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

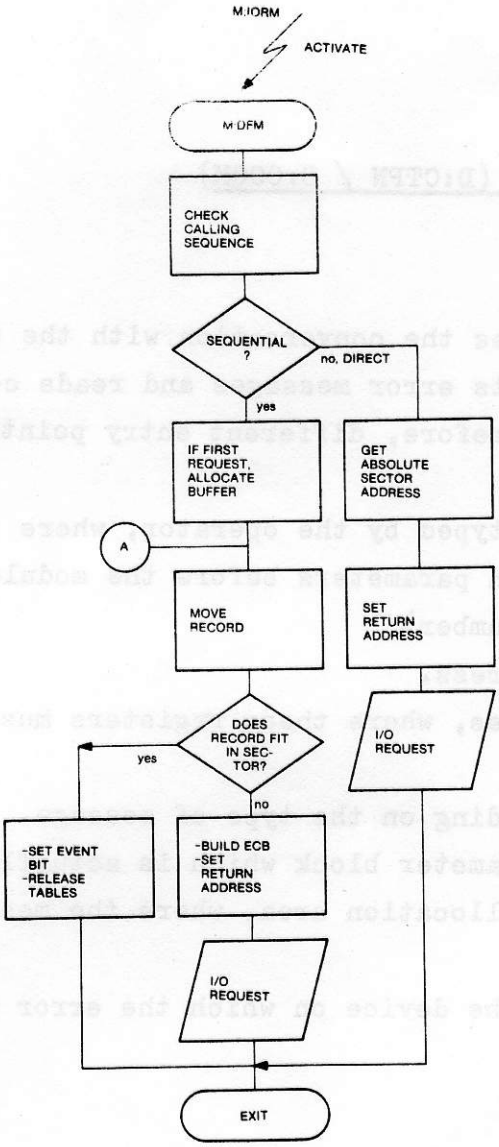
Disc File Management uses the disc file codes /F0 to /FF to perform physical I/O operations.

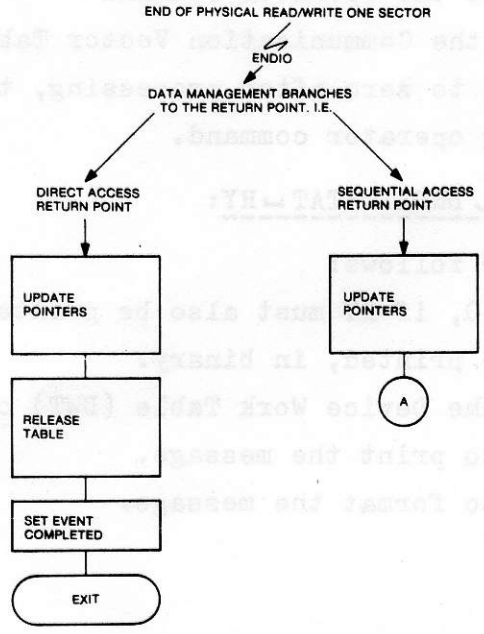
Memory Layout

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?
I/O REQUEST
yes
no
-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.

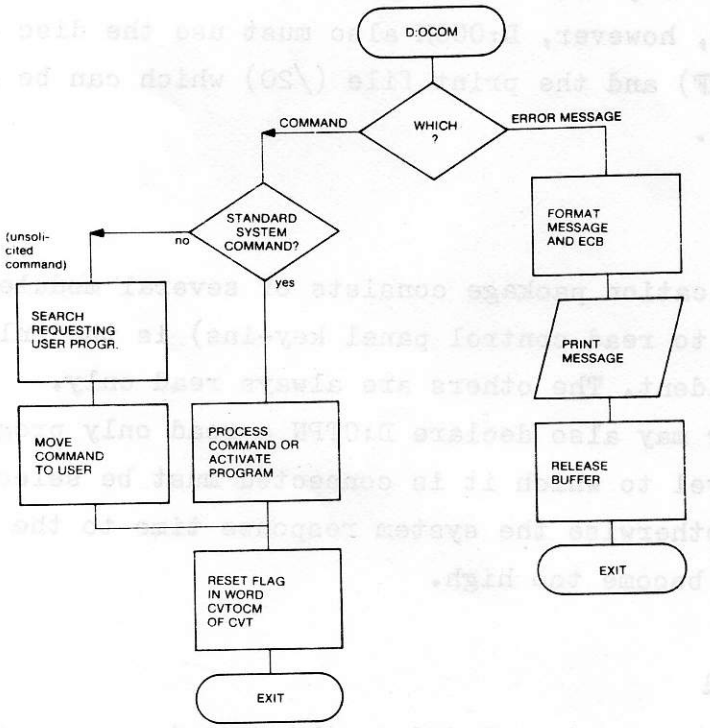
Memory Layout

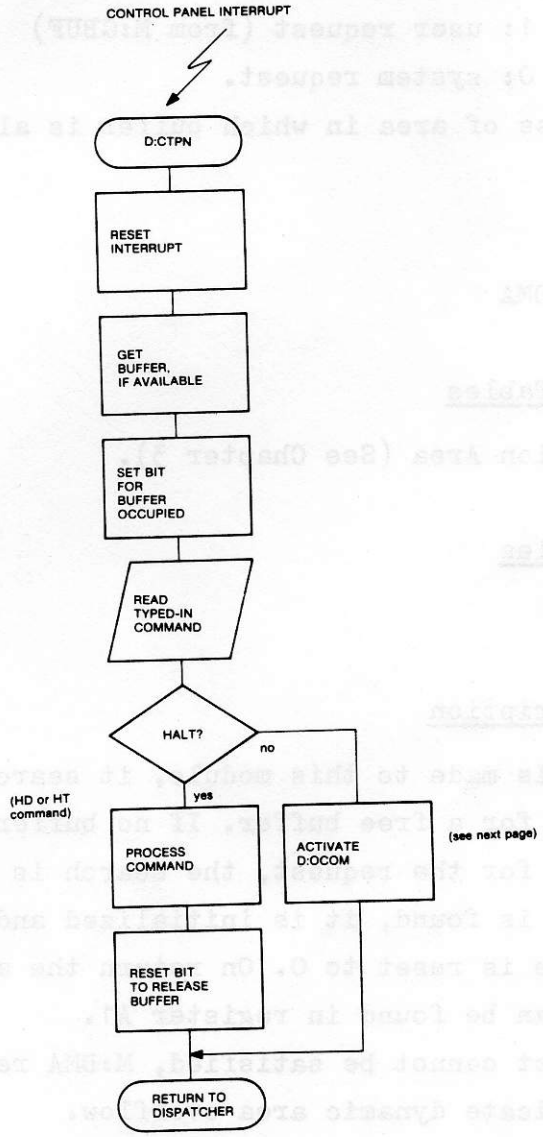
The operator communication package consists of several modules. The D:CTPN module (used to read control panel key-ins) is the only one which is memory resident. The others are always read only.
If desired, the user may also declare D:CTPN a read only program, but the priority level to which it is connected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

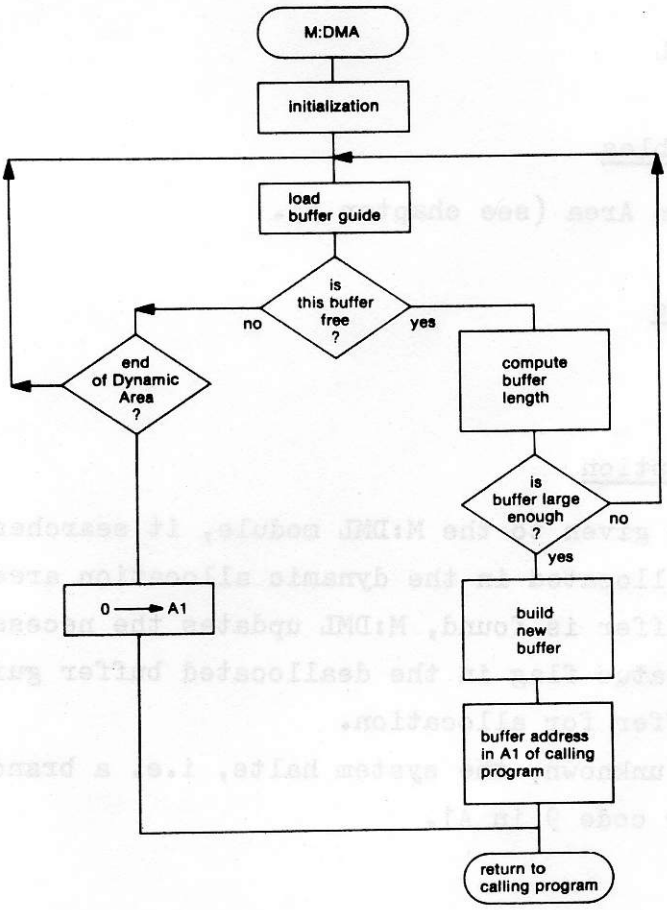
Dynamic Allocation Area (See Chapter 3).

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

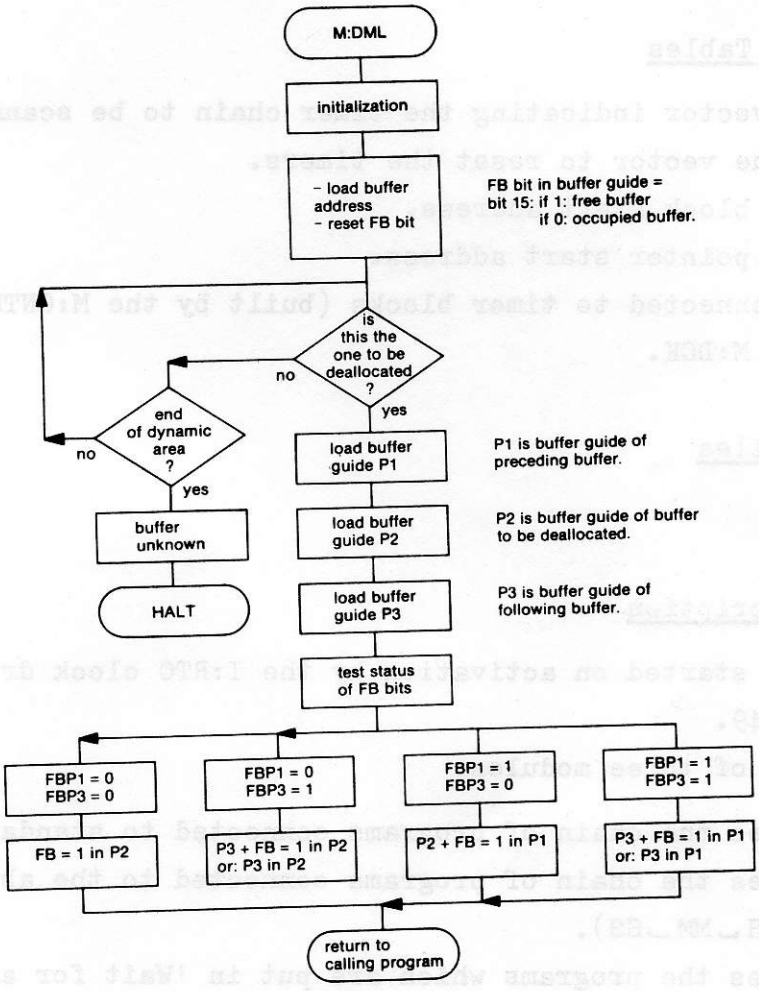
Dynamic Allocation Area (see chapter 3).

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
buffer unknown
HALT
load buffer guide P1
P1 is buffer guide of preceding buffer.
load buffer guide P2
P2 is buffer guide of buffer to be deallocated.
load buffer guide P3
P3 is buffer guide of following buffer.
test status of FB bits
FBP1 = 0
FBP3 = 0
FB = 1 in P2
FBP1 = 0
FBP3 = 1
P3 + FB = 1 in P2
or: P3 in P2
FBP1 = 1
FBP3 = 0
P2 + FB = 1 in P1
FBP1 = 1
FBP3 = 1
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.

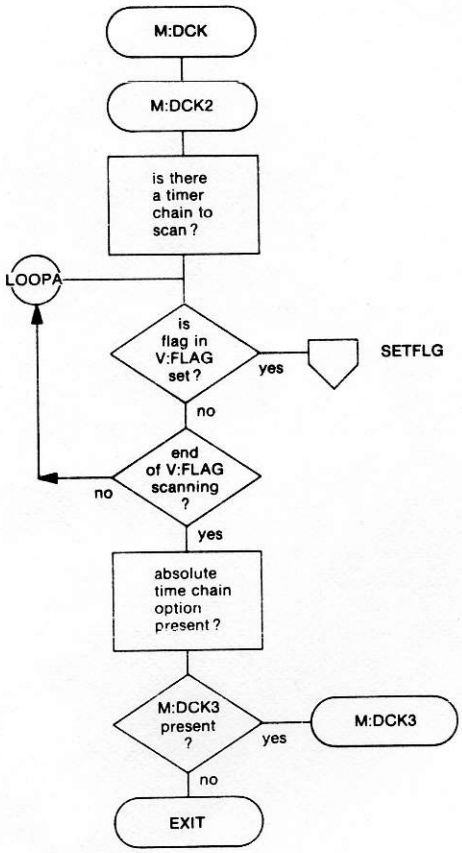
Functional Description

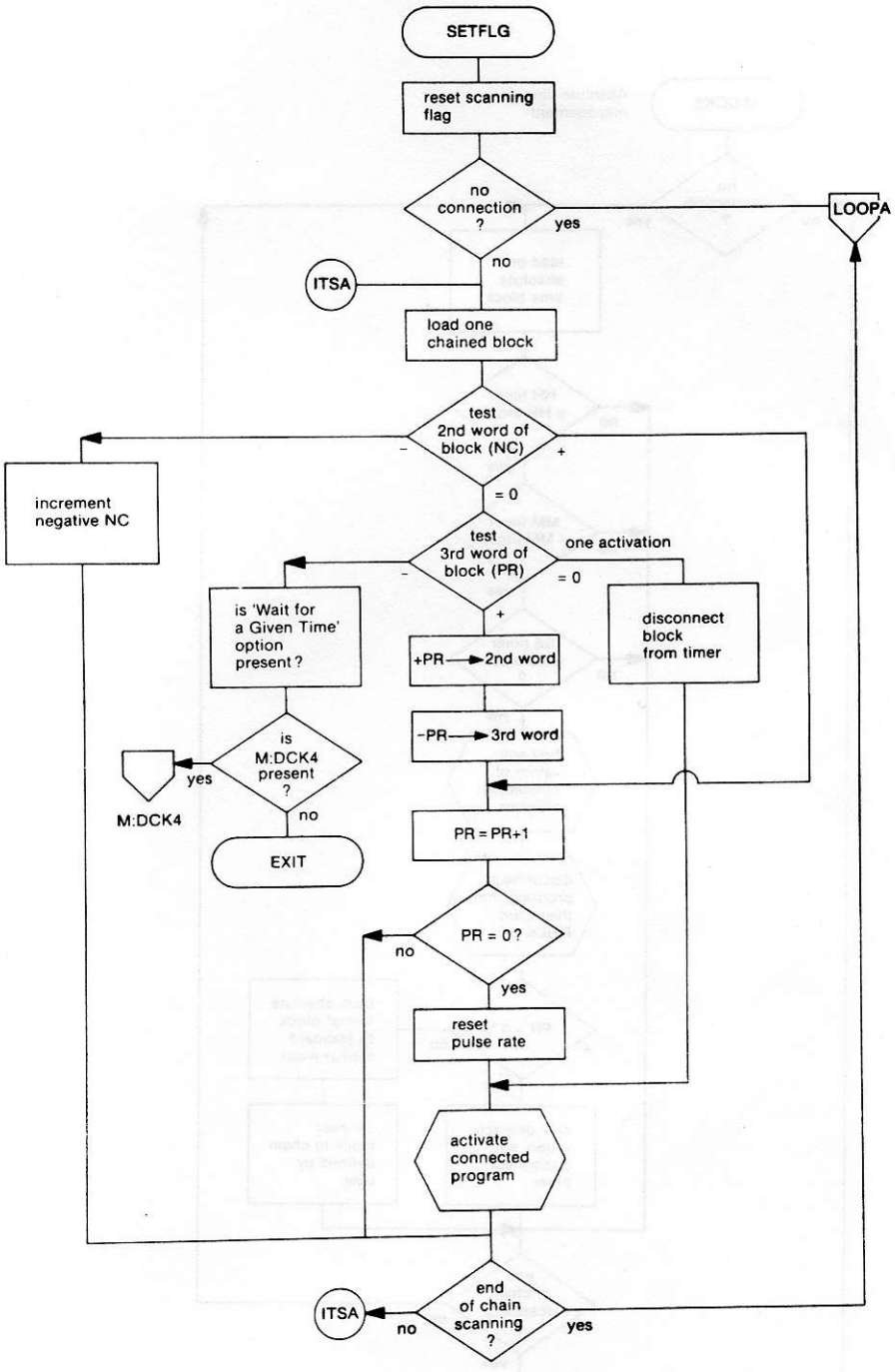
This module is started on activation by the I:RTC clock driver and runs at level 49.
It is composed of three modules:

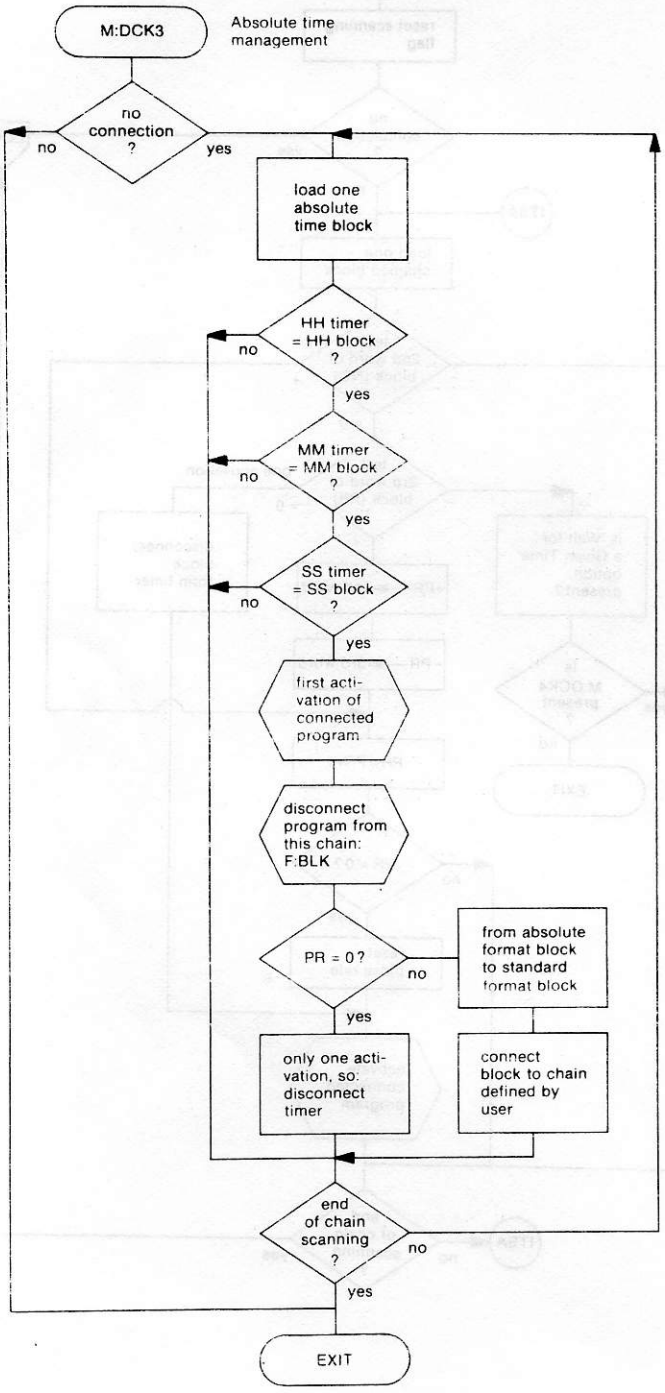
- M:DCK2 manages the chain of programs connected to standard timers.
- M:DCK3 manages the chain of programs connected to the absolute time (NC - HH␣MM␣SS).
- M:DCK4 manages the programs which are put in 'Wait for a Given Time'.

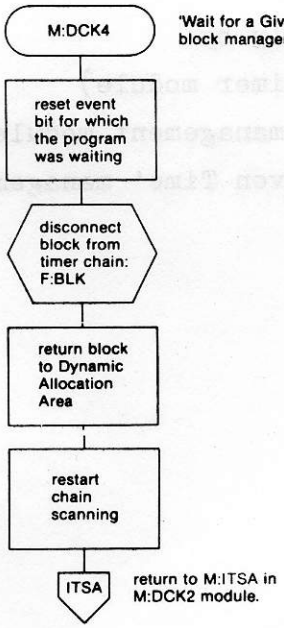
M:DCK2 first checks V:FLAG (initialized by I:RTC) to find out if a chain of programs connected to a timer must be analyzed.
At the end of M:DCK2, or in case no programs are connected to the scanned timer chains, the absolute time option is asked for. If it is present, M:DCK3 is started, otherwise M:DCK2 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
-PR → 3rd word
is M:DCK4 present ?
yes
M:DCK4
no
EXIT
PR = PR+1
PR = 0 ?
no
yes
reset pulse rate
activate connected program
ITSA
no
end of chain scanning ?
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
from absolute format block to standard format block
yes
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

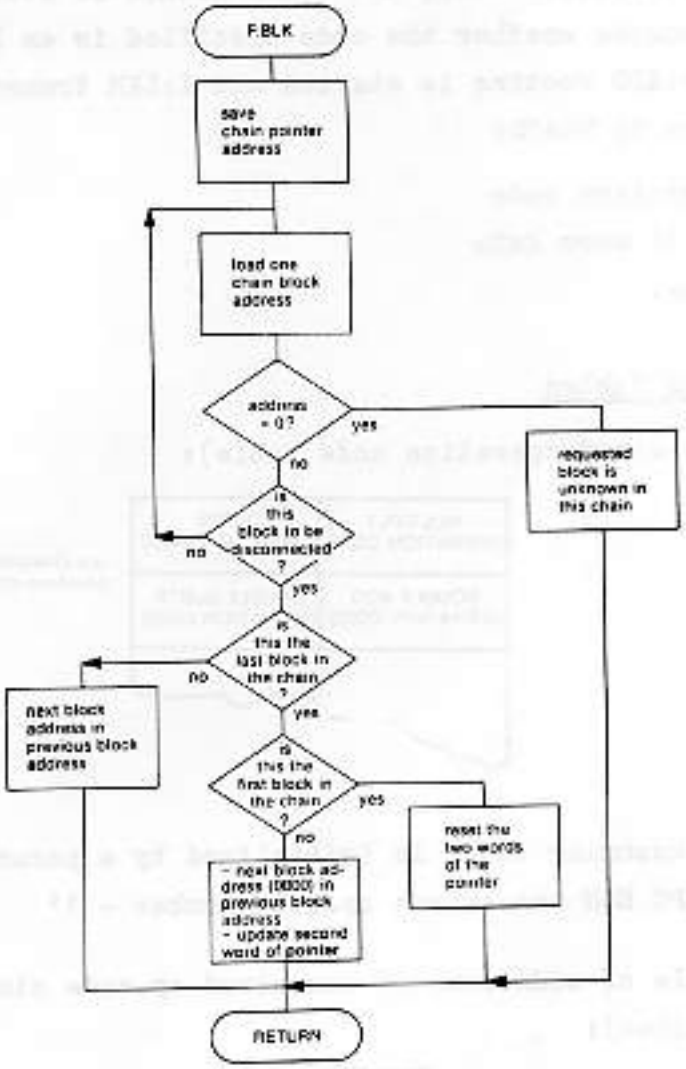
H:POIN: chain pointer

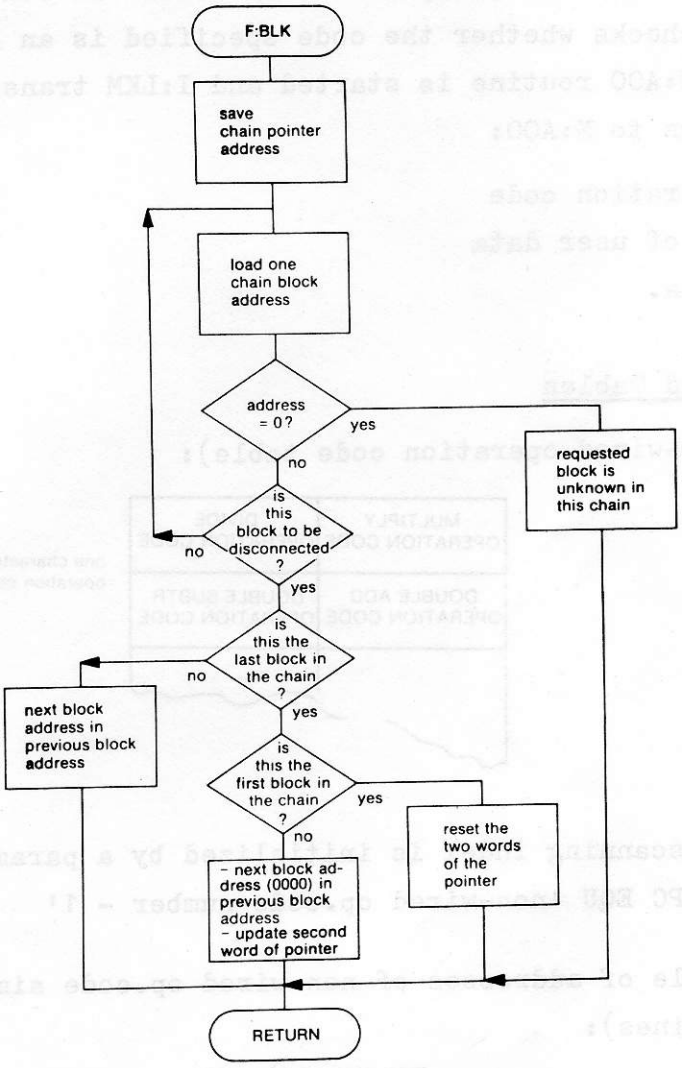
Input/Output Files

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 ≠ -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


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:A00 (NON-WIRED INSTRUCTION SIMULATION ROUTINE)

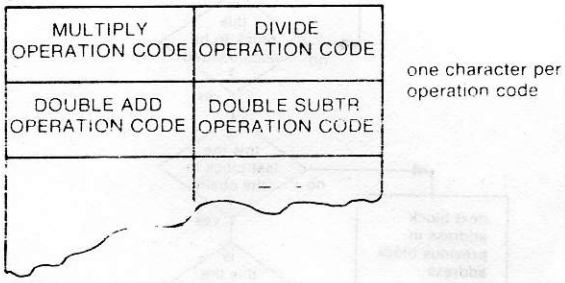
Calling Sequence

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:A00 routine is started and I:LKM transfers the following parameters to M:A00:

A1: user operation code
A2: address of user data
A3: user data.

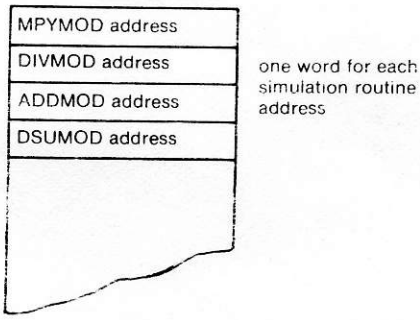
Work Areas and Tables

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

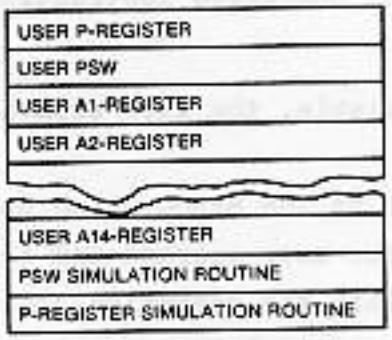


The scanning index is initialized by a parameter L:TOPC:
L:TOPC EQU 'non-wired op.code number - 1'

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

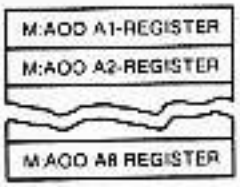


T:SVR (user information safeguard table)



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.

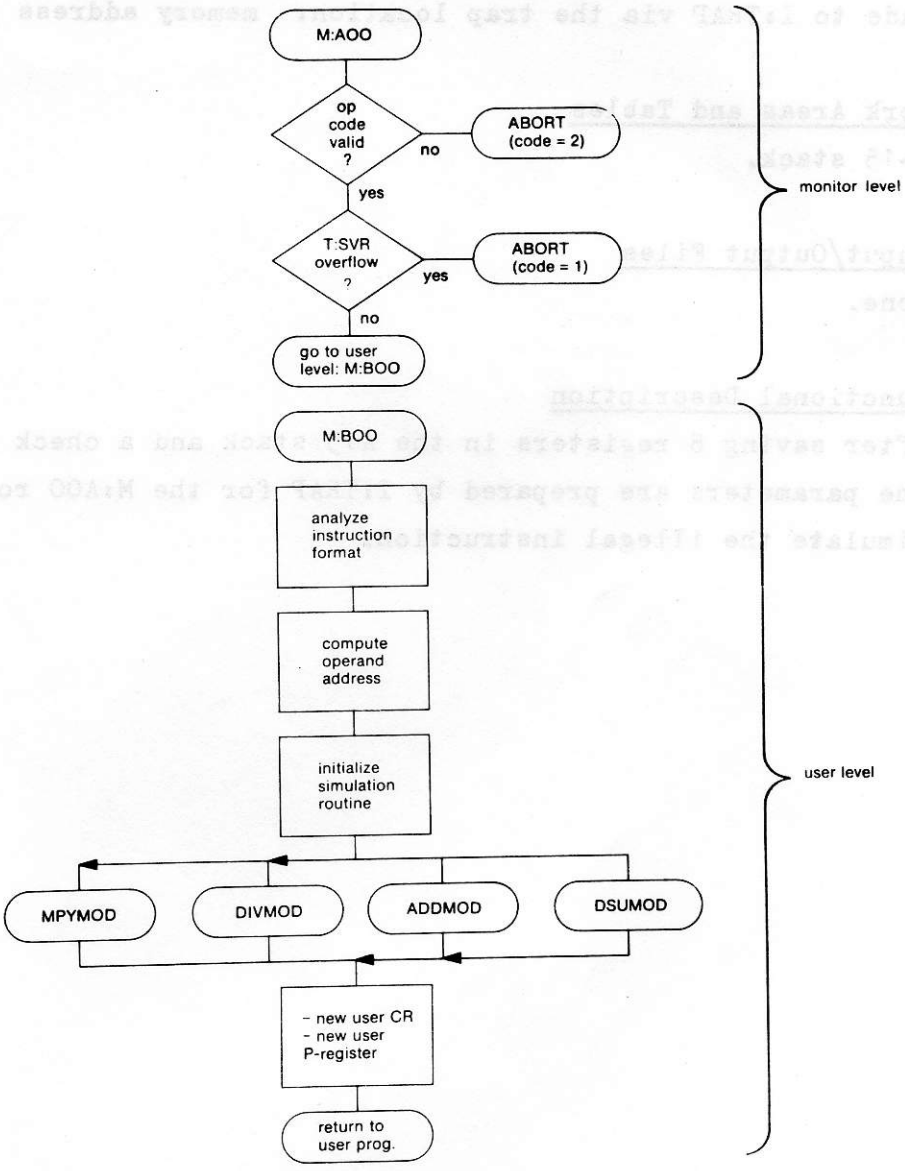
Functional Description

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

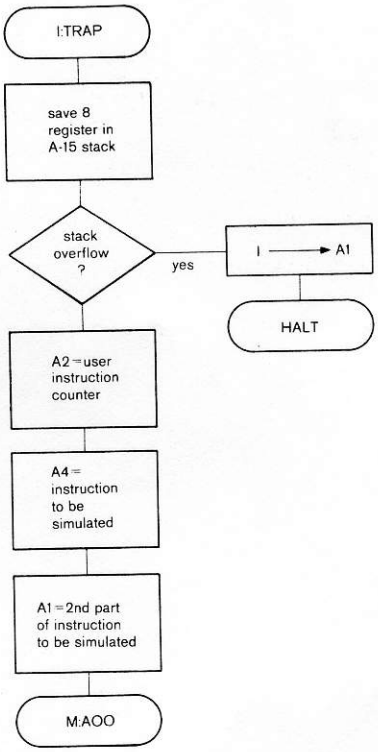
A-15 stack.

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 register in A-15 stack
stack overflow ?
yes
I → A1
HALT
A2 = user instruction counter
A4 = instruction to be simulated
A1 = 2nd part of instruction to be simulated
M:AOO