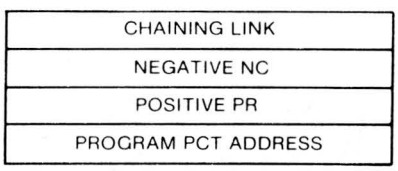
The program specified via A7 is connected to the absolute time chain. At the time defined by the user (HH MM SS), it is started, disconnected from this chain and connected to the chain on the timer defined by the user in bits 1 to 3. This is managed by the M:DCK module.

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.

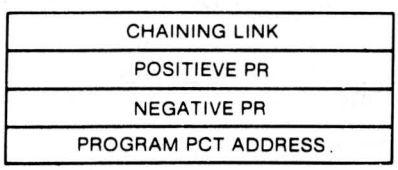
Work Areas and Tables

H:POIN Chain Pointer
When a connection is requested from D:CNTM, a 4-word block is automatically reserved in the dynamic allocation area. The format of such a block is as follows:

- Standard Connection

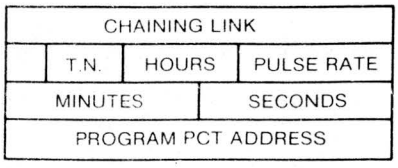


After the first program activation, the block format is as follows (unless PR was 0, in which case automatic disconnection will follow):



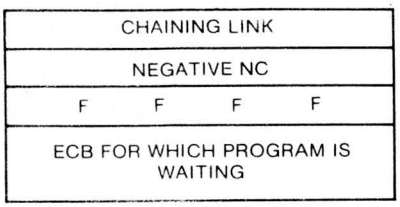
where the third word contains the PR value as updated by M:DCK.

- Absolute Time Connection



After the first program activation, this block is reinitialized in standard format.

- 'Wait for a Given Time' Connection



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 LKM request, which activates D:RMAC if it is core resident or D:USV2 if the request handler is disc resident.
First, D:CNTM checks the type of request in order to set or reset the WGT 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:CNTM returns to D:RMAC or D:USV2 with the Status set. One of the following values is returned to the

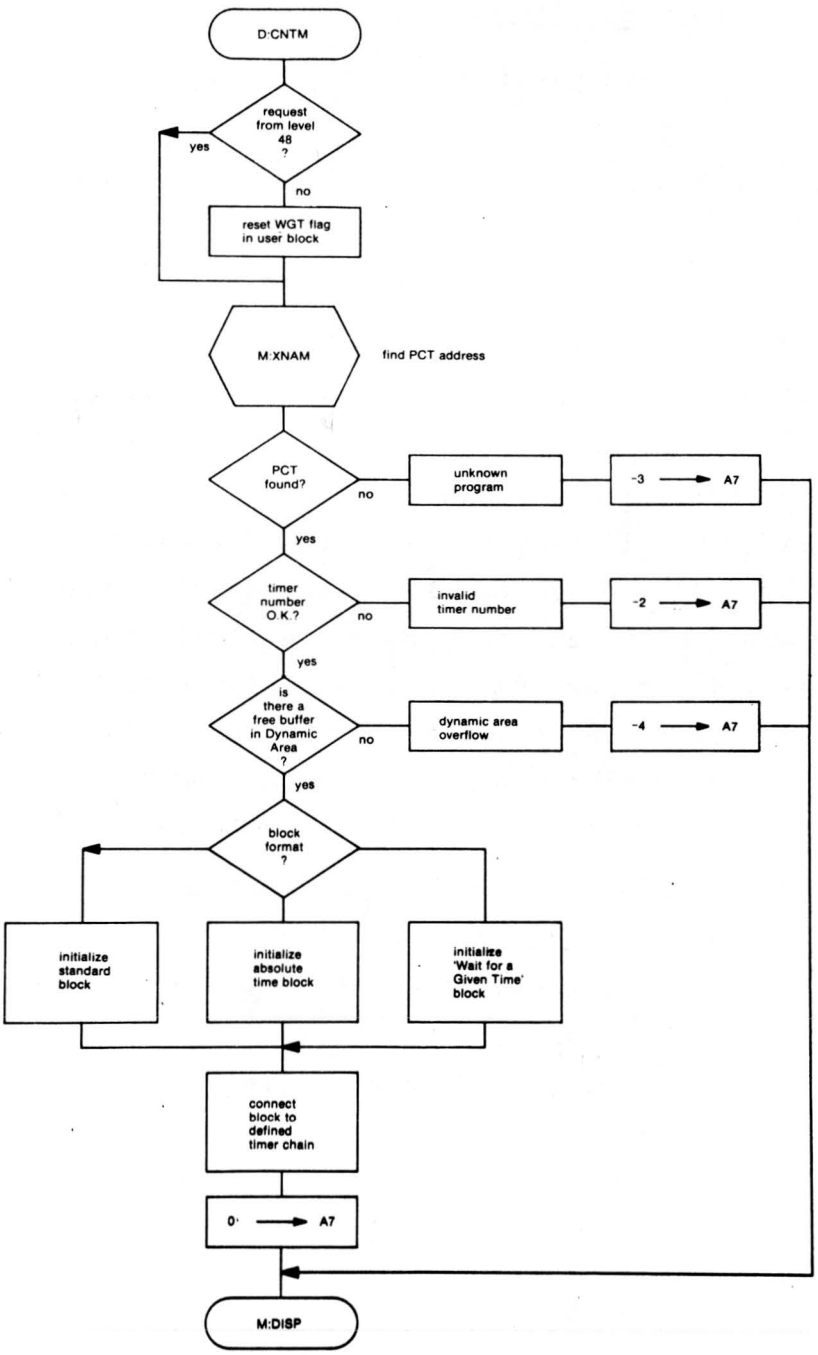
user in the A7 register:

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

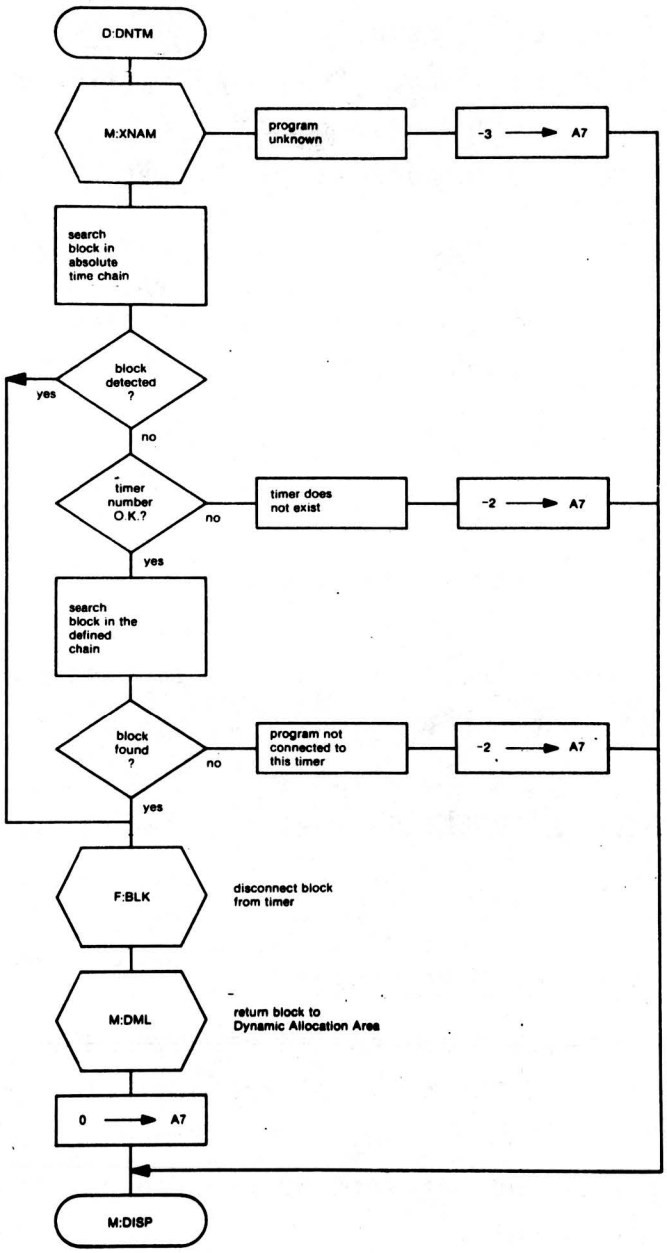
Input/Output Files

None.

Functional Description

D:DNTM is started by LKM request, activating D:RMAC 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:XNAM 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.


D:DNTM
M:XNAM
program unknown
-3 → A7
search block in absolute time chain
block detected ?
yes
no
timer number O.K.?
no
timer does not exist
-2 → A7
yes
search block in the defined chain
block found ?
no
program not connected to this timer
-2 → A7
yes
F:BLK
disconnect block from timer
M:DML
return block to Dynamic Allocation Area
0 → A7
M:DISP

M:ACT (ACTIVATE - LKM12)

Calling Sequence

- If request comes via an LKM interrupt:
  A7: address of program name block of program to be activated.
  A8: ECB address.
- If request comes from D:ASYS (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:ASYS

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

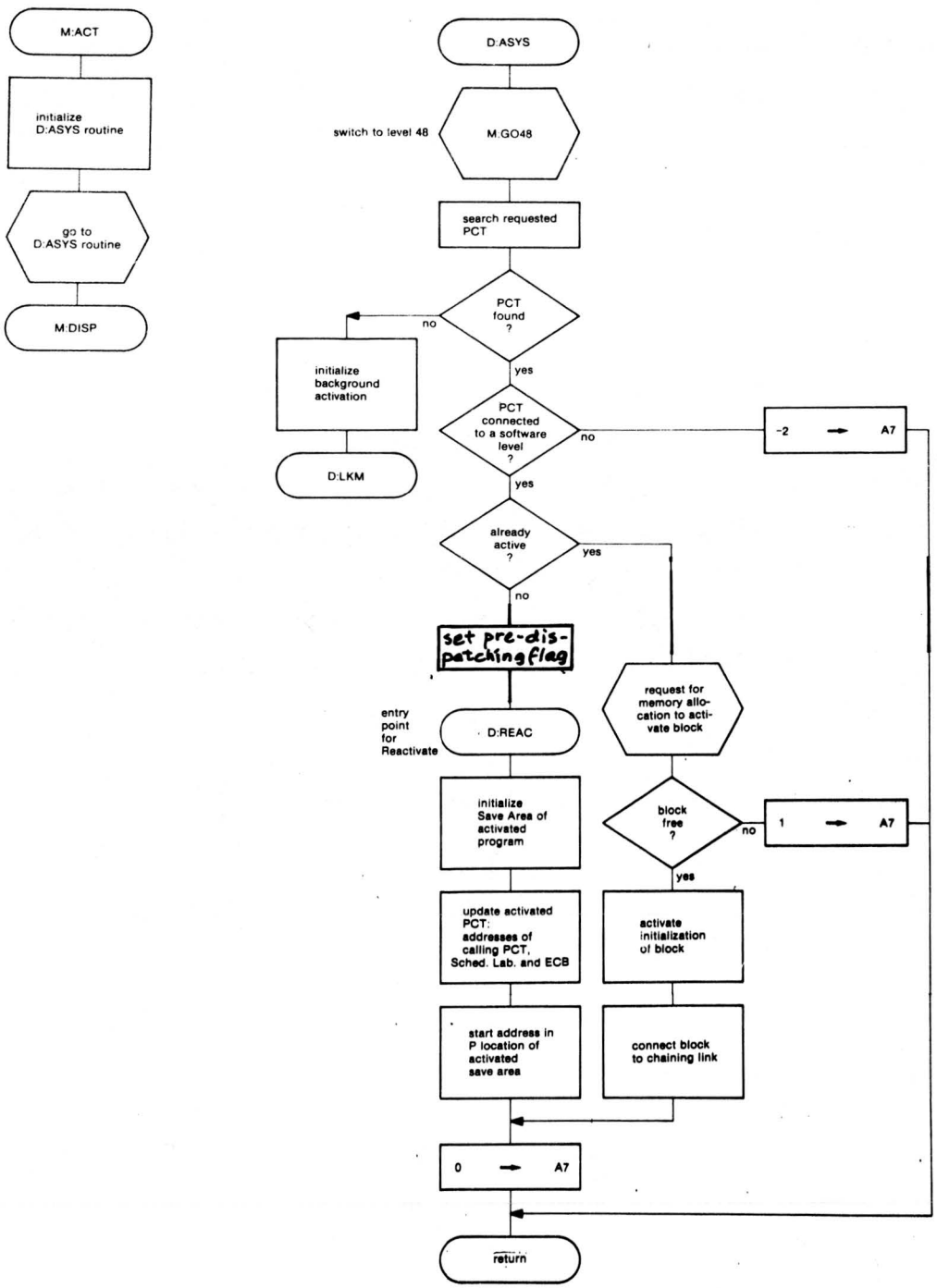
None.

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:ABCK, 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 - LKM13)

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).

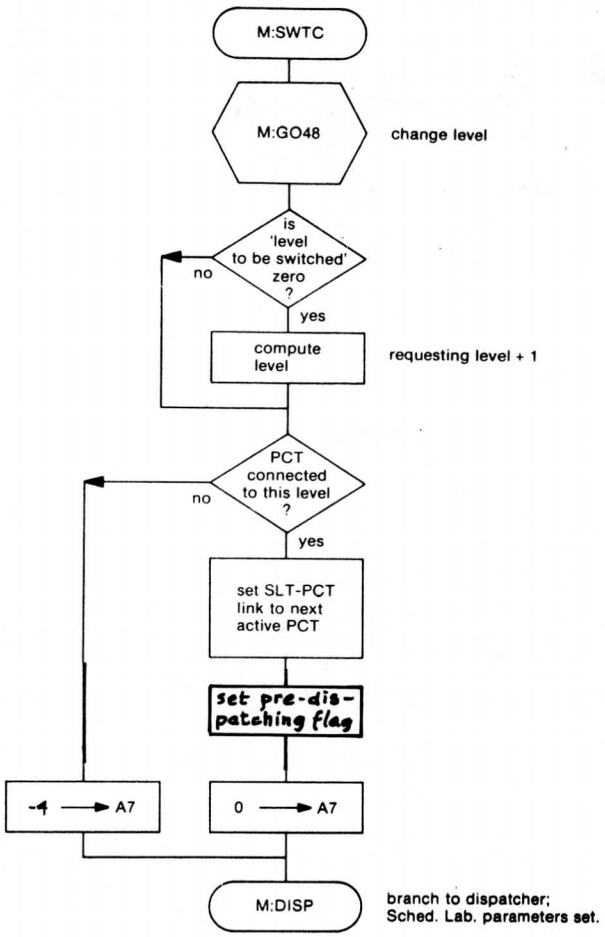
Input/Output Files

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.


M:SWTC
M:GO48
change level
is 'level to be switched' zero ?
no
yes
compute level
requesting level + 1
PCT connected to this level ?
no
yes
set SLT-PCT link to next active PCT
set pre-dis-patching flag
-1 → A7
0 → A7
M:DISP
branch to dispatcher; Sched. Lab. parameters set.

D:ATDT (ATTACH/DETACH A DEVICE TO/FROM A PROGRAM - LKM 14 + 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: M:ATDV

Work Areas and Tables

FCT (File Code Table)
DWT (Device Work Table)
LFT (Logical File Table)
PCT (Program Control Table)
An 8-word block in Dynamic Allocation Area.

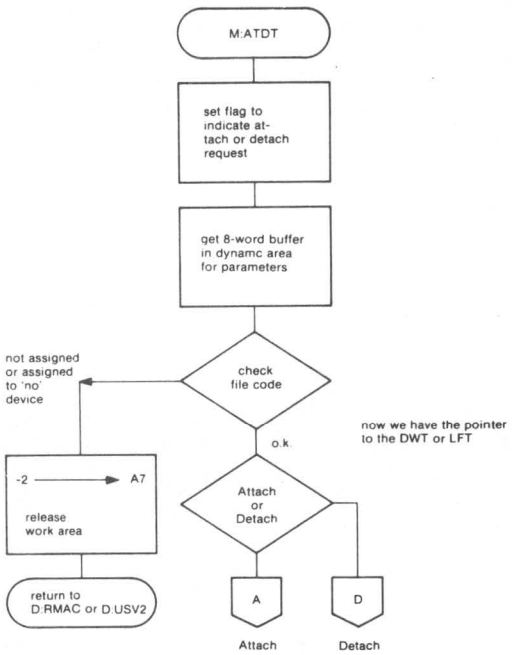
Input/Output Files

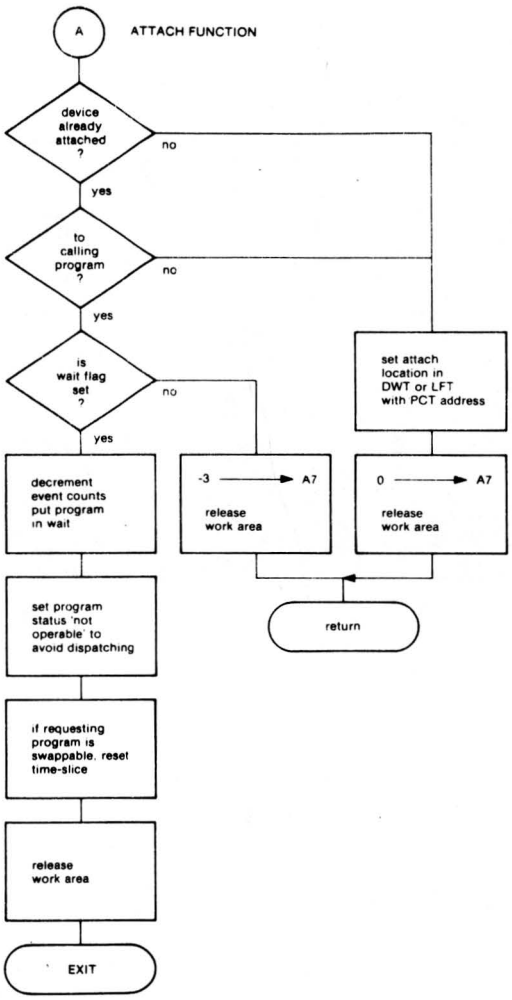
None.

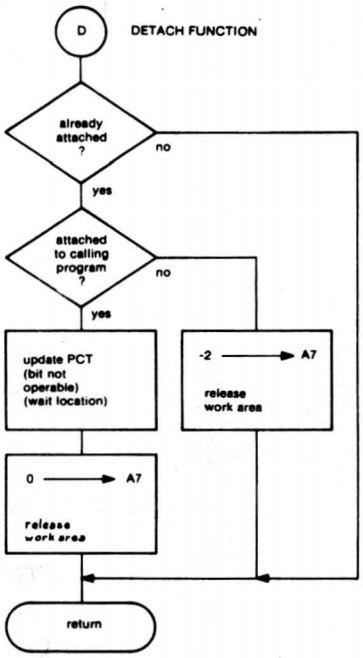
Functional Description

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 /8000.

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.


M:ATDT
set flag to indicate attach or detach request
get 8-word buffer in dynamc area for parameters
check file code
not assigned or assigned to 'no' device
o.k.
now we have the pointer to the DWT or LFT
-2 → A7
release work area
return to D:RMAC or D:USV2
Attach or Detach
A
D
Attach
Detach


A
ATTACH FUNCTION
device already attached ?
no
yes
to calling program ?
no
yes
is wait flag set ?
no
yes
set attach location in DWT or LFT with PCT address
-3 → A7
release work area
0 → A7
release work area
return
decrement event counts put program in wait
set program status 'not operable' to avoid dispatching
if requesting program is swappable, reset time-slice
release work area
EXIT


D
DETACH FUNCTION
already attached ?
no
yes
attached to calling program ?
no
yes
update PCT (bit not operable) (wait location)
-2 → A7
release work area
0 → A7
release work area
return

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

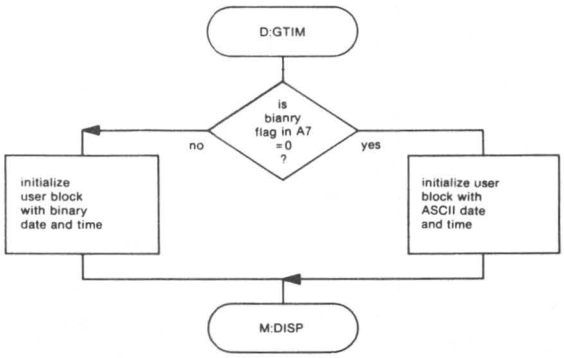
The timer block is read.

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).


D:GTIM
is
bianry
flag in A7
=0
?
no
yes
initialize
user block
with binary
date and time
initialize user
block with
ASCII date
and time
M:DISP

M:RSEV (SET AN EVENT - LKM18)

Calling Sequence

A5: PCT address of calling program

A6: Scheduled Label

A8: Event Control Block address.

Entry Point: M:RSEV

M:SEEV (pointed to through CVTSET and used by the read-only system programs)

Work Areas and Tables

T:SLT (Software Level Table).

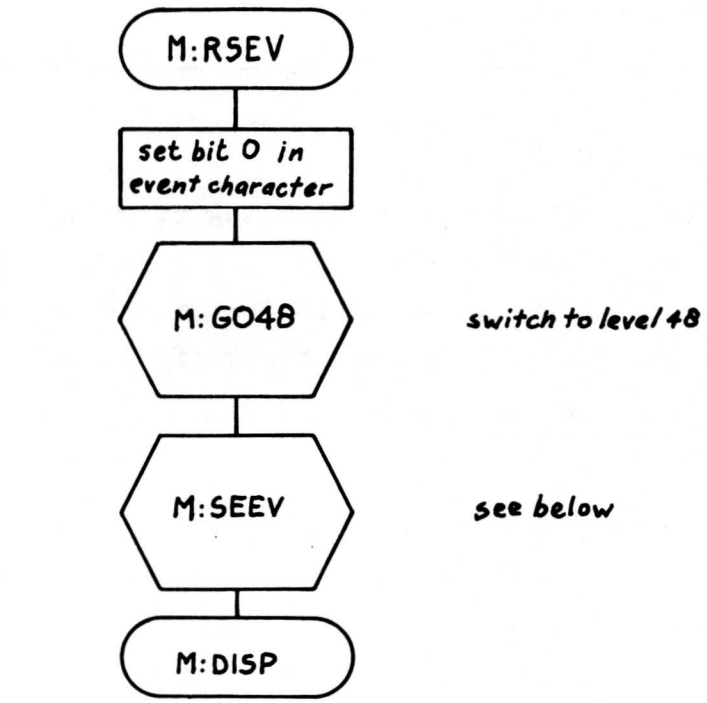
PCT (Program Control Table).

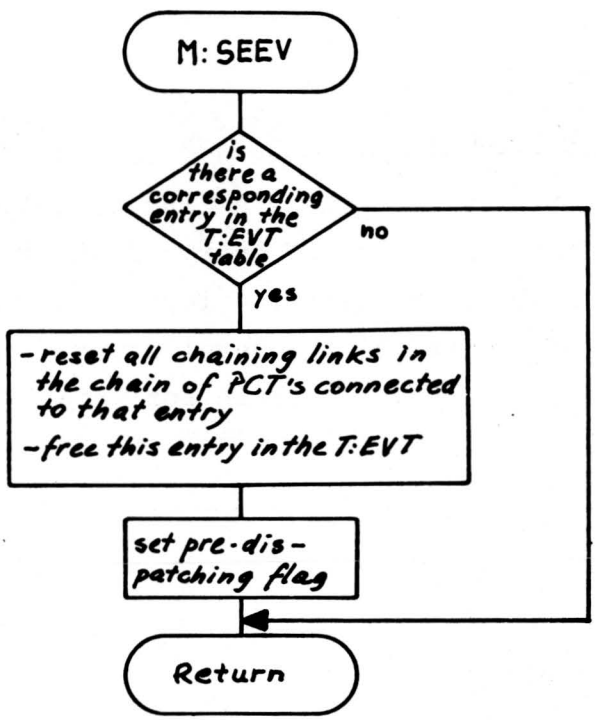
Input/Output Files

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 The M:RSEV module looks for the ECB 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.


M:RSEV
set bit 0 in event character
M:GO48
switch to level 48
M:SEEV
see below
M:DISP


M:SEEV
is there a corresponding entry in the T:EVT table
no
yes
- reset all chaining links in the chain of PCT's connected to that entry
- free this entry in the T:EVT
set pre-dis-patching flag
Return

D:CNLV (CONNECT A LEVEL TO A PROGRAM - LKM20)

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

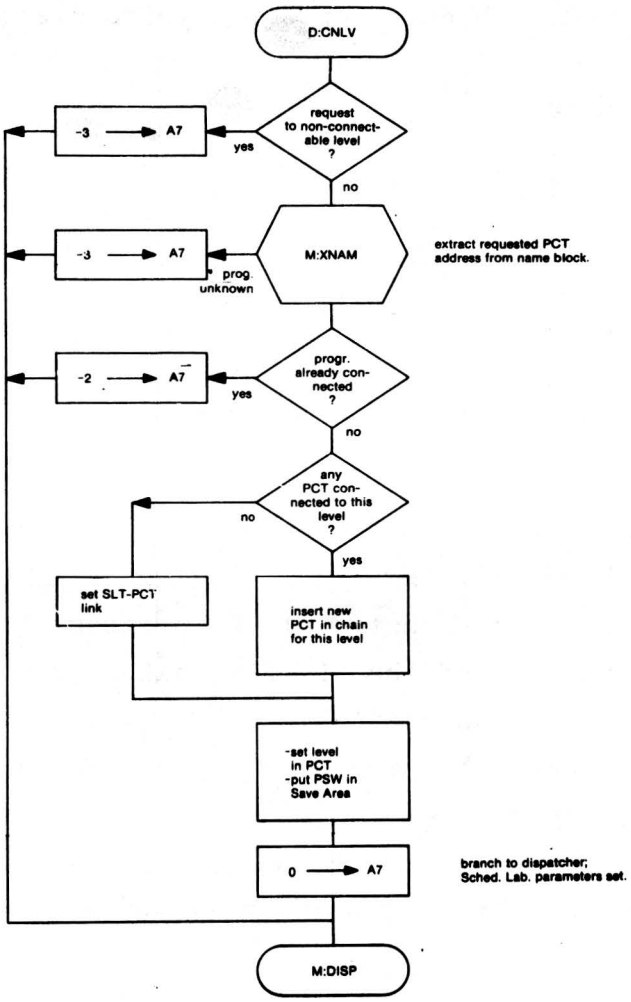
T:SLT (Software Level Table)
PCT (Program Control Table)
Save area of program which must be connected.

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:CNLV
request to non-connect-able level ?
yes
-3 → A7
no
M:XNAM
extract requested PCT address from name block.
* prog. unknown
-3 → A7
progr. already con-nected ?
yes
-2 → A7
no
any PCT con-nected to this level ?
no
yes
set SLT-PCT link
insert new PCT in chain for this level
-set level in PCT
-put PSW in Save Area
0 → A7
branch to dispatcher; Sched. Lab. parameters set.
M:DISP

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

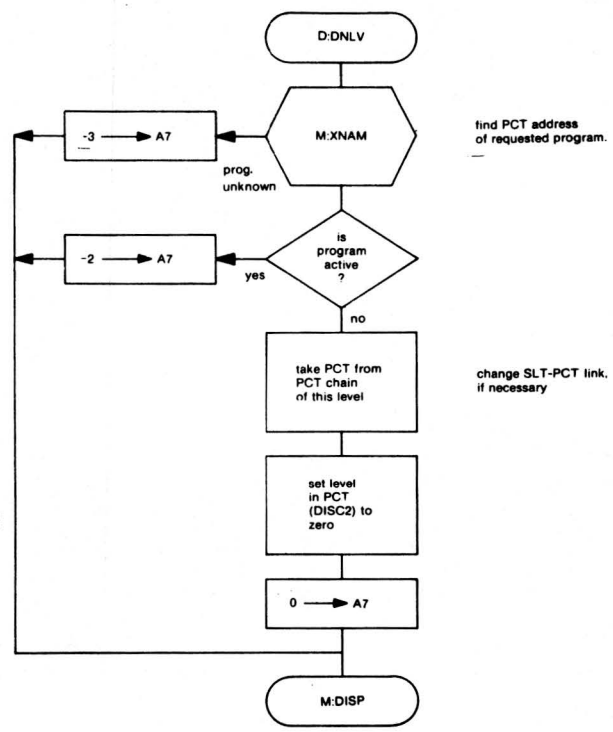
T:SLT (Software Level Table).
PCT (Program Control Table).

Input/Output Files

None.

Functional Description

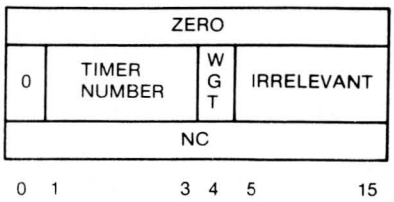
With this request the PCT entry of the program which must be disconnected is taken out of the SLT-PCT chain, provided the corresponding program is in inactive state.
If there is only one PCT entry in the chain for this level, the SLT-PCT link is reset to zero.
If there are more entries in the chain, the requested PCT entry is removed. In both cases the level specified in the PCT entry (DISK2), is reset to zero.


D:DNLV
M:XNAM
find PCT address
of requested program.
-3 → A7
prog.
unknown
is
program
active
?
-2 → A7
yes
no
take PCT from
PCT chain
of this level
change SLT-PCT link,
if necessary
set level
in PCT
(DISC2) to
zero
0 → A7
M:DISP

D:WGT (WAIT FOR A GIVEN TIME - LKM22)

Calling Sequence

A8: address of an Event Control Block initialized as follows:



where NC is the number of timer cycles before the program is re-started.

Entry Point: D:WGT

Work Areas and Tables

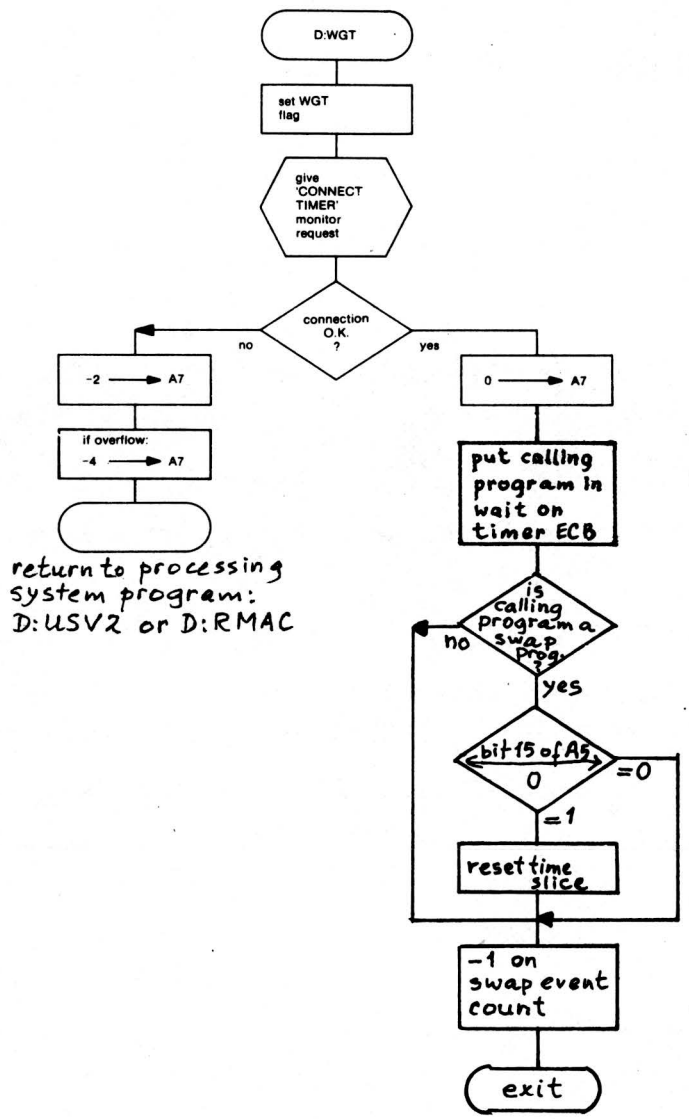
None.

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 ECB 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 M:DCK4 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.


D:WGT
set WGT flag
give 'CONNECT TIMER' monitor request
connection O.K. ?
no
yes
-2 → A7
if overflow: -4 → A7
return to processing system program: D:USV2 or D:RMAC
0 → A7
put calling program in wait on timer ECB
is calling program a swap Prog. ?
no
yes
bit 15 of A5
0
=0
=1
reset time slice
-1 on swap event count
exit

ASGPRO (ASSIGN A FILE CODE - LKM23)

Calling Sequence

A5: PCT address of calling program
A6: Scheduled Label, if any
A8: Pointer to user parameter block
A11: PCT address of D:USV1 or D:RMAC
A12: Return address (to D:USV1 or D:RMAC)
A13: CVT address

Work Areas and Tables

FCT (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.

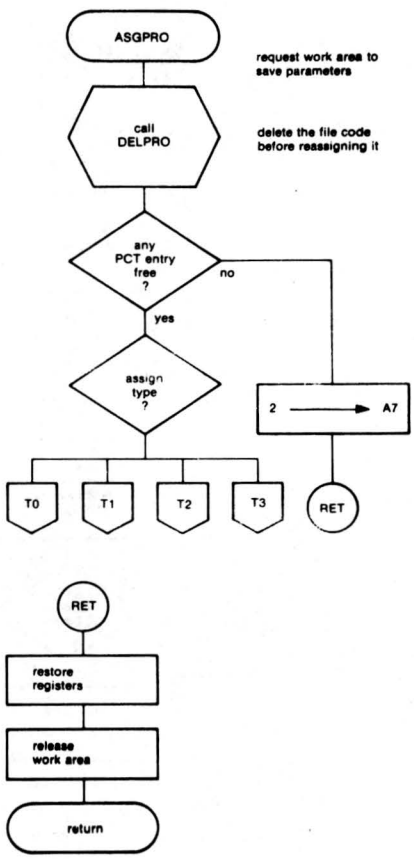
Input/Output Files

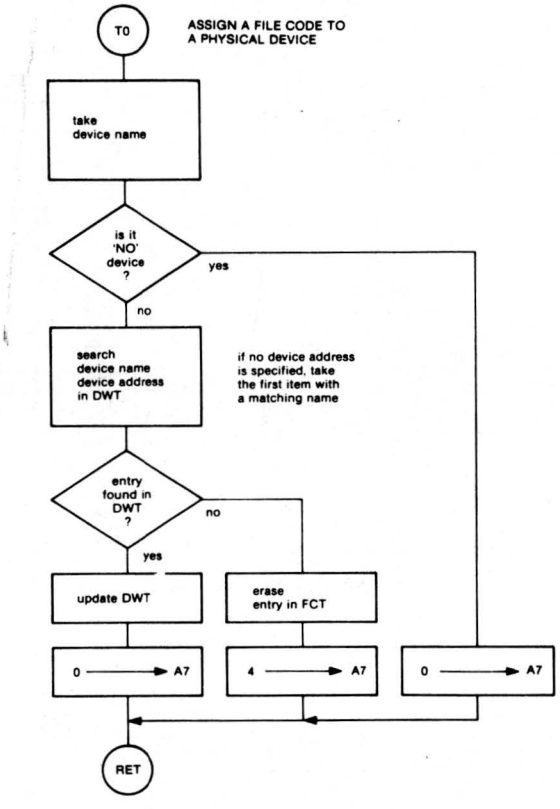
Disc is used to read in the directory or GRANTB.

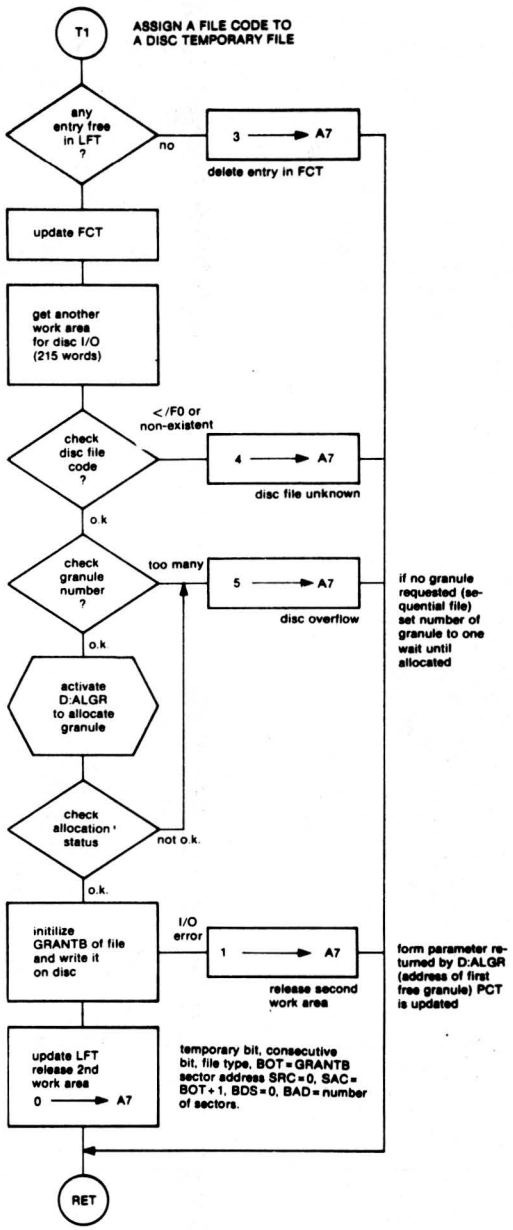
Functional Description

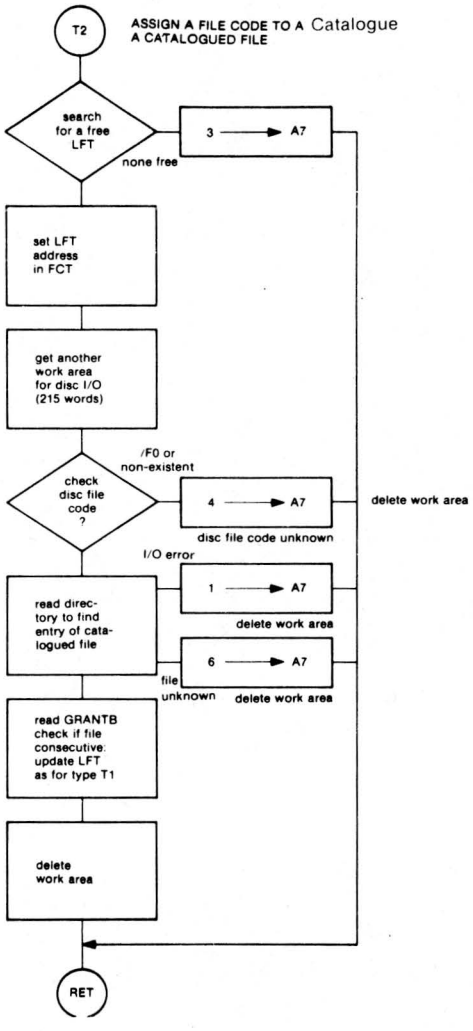
The action taken depends on the type of file code assignment
- for physical devices, the Device Work Table is updated
- for disc temporary files, the File Code Table is updated, any necessary granules are allocated, the granule table GRANT is initialized and the Logical File Table is updated.
- for catalogued 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 equalizing file codes, the File Code Table is updated. If it is a disc logical file, ASCNT in LFT is incremented as well.

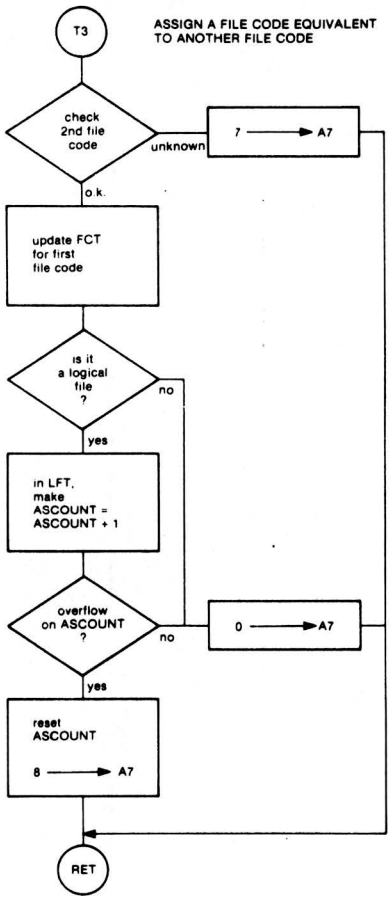
On return, registers A5, A6, A11 and A13 are not destroyed and A7 will contain the status of the operation.


ASGPRO
request work area to save parameters
call DELPRO
delete the file code before reassigning it
any PCT entry free ?
no
yes
assign type ?
2
A7
T0
T1
T2
T3
RET
RET
restore registers
release work area
return


T0
ASSIGN A FILE CODE TO
A PHYSICAL DEVICE
take
device name
is it
'NO'
device
?
yes
no
search
device name
device address
in DWT
if no device address
is specified, take
the first item with
a matching name
entry
found in
DWT
?
no
yes
update DWT
erase
entry in FCT
0 → A7
4 → A7
0 → A7
RET


T1
ASSIGN A FILE CODE TO A DISC TEMPORARY FILE
any entry free in LFT ?
no
3 → A7
delete entry in FCT
update FCT
get another work area for disc I/O (215 words)
check disc file code ?
< /F0 or non-existent
4 → A7
disc file unknown
o.k
check granule number ?
too many
5 → A7
disc overflow
if no granule requested (sequential file) set number of granule to one wait until allocated
o.k
activate D:ALGR to allocate granule
check allocation status
not o.k.
o.k.
initilize GRANTB of file and write it on disc
I/O error
1 → A7
release second work area
form parameter returned by D:ALGR (address of first free granule) PCT is updated
update LFT release 2nd work area
0 → A7
temporary bit, consecutive bit, file type, BOT = GRANTB sector address SRC = 0, SAC = BOT + 1, BDS = 0, BAD = number of sectors.
RET


T2
ASSIGN A FILE CODE TO A Catalogue
A CATALOGUED FILE
search for a free LFT
none free
3 → A7
set LFT address in FCT
get another work area for disc I/O (215 words)
/F0 or non-existent
check disc file code ?
4 → A7
disc file code unknown
delete work area
I/O error
read directory to find entry of catalogued file
1 → A7
delete work area
6 → A7
file unknown
delete work area
read GRANTB check if file consecutive: update LFT as for type T1
delete work area
RET


T3
ASSIGN A FILE CODE EQUIVALENT TO ANOTHER FILE CODE
check 2nd file code
unknown
7 → A7
o.k.
update FCT for first file code
is it a logical file ?
no
yes
in LFT, make ASCOUNT = ASCOUNT + 1
overflow on ASCOUNT ?
no
0 → A7
yes
reset ASCOUNT
8 → A7
RET

DELPRO (DELETE A FILE CODE - LKM24)

Calling Sequence

A5: PCT address of calling program
A6: Scheduled Label, if any
A8: User parameter block address
A11: PCT address of D:USV1 or D:RMAC
A12: Return address (to D:USV1 or D:RMAC)
A13: CVT address

Work Areas and Tables

FCT (File Code Table)
LFT (Logical File Table)
DCT (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.

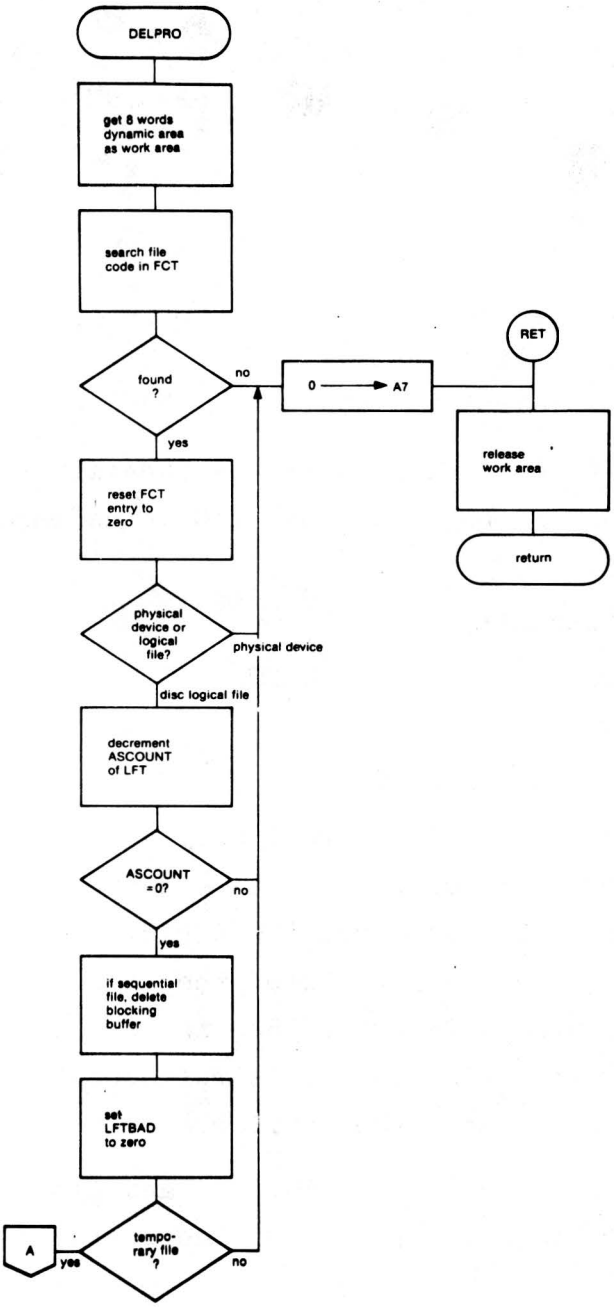
Input/Output Files

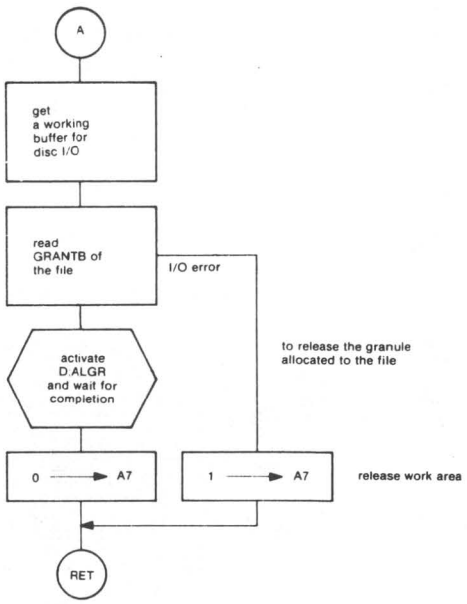
Disc is used to read in GRANTB.

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 LFTMD2 of the Logical File Description table is decremented and word LFTBAD 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.


DELPRO
get 8 words dynamic area as work area
search file code in FCT
found ?
no
0 → A7
RET
release work area
return
yes
reset FCT entry to zero
physical device or logical file?
physical device
disc logical file
decrement ASCOUNT of LFT
ASCOUNT =0?
no
yes
if sequential file, delete blocking buffer
set LFTBAD to zero
A
yes
tempo-rary file ?
no


A
get
a working
buffer for
disc I/O
read
GRANTB of
the file
I/O error
to release the granule
allocated to the file
activate
D:ALGR
and wait for
completion
0 → A7
1 → A7
release work area
RET

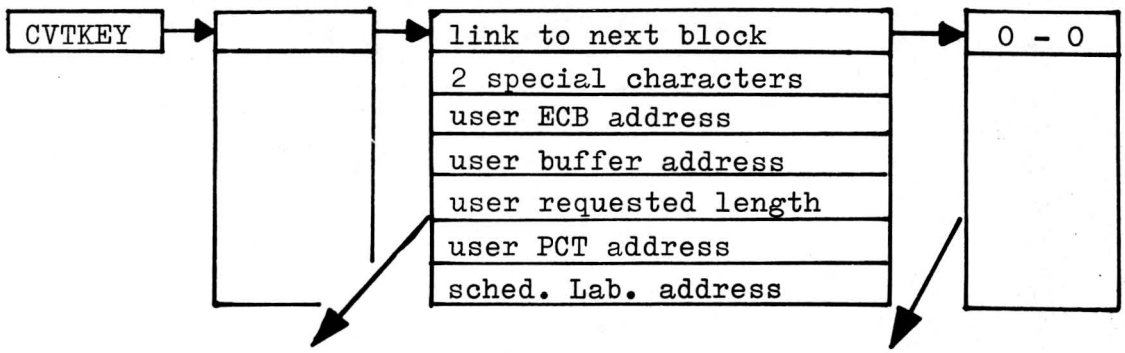
KEYPRO (READ AN UNSOLICITED KEY-IN - LKM25)

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:



format of one block

chaining pointer of last entry = 0

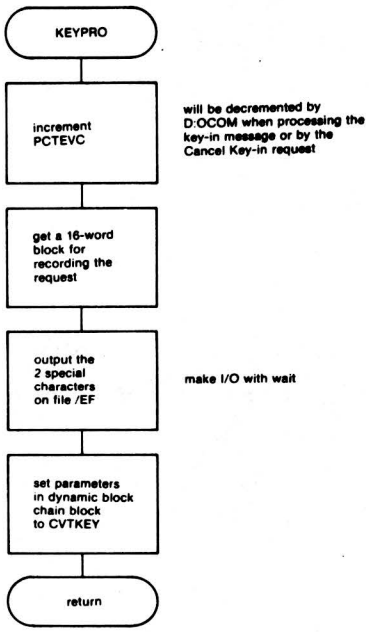
Input/Output Files

File Code /EF (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.



CANKEY (CANCEL AN UNSOLICITED KEY-IN REQUEST - LKM26)

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:

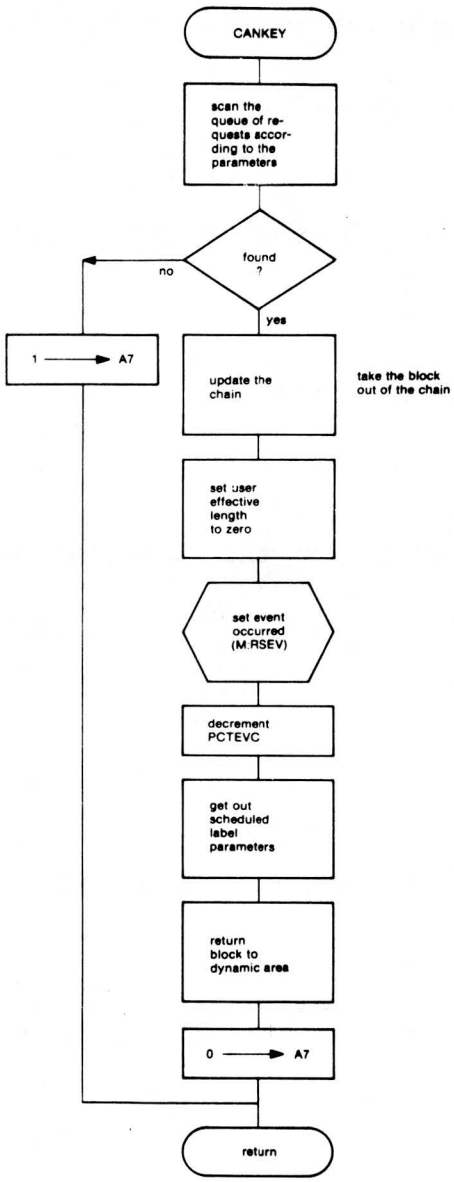
See under KEYPRO

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 countains the status of the request after completion: if 0, the operation was successful, if 1, it was not.


CANKEY
scan the queue of requests according to the parameters
found ?
no
yes
1 → A7
update the chain
take the block out of the chain
set user effective length to zero
set event occurred (M:RSEV)
decrement PCTEVC
get out scheduled label parameters
return block to dynamic area
0 → A7
return

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

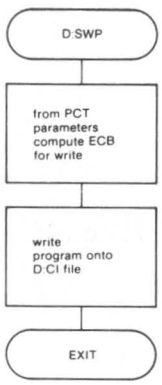
PCT (Program Control Table)
A 6-word block to format an ECB for the I/O operation.

Input/Output Files

This module writes on the D:CI file.

Functional Description

See page 2-17 for a description of the swapping procedure.


D:SWP
from PCT
parameters
compute ECB
for write
write
program onto
D:CI file
EXIT

DISC FILE MANAGEMENT (M:DFM)

Calling Sequence

Upon entry, M:DFM requires the address of the Logical File Description Table (T:LFT) 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:RMAC, because Disc File Management is part of the D:RMAC 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:LFT)
- Disc Control Table (T:DCT)

Input/Output Files

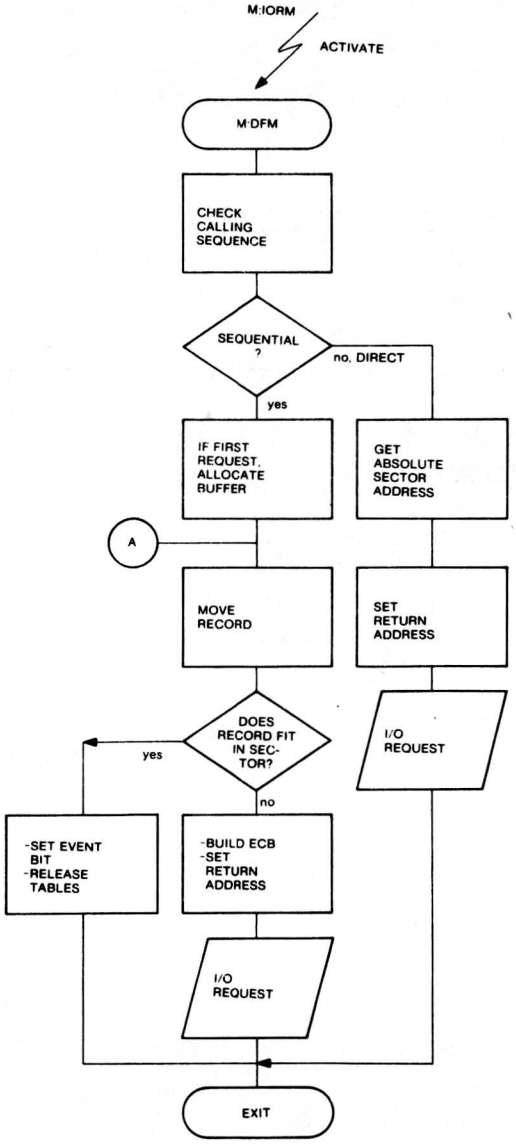
Disc File Management uses the disc file codes /F0 to /FF to perform physical I/O operations.

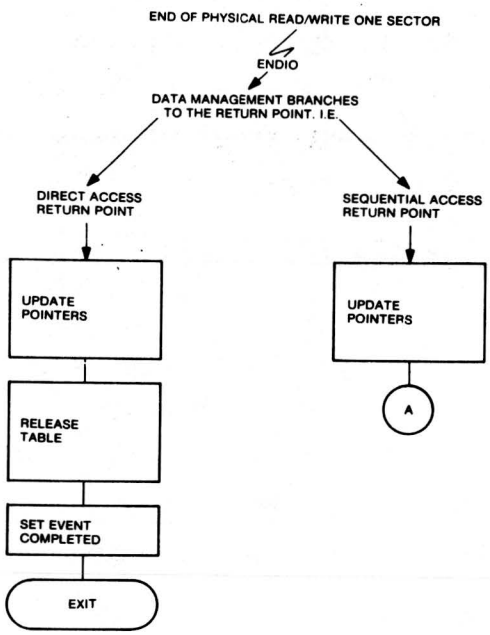
Memory Layout

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:IORM 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.


M:IORM
ACTIVATE
M:DFM
CHECK CALLING SEQUENCE
SEQUENTIAL ?
no, DIRECT
yes
IF FIRST REQUEST, ALLOCATE BUFFER
GET ABSOLUTE SECTOR ADDRESS
A
MOVE RECORD
SET RETURN ADDRESS
DOES RECORD FIT IN SEC-TOR?
yes
no
I/O REQUEST
-SET EVENT BIT
-RELEASE TABLES
-BUILD ECB
-SET RETURN ADDRESS
I/O REQUEST
EXIT


END OF PHYSICAL READ/WRITE ONE SECTOR
ENDIO
DATA MANAGEMENT BRANCHES TO THE RETURN POINT. I.E.
DIRECT ACCESS RETURN POINT
SEQUENTIAL ACCESS RETURN POINT
UPDATE POINTERS
UPDATE POINTERS
A
RELEASE TABLE
SET EVENT COMPLETED
EXIT

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 PU␣DNDA␣STAT␣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 - 8: ECB used to print the message.
- 9 - n: are used to format the message.

To print the message DKER 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 typwriter, with file code /EF.
For the dump command, however, D:OCOM also must use the disc unit file code (/F0 to /FF) and the print file (/20) which can be assigned to any output device.

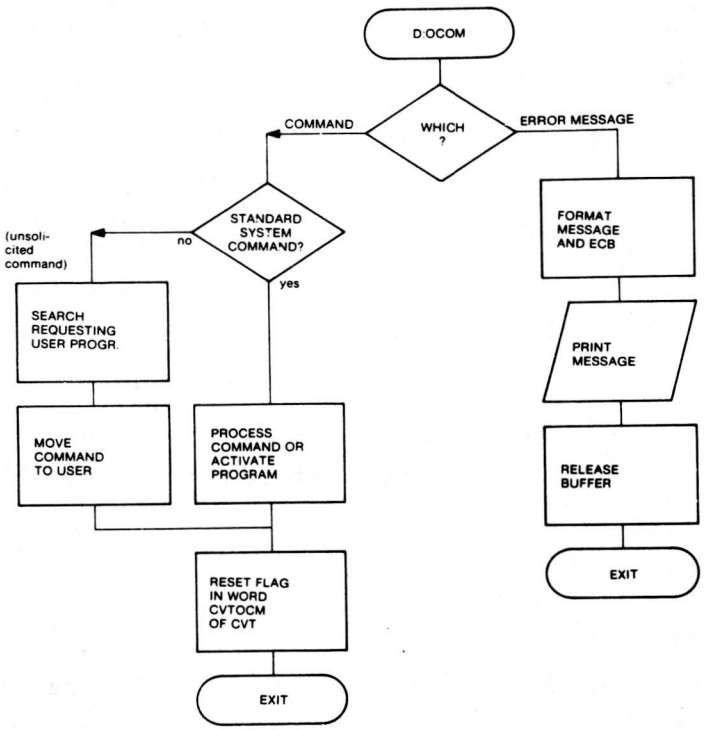
Memory Layout

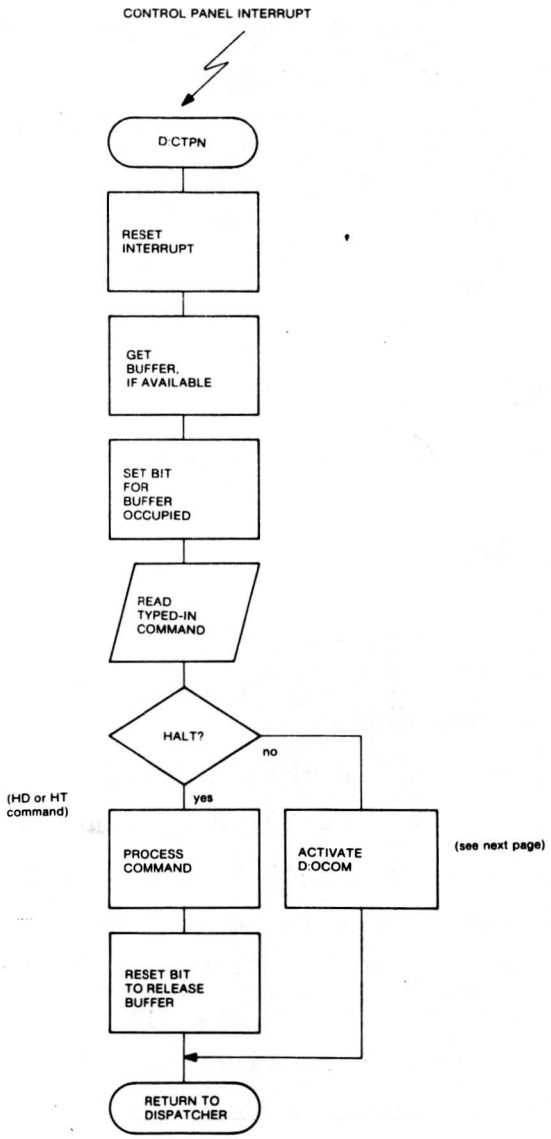
The operator communication package consists of several modules. The D:CTPN module (used to read control panel key-ins) is the only one which is memory resident. The others are always read only.
If desired, the user may also declare D:CTPN a read only program, but the priority level to which it is connected 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 processor 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.




CONTROL PANEL INTERRUPT
D:CTPN
RESET INTERRUPT
GET BUFFER, IF AVAILABLE
SET BIT FOR BUFFER OCCUPIED
READ TYPED-IN COMMAND
HALT?
no
yes
(HD or HT command)
PROCESS COMMAND
ACTIVATE D:OCOM
(see next page)
RESET BIT TO RELEASE BUFFER
RETURN TO DISPATCHER

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:GBUF)
If bit 0 = 0: system request.

A3: Base address of area in which buffer is allocated.

```
INH
CF A15, M:DMA
```

Entry Point: M:DMA

Work Areas and Tables

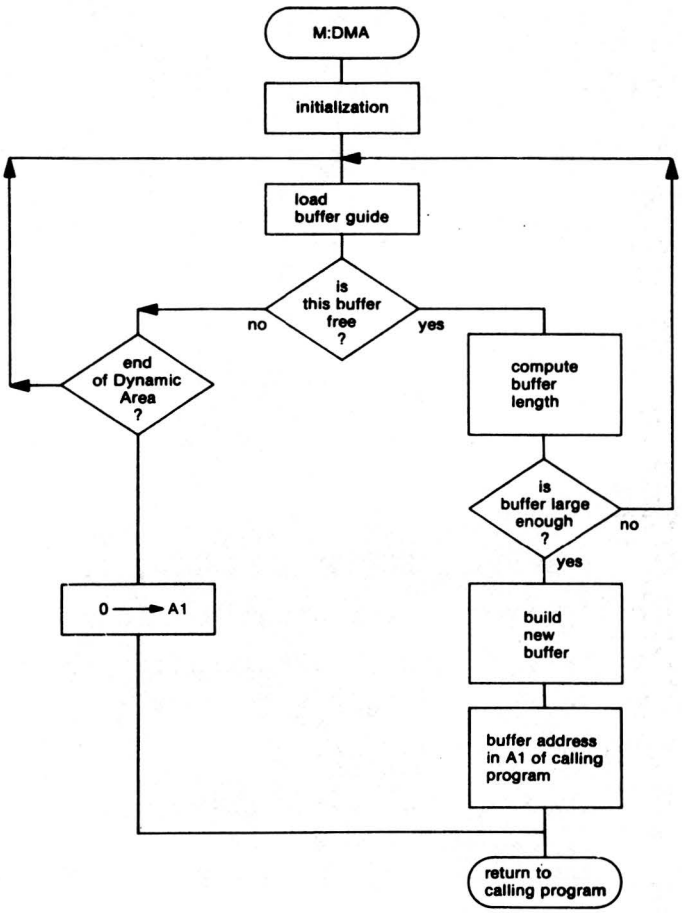
Dynamic Allocation Area (See Chapter 3).

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:DMA
initialization
load
buffer guide
is
this buffer
free
?
no
yes
end
of Dynamic
Area
?
compute
buffer
length
is
buffer large
enough
?
no
yes
0 → A1
build
new
buffer
buffer address
in A1 of calling
program
return to
calling program

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
INH
CF A15, M:DML

Entry Point: M:DML

Work Areas and Tables

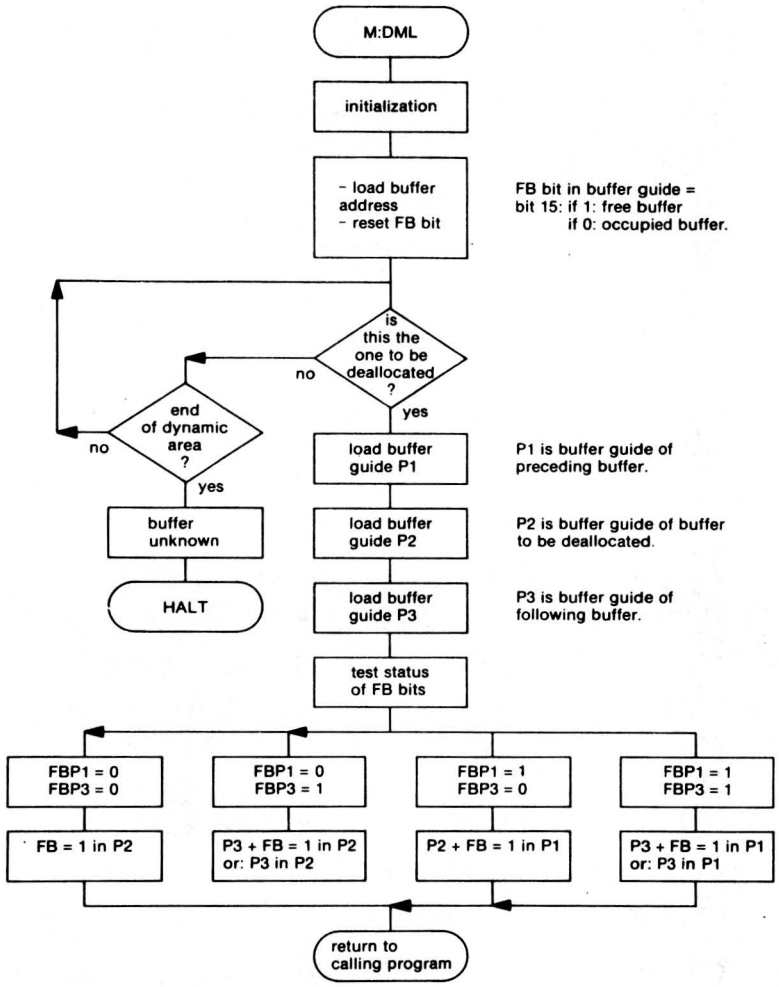
Dynamic Allocation Area (see chapter 3).

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:DML
initialization
- load buffer address
- reset FB bit
FB bit in buffer guide =
bit 15: if 1: free buffer
if 0: occupied buffer.
is this the one to be deallocated ?
no
yes
end of dynamic area ?
no
yes
load buffer guide P1
P1 is buffer guide of preceding buffer.
buffer unknown
load buffer guide P2
P2 is buffer guide of buffer to be deallocated.
HALT
load buffer guide P3
P3 is buffer guide of following buffer.
test status of FB bits
FBP1 = 0
FBP3 = 0
FBP1 = 0
FBP3 = 1
FBP1 = 1
FBP3 = 0
FBP1 = 1
FBP3 = 1
FB = 1 in P2
P3 + FB = 1 in P2
or: P3 in P2
P2 + FB = 1 in P1
P3 + FB = 1 in P1
or: P3 in P1
return to calling program

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:RSET: a value vector to reset the timers.
H:TIME: timer block start address.
H:POIN: chain pointer start address.
All programs connected to timer blocks (built by the M:CNTM module) are managed by M:DCK.

Input/Output Files

None.

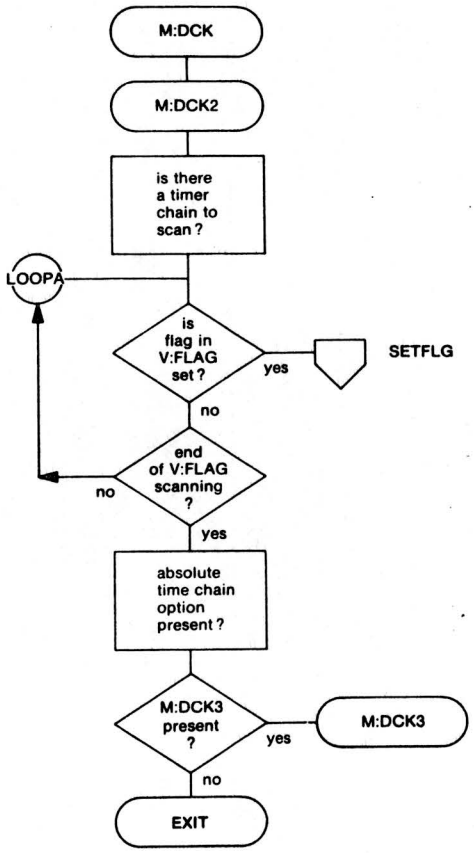
Functional Description

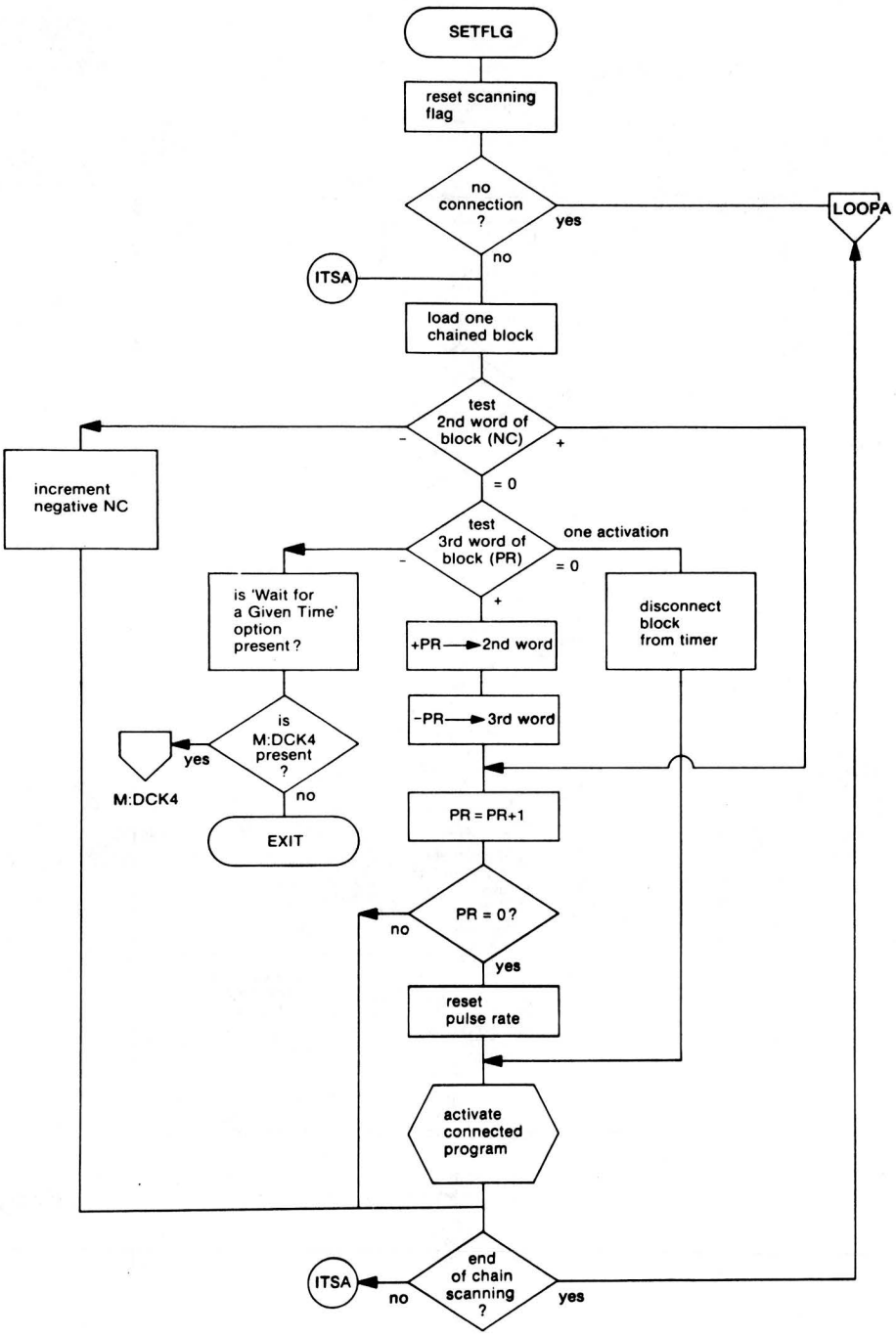
This module is started on activation by the I:RTC clock driver and runs at level 49.
It is composed of three modules:

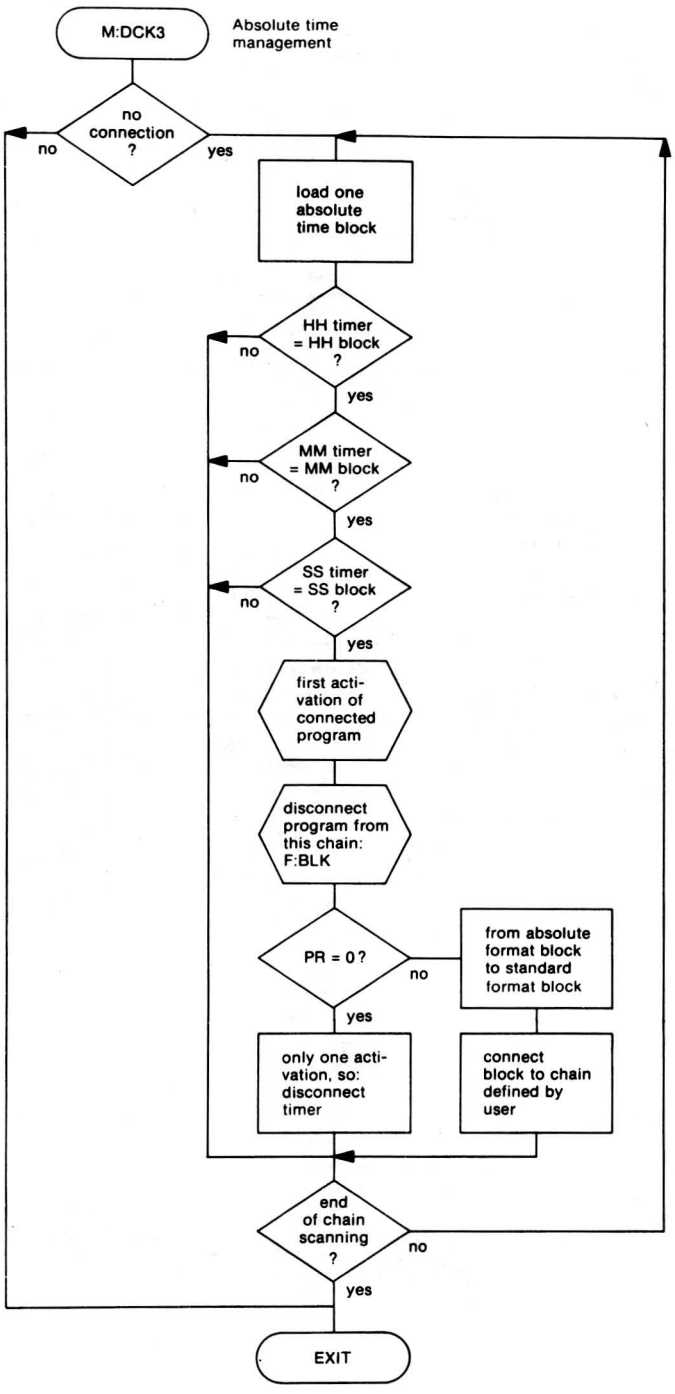
- M:DCK2 manages the chain of programs connected to standard timers.
- M:DCK3 manages the chain of programs connected to the absolute time (NC = HH␣MM␣SS).
- M:DCK4 manages the programs which are put in 'Wait for a Given Time'.

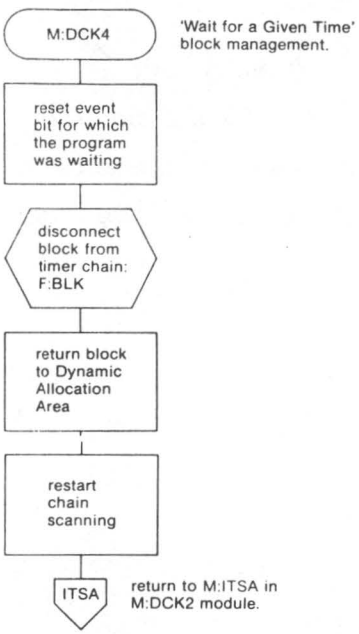
M:DCK2 first checks V:FLAG (initialized by I:RTC) to find out if a chain of programs connected to a timer must be analyzed.
At the end of M:DCK2, or in case no programs are connected to the scanned timer chains, the absolute time option is asked for. If it is present, M:DCK3 is started, otherwise M:DCK2 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, M:DCK4 is started. Otherwise, the block is ignored. At the end, control is returned to the dispatcher through an Exit request.


M:DCK
M:DCK2
is there a timer chain to scan ?
LOOPA
is flag in V:FLAG set ?
yes
SETFLG
no
end of V:FLAG scanning ?
no
yes
absolute time chain option present ?
M:DCK3 present ?
yes
M:DCK3
no
EXIT


SETFLG
reset scanning flag
no connection ?
yes
LOOPA
no
ITSA
load one chained block
test 2nd word of block (NC)
-
+
increment negative NC
= 0
test 3rd word of block (PR)
one activation
-
= 0
+
is 'Wait for a Given Time' option present ?
+PR → 2nd word
disconnect block from timer
is M:DCK4 present ?
-PR → 3rd word
yes
M:DCK4
no
EXIT
PR = PR+1
PR = 0 ?
no
yes
reset pulse rate
activate connected program
ITSA
end of chain scanning ?
no
yes


M:DCK3
Absolute time management
no connection ?
no
yes
load one absolute time block
HH timer = HH block ?
no
yes
MM timer = MM block ?
no
yes
SS timer = SS block ?
no
yes
first activation of connected program
disconnect program from this chain: F:BLK
PR = 0 ?
no
yes
from absolute format block to standard format block
only one activation, so: disconnect timer
connect block to chain defined by user
end of chain scanning ?
no
yes
EXIT


M:DCK4
'Wait for a Given Time' block management.
reset event bit for which the program was waiting
disconnect block from timer chain: F:BLK
return block to Dynamic Allocation Area
restart chain scanning
ITSA
return to M:ITSA in M:DCK2 module.

F: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: F:BLK.

This module is called only by:
- D:DNTM (Disconnect a Timer module)
- M:DCK3 (Absolute time management module)
- M:DCK4 ('Wait for a Given Time' management module).

Work Areas and Tables

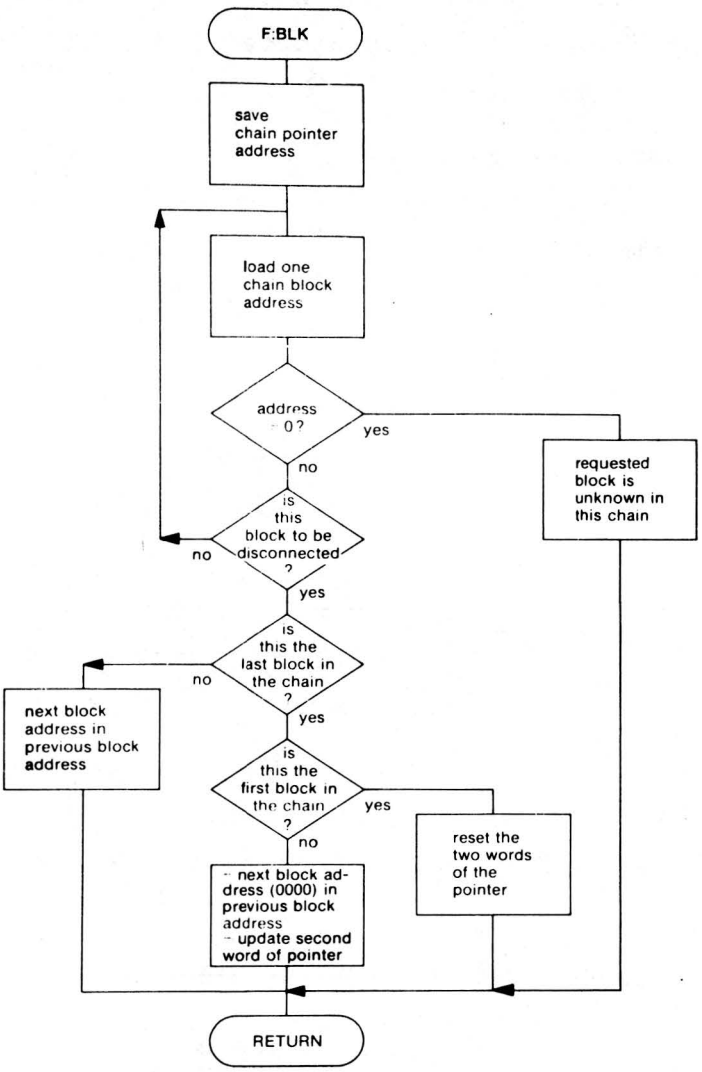
H:POIN: chain pointer

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 looked up and the chain pointer updated.
At the end of the routine A1 will contain one of these values:
A1 $\neq$ -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.


F:BLK
save chain pointer address
load one chain block address
address = 0?
yes
no
requested block is unknown in this chain
is this block to be disconnected ?
no
yes
is this the last block in the chain ?
no
yes
next block address in previous block address
is this the first block in the chain ?
yes
no
reset the two words of the pointer
- next block address (0000) in previous block address
- update second word of pointer
RETURN

M:AOO (NON-WIRED INSTRUCTION SIMULATION ROUTINE)

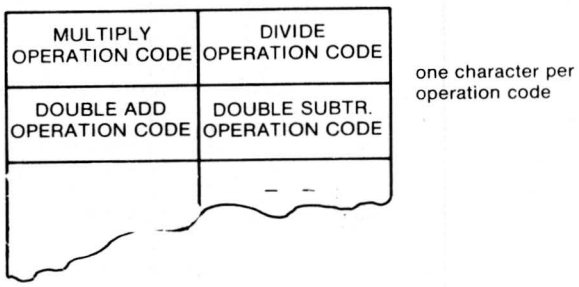
Calling Sequence

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.

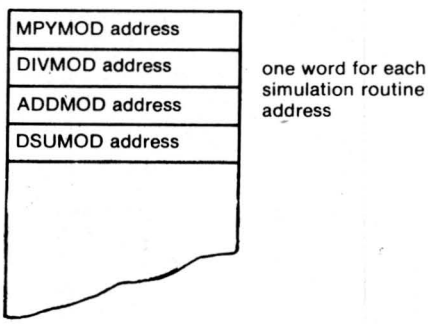
Work Areas and Tables

T:OPC (non-wired operation code table):

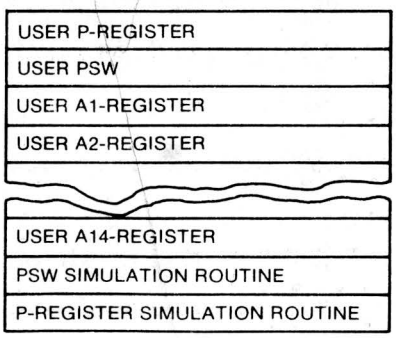


The scanning index is initialized by a parameter L:TOPC:
L:TOPC EQU 'non-wired op.code number - 1'

T:SRA (table of addresses of non-wired op.code simulation routines):

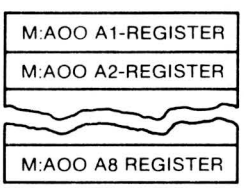


T:SVR (user information safeguard table)



Each entry (18 words) in this table contains P-register, PSW and registers A1 to A14 of the user program, plus the P-register and PSW 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.

Z:SVR (M:AOO routine register save area)
The length of this area is 8 words:



Input/Output Files

None.

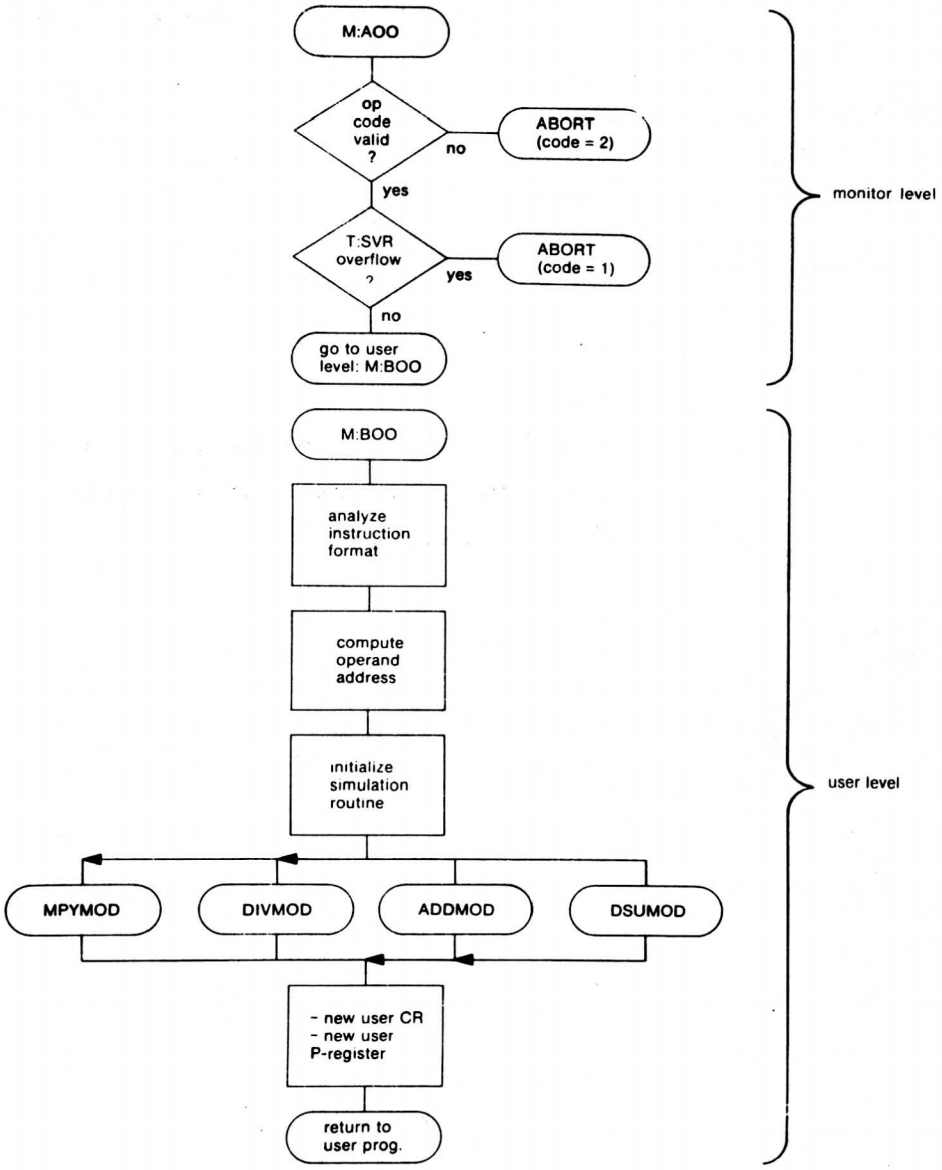
Functional Description

First the M:AOO 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 surrender (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:A00 gives control to the M:B00 which operates at user level. This routine looks for the user operand and initalizes 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.


M:AOO
op code valid ?
no
ABORT (code = 2)
yes
T:SVR overflow ?
yes
ABORT (code = 1)
no
go to user level: M:BOO
monitor level
M:BOO
analyze instruction format
compute operand address
initialize simulation routine
MPYMOD
DIVMOD
ADDMOD
DSUMOD
- new user CR
- new user P-register
return to user prog.
user level

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

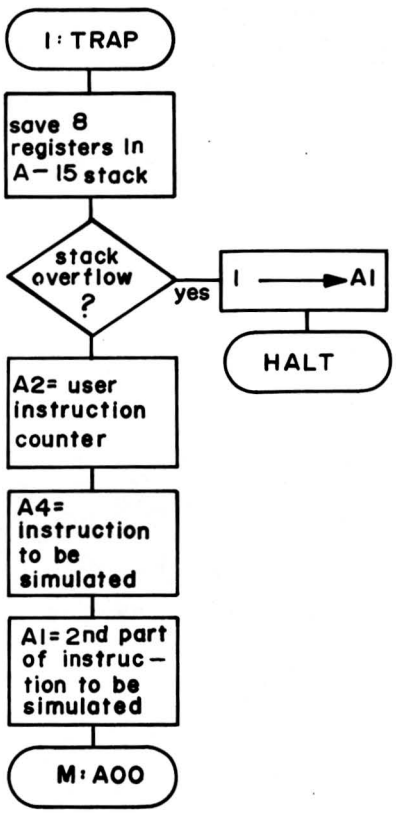
A-15 stack.

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 M:A00 routine, which will simulate the illegal instruction.


I: TRAP
save 8 registers in A-15 stack
stack overflow ?
yes
I → AI
HALT
A2= user instruction counter
A4= instruction to be simulated
AI= 2nd part of instruc-tion to be simulated
M: A00

Comment Sheet

P800M Programmer's Guide 2, Volume IV (5122 991 27402)

Name _____

Company _____

Address _____

Telephone Number _____ Ext. _____

Comments or suggestions

PHILIPS DATA SYSTEMS B.V.

MARKETING GROUP SMALL COMPUTERS
P.O. Box 245, Apeldoorn, The Netherlands
Phone: 055-230123; telex: 49142

For further details contact the above address or:

EUROPE

Sweden
Svenska AB Philips
Data Systems
Minidatorer
Rissneleden 16
Fack
172 07 Sundbyberg
Tel. 08 830300

Denmark
Philips Data Systems A/S
Prags Boulevard 80
2300 København S
Tel. 0127 2222

Norway
Norsk A/S Philips
Data Systems Division
Nils Hansens vei 2
P.O. Box 5040
Oslo 6
Tel. 02 679380

Finland
OY Philips AB
Department Data Systems
Kaivokatu 8
P.O. Box 10255
Helsinki 10
Tel. 90 17271

Belgium
Philips Data Systems SA
Marketing Group Small Computers
Anspachlaan 1
1000 Brussel
Tel. 02 2193900

France
Philips Data Systems
Département Mini-ordinateurs
5 Square Max-Hymans
75015 Paris 15
Tel. 01 734 7759

Western Germany
Philips GmbH-Eiserfeld
Bereich Prozessrechner
Münsterstrasse 330
4 Düsseldorf 30
Tel. 0211 632087

Höhenstrasse 17
7012 Fellbach bei Stuttgart
Tel. 0711 523086
523088

Austria
Österreichische Philips GmbH
Industrie Elektronik
Breitenfurterstrasse 219
1230 Wien
Tel. 0222 831501

Italy
Philips S.p.A.
Sezione S and I
Viale Elvezia 2
20052 Monza
Tel. 039 361441

Switzerland
Philips AG
Data Systems
Binzstrasse 18
8027 Zürich
Tel. 01 442211

Great Britain
Philips Data Systems
Elektra House
2 Bergholt Road
Colchester
C04-5AA Essex
Tel. 206 5115

The Netherlands
Philips Data Systems B.V.
Bordewijkstraat 4
Rijswijk (Z-H)
Tel. 070 906720

Spain
Philips Ibérica S.A.E.
Grupo Instrumentación
Martinez Villergas 2
Madrid 27
Tel. 091 4042200

FAR EAST

Japan
Nihon Philips Corporation
P.O. Box 13
World Trade Centre
Hamamatsu-cho, Minato-ku
Tokyo 105
Tel. 03 435 5211

NORTH AMERICA

U.S.A.
North American Philips Corp.
Dept. 007
100 East 42nd Street
New York N.Y. 10017
Tel. 212 697 3600

Canada
Philips Electronics Industries Ltd.
Telecommunication Division
1001, Ellesmere Road
Scarborough 706
Ontario
Tel. 416 7521980