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

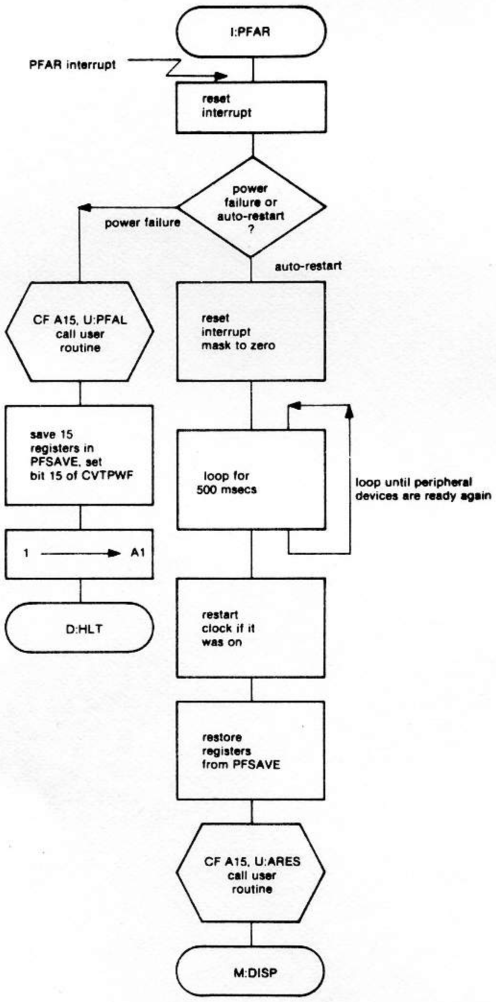
The three optional routines mentioned in the flowchart for this module:
U:PFAL
U:ARES
U:RST
must be provided by the user. If they are not used, they must be simulated with an RTN A15 instruction.

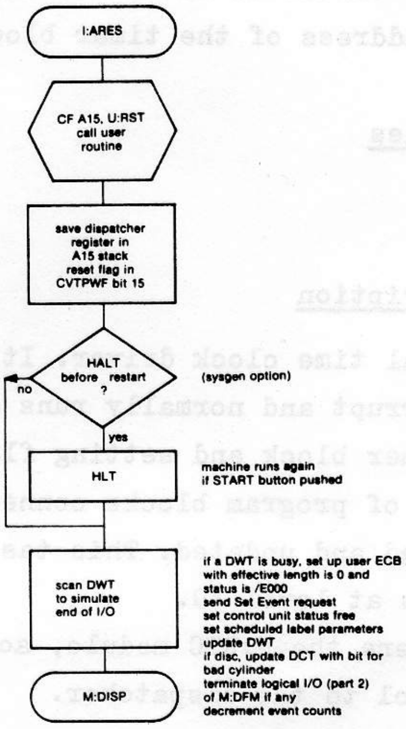
- When a power failure interrupt has been recognized and the interrupt has been reset, the system calls the user power failure routine, before saving the 15 registers:
  CF A15, U:PFAL
- When the automatic restart interrupt has been recognized, the system restores the 15 registers and the user automatic 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
save 15 registers in PFSAVE, set bit 15 of CVTPWF
1 → A1
D:HLT
reset interrupt mask to zero
loop for 500 msecs
loop until peripheral devices are ready again
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

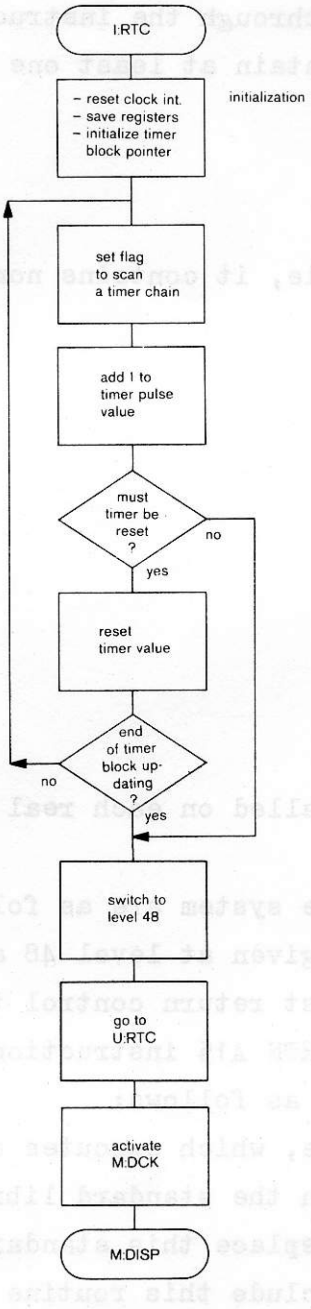
I:CPLS:	a word in the Communication Vector Table giving the pulse rate.
V:FLAG:	a flag vector indicating the timer to be scanned.
V:REST:	a value vector to reset the timers.
H:TIME:	start address of the timer block.

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 updating ?
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

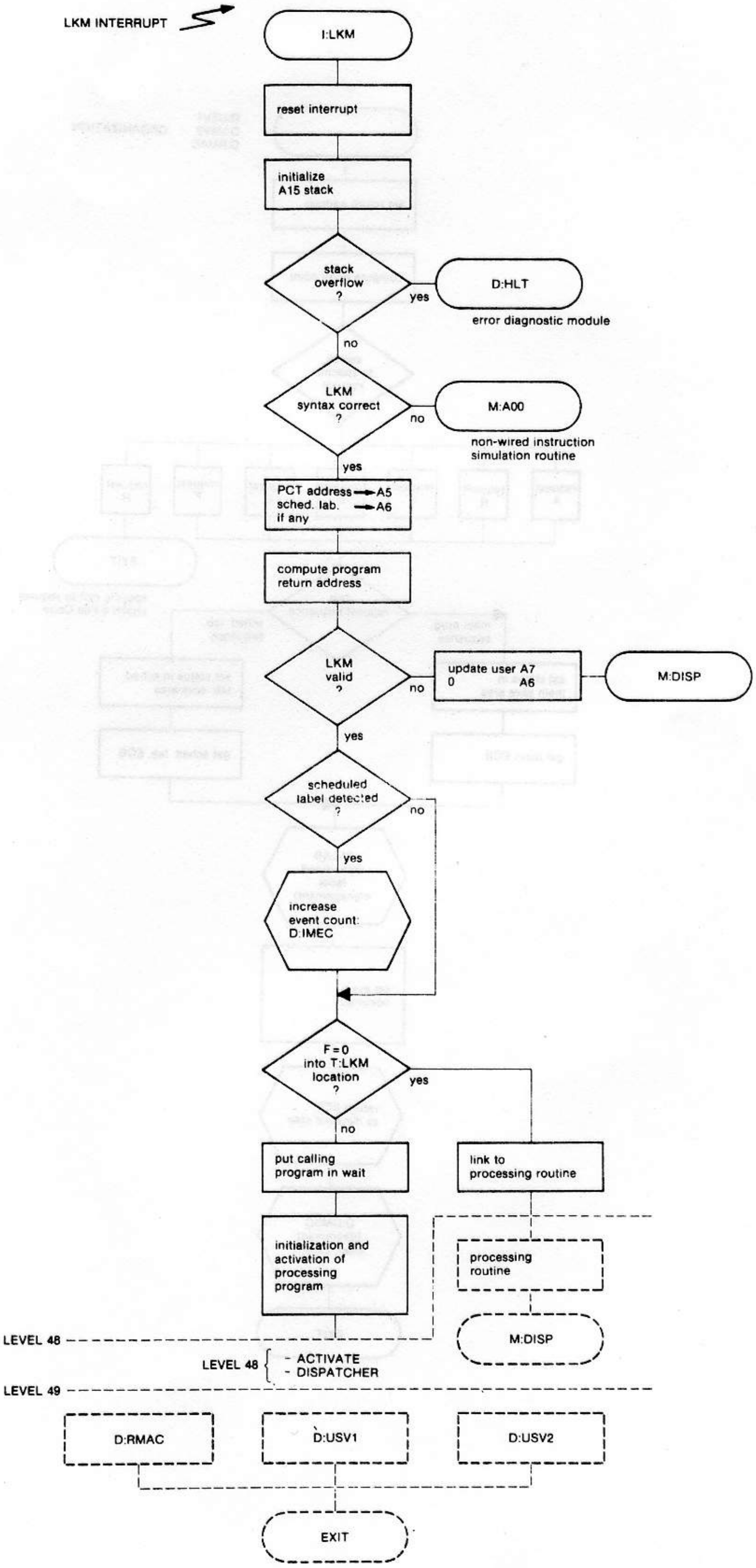
A7: User A7 parameter

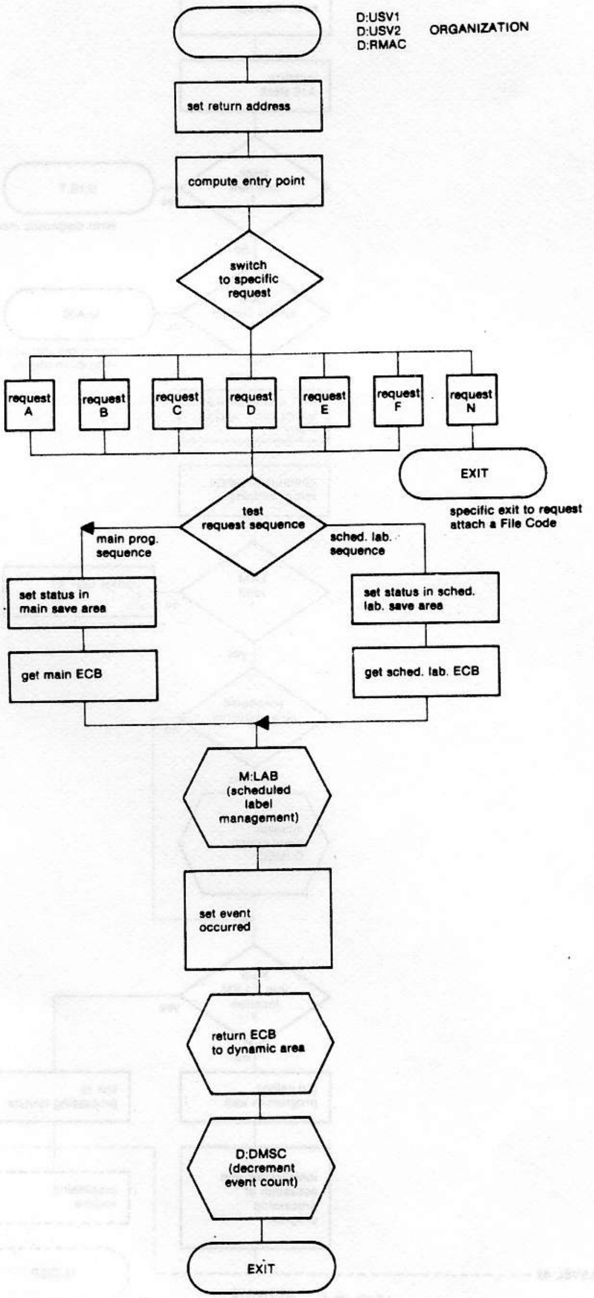
A8: User A8 paramerer.

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
get main ECB
set status in sched. lab. save area
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.

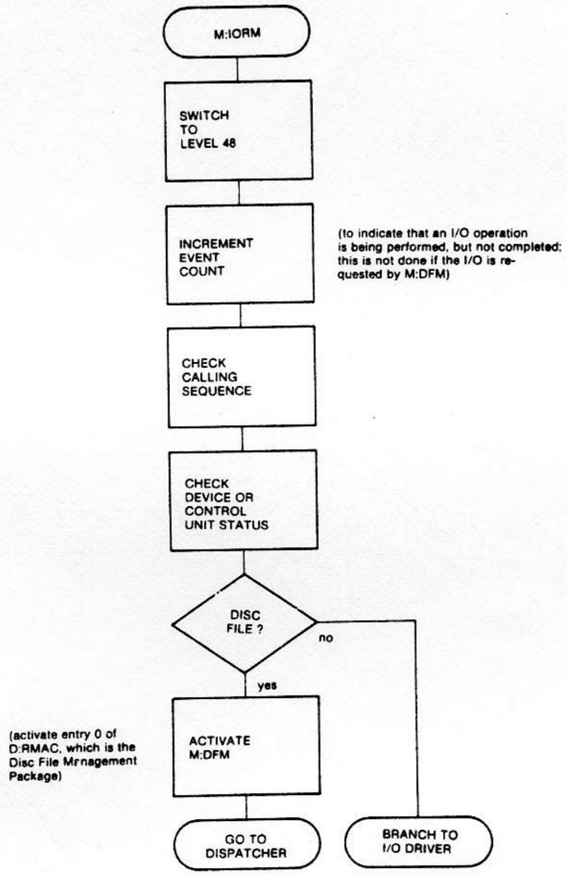
Functional Description

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

- pre-processing
- processing
- post-processing

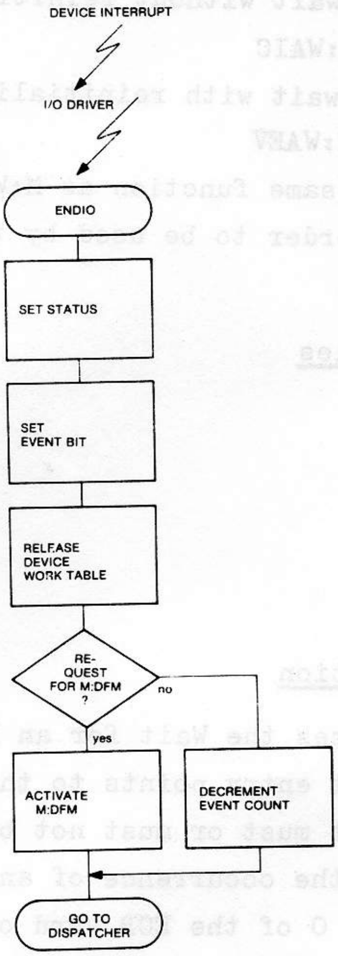
Pre-processing is done by the M:IORM module by analyzing the I/O requests, checking for device availability and verifying the validity of the user parameters.

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.



(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)

(activate entry 0 of D:RMAC, which is the Disc File Mrnagement Package)


DEVICE INTERRUPT
I/O DRIVER
ENDIO
SET STATUS
SET
EVENT BIT
RELEASE
DEVICE
WORK TABLE
RE-
QUEST
FOR M:DFM
?
no
yes
ACTIVATE
M:DFM
(entry point 2
of D:RMAC)
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 in order to be used by the read-only system programs)

Work Areas and Tables

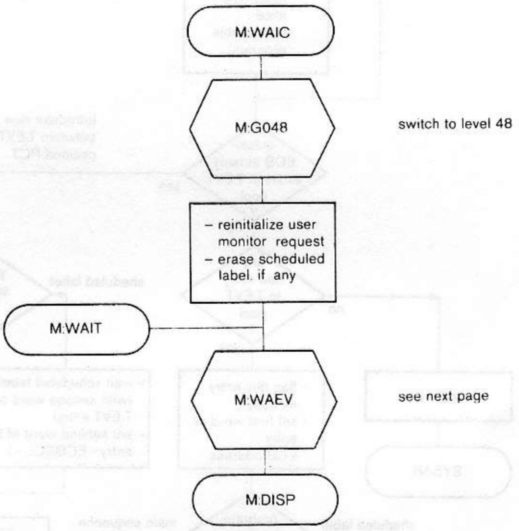
None.

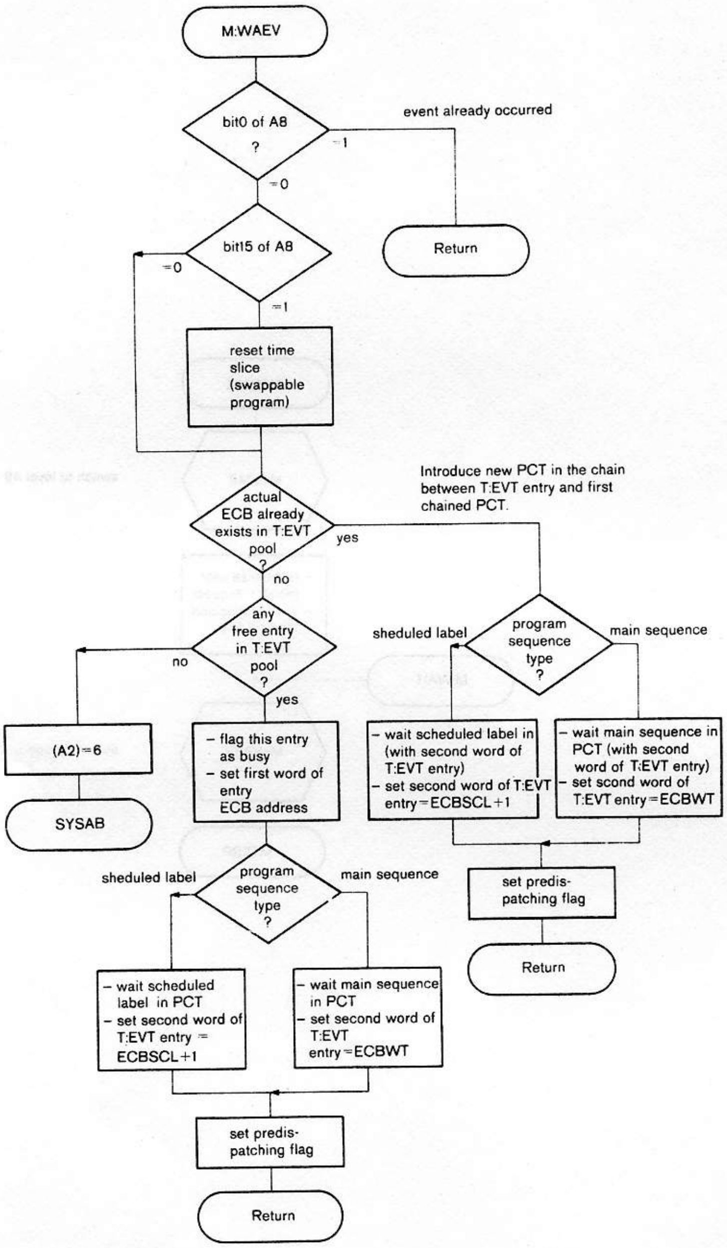
Input/Output Files

None.

Functional Description

This module processes the Wait for an Event monitor request (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:G048
switch to level 48
– reinitialize user monitor request
– erase scheduled label, if any
M:WAIT
M:WAEV
see next page
M:DISP


M:WAEV
bit0 of A8 ?
=1
event already occurred
Return
=0
bit15 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
(A2)=6
SYSAB
yes
– flag this entry as busy
– set first word of entry ECB address
program sequence type ?
sheduled 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 predis-patching flag
Return
program sequence type ?
sheduled label
main sequence
– wait scheduled label in (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 scond word of T:EVT entry=ECBWT
set predis-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

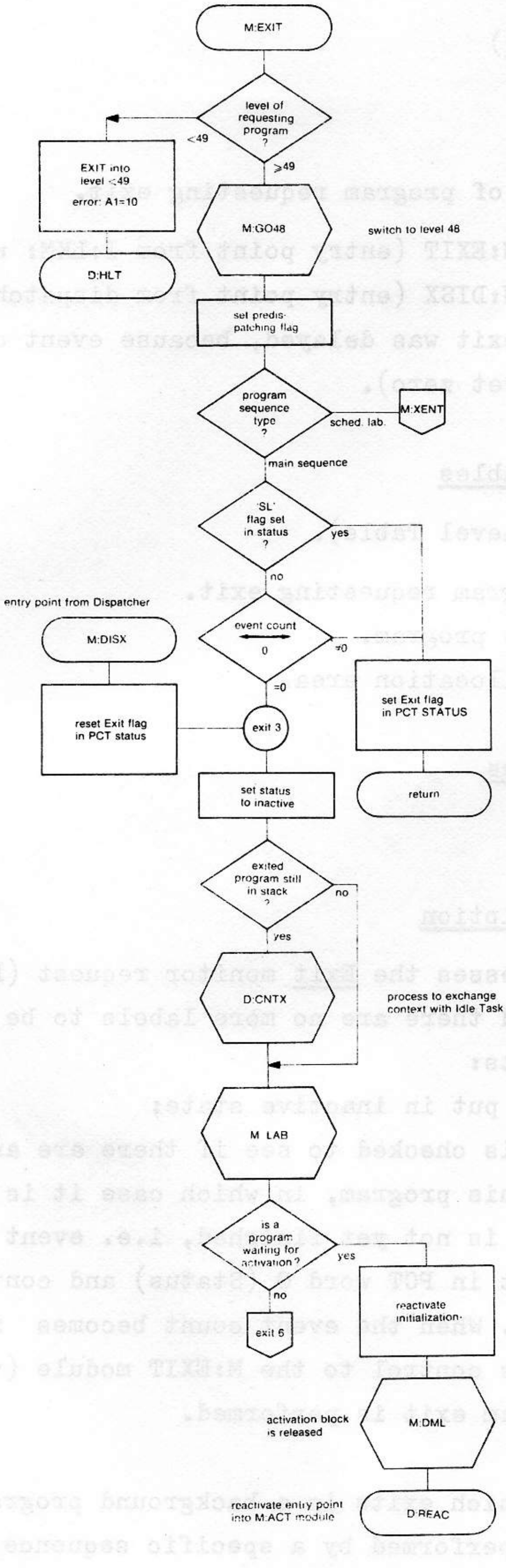
None.

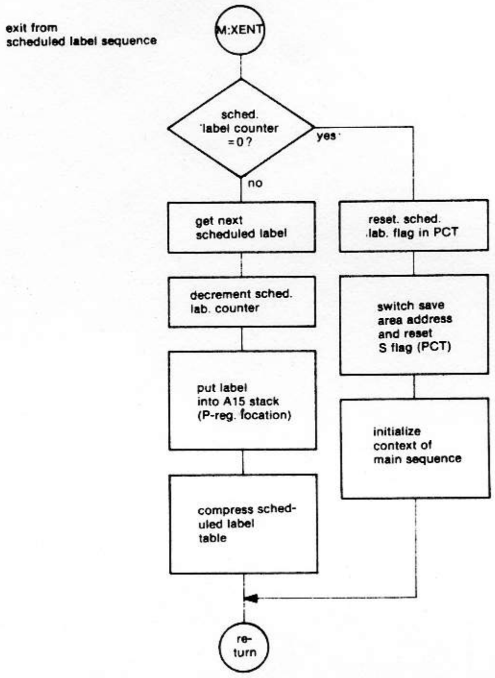
Functional Description

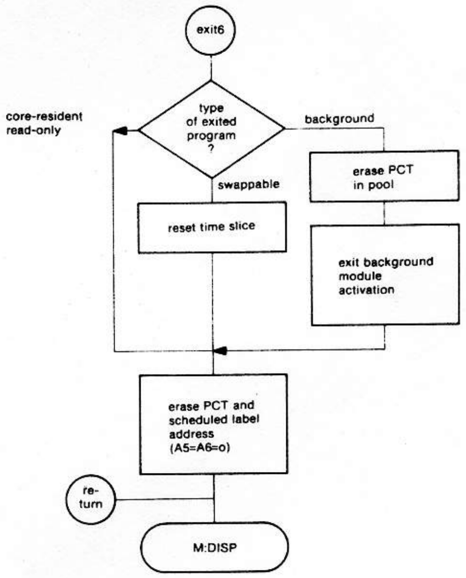
This module processes the Exit monitor request (LKM3). If all I/O is terminated and there are no more labels to be scheduled, the main program exits:

- 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 ≠ 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:EBCX) 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 predis-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 scheduled 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
type of exited program ?
core-resident read-only
background
swappable
reset time slice
erase PCT in pool
exit background module activation
erase PCT and scheduled label address (A5=A6=o)
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

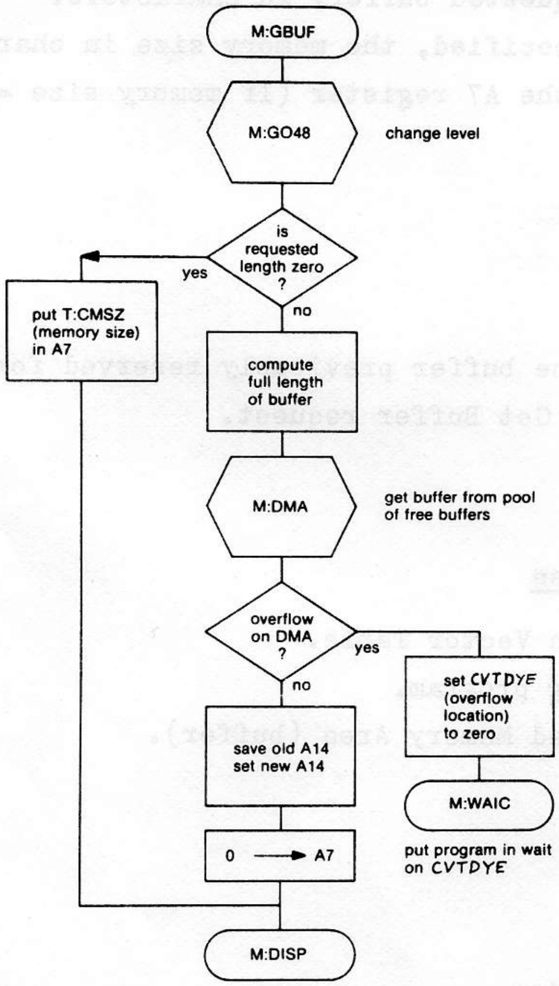
T:CVT: Communication Vector Table.
PCT entry of calling program.
Dynamically Allocated Memory Area (buffer).

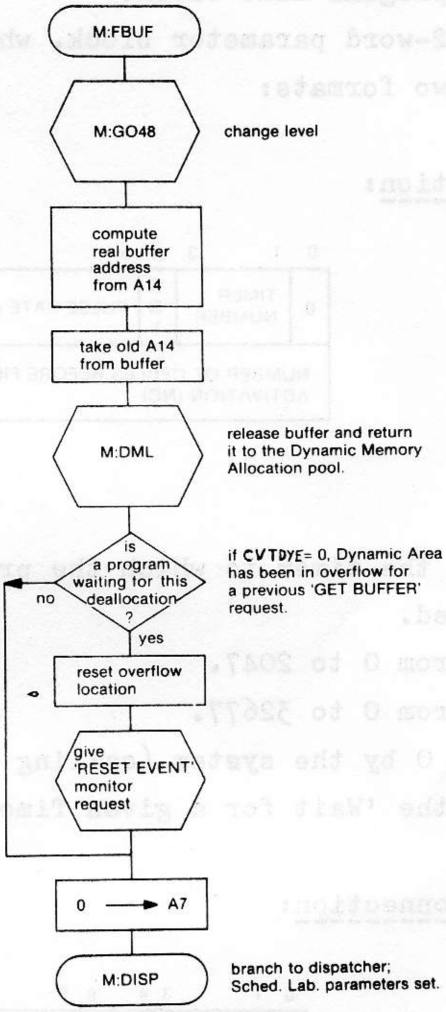
Input/Output Files

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
compute real buffer address from A14
take old A14 from buffer
M:DML
is a program waiting for this deallocation ?
no
yes
reset overflow location
give 'RESET EVENT' monitor request
0 → A7
M:DISP

change level

release buffer and return it to the Dynamic Memory Allocation pool.

if CVTDYE= 0, Dynamic Area has been in overflow for a previous 'GET BUFFER' request.

branch to dispatcher; Sched. Lab. parameters set.