part 2.

P855/860

DISC REAL TIME MONITOR DRTM

PRELIMINARY

CONFIDENTIAL

June/72

Introduction

The Disc Real Time Monitor has been designed to supervise a multi-tasking ; foreground/background system using discs for program storage.

The memory is divided into two main areas: the Foreground Area and the Background Area.
In the background, programs can be executed or prepared - i.e. assembled, compiled, link-edited, etc.- for future use in the foreground.
In the foreground, programs are executed within a multilevel 13 levels priority system, based on priority interrupts.
Hardware interrupts are connected to the highest levels and therefore have priority over software levels, to which user programs can be connected. The user can fix these priority levels as he wishes, except for some hardware interrupts which have fixed priority levels.
Background: 1 level.
This system provides for the following real-time facilities:

- multi-tasking
- time-slicing
- program activation under timer control
- reentrant routines
- Storage of programs on disc
- dynamic loading of disc-resident programs. | typical for disk
- Swapping

For these purposes, the foreground area of memory can be divided into several partitions, for different functions.
Communication between programs with different functions in different partitions is possible because, if the memory protect option is included, they are protected against writing only - any program can read data located in any partition.
Programs must be connected to a level but, in addition, the possibility exists to connect them to a Real Time Clock or to a timer.
Programs on disc can be loaded dynamically into a special partition.
If necessary, such a program can be swapped out of memory back onto disc, thus releasing memory space for a program of higher priority.

The minimum configuration for this system is

- P855 or P860 computer with 8k words of memory.
- operator's typewriter with punched tape equipment.
- one fixed-head disc unit.

GENERAL CONCEPTS

Priority System

The priority system is based on a 64-level structure.
These priority levels are subdivided as follows:
- levels 0 to 47 are hardware interrupt lines
- levels 48 to 63 are software priority levels, assigned as follows:
  - 48 and 49 are standard system levels of the monitor
  - 50 to 62 are levels for foreground user programs
  - 63 is reserved for programs running in the background.

The higher the level number, the lower its priority, i.e. level 0 has the highest, 63 the lowest priority. From this it follows that hardware interrupts always have priority over software levels.

Hardware Interrupt Lines

To these lines (memory locations 32 through 63 and 96 through 111) routines are connected which are required to service internal or external hardware interrupts, such as from power failure, real time clock, peripheral devices, etc.
These routines, connected to any of the levels 0 through 47, will, when called through an interrupt, run on the priority level associated to that interrupt. Only a higher priority interrupt will be able to overrule the running one.
Some of these hardware interrupts have fixed priority levels in the standard monitors delivered with the systems. If the user wishes to fix these priority levels in another way, he must change the monitor.

0: power failure detection
1: LKM (monitor request);
   stack overflow;
   illegal code used
2: real time clock
3: memory protect;
   privileged instruction used
4: control panel

Moreover, if any of the following peripherals are included in a configuration,

the following interrupt levels are standard for them:

5: cassette tape
6: punched tape reader
7: tape punch
8: operator's typewriter
9: disc
10: line printer
11: card reader
12: magnetic tape.

The other levels can be connected by the user as he wishes.

When an interrupt occurs, two results are possible:

- if the program being executed at the time of the interrupt is of a higher priority, the interrupt is not accepted but must wait until its priority level can be serviced, i.e. until it becomes the highest level requesting central processor time.
- if the program being executed at the time of the interrupt is of a lower priority, the interrupt is accepted, i.e. control is transferred from the running program to the program required to service the interrupt (the interrupt routine).

If the interrupt is accepted, the running program is stopped at the end of the current instruction and control is transferred to the interrupt routine after register A0 and the Program Status Word (PSW), have been stored by hardware in the stack. As it is mandatory for the interrupt routine to use only the registers A1 through A8, it must itself take care of saving the contents of these registers in the stack as well as any other parameters of the interrupted program that need saving.
This stack is handled automatically by the monitor, using A15 as the stack pointer. Therefore it is forbidden to the user to use register A15 in his instructions.

Note: One of the interrupt lines is a common line, able to accept up to 16 signals. This line can be connected to any of the hardware interrupt locations and the signals on the line can be inhibited or enabled by means of

a mask register, written in a WMM instruction.
However, when this common line is connected to level 0, it is advised to write only a half mask (right-hand character), because in that case the fixed interrupts mentioned above will be placed on the signals of this line in the same oreder of priority, i.e bit 0 corresponding to level 0, bit 1 to level 1, etc. The important internal interrupts will then run through the left-hand character of the mask and may be better left unmasked, i.e. enabled.

Software Priority Levels

This part of the priority system permits multi-tasking between programs. The priority levels 50 to 62 are connected to foreground user tasks. One or more programs may be connected to the same level. These tasks must be activated through monitor requests or operator control statements. It is possible for one task to activate another one.
Level 63 is reserved for programs using the background area.
Requests for activation of a program are handled by the dispatcher, a monitor part (on level 48) which divides the central processor time so as to start programs according to priority. When the dispatcher starts a program, it places the level connected to that program in the Priority Level register (PL). This register consists of 6 bits and, consequently, can hold any number from 0 to 63. The PL-register always contains the level number of the running program or routine. Thus, upom interrupt, the dispatcher can compare the level of the interrupting program with the contents of the PL-register which contains the level of the running program. If the interrupt was made by a program with a higher priority level than the one currently contained in the PL-register, the running program is stopped, the level of the interrupting program is placed in the PL-register and this program is started.

Memory Allocation

The lower part of the memory is reserved for the monitor.
The upper part is available to the user and can be divided into a Foreground Area and a Background Area, where a further subdivision into different partitions is possible in the Foreground Area.

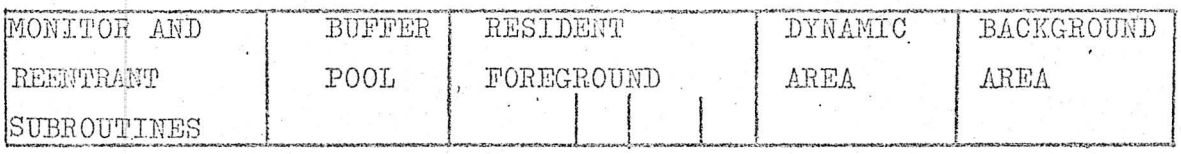
Foreground Area

In this part of the memory the real-time processing takes place.

The foreground can be divided into partitions of four types, each type having a particular function. Up to seven partitions aan be allocated in the foreground. Each partition must be at least 1k words long and, if longer, must consist of contiguous blocks of 1k words. It is possible to define only certain types of partition, not making use of the possibility to have all four types.

Within this partitioned foreground area, the monitor can supervise the execution of up to 13 user tasks (connected to levels 50 to 62) and the interrupt routines (levels 0 to 47).

The following memory partitions can be defined (by operator control message DP):



P0: This partition contains the bootstrap, interrupt locations and monitor. Moreover, the reentrant subroutines, i.e. subroutines which can be called by any program, are also loaded into this area. This must be done at system generation time. It is not possible to add new subroutines dynamically.

P1: This partition can be used by foreground programs to reserve space for buffers ('Get Buffer' monitor requests).

P2 to P5: Up to 4 partitions for resident foreground programs.

P6: A partition where programs one at a time can be loaded dynamically ('Activate' monitor request) from disc to be executed. It is possible to have program swapping in this area.

P7: Background area.

All user programs loaded into the same partition are allowed to work within the boundaries of that partition during a run. All have the same memory protection mask. This protection is against writing only; any program can read data located in any partition, so as to make interprogram communication possible.

Background Area

This area is used independently by the processors and user programs. Its main use is for preparation of programs to be used later in the foreground. This area has a standard priority level, 63, the lowest level.

Programs

As mentioned above, there are two types of programs: interrupt routines (levels 0 to 47) and software level programs (levels 50 to 63).

Interrupt Routines

An interrupt routine is a special program called through an interrupt to take a specific action when a certain event occurs, for example when an optional instruction is used which is not provided for in a particular configuration (illegal code interrupt: a simulation routine is executed) or when the control panel interrupt button is pushed, etc.
An interrupt routine runs with the hardware interrupt level associated to the interrupt signal through which it is called.
Each interrupt is associated to a location in memory which contains the address of the interrupt routine.

If a user writes an interrupt routine, to be connected to one of the interrupt levels, he must connect it to the correct interrupt level location.
This may be done as follows:
For example, a routine labeled INTRO is to be connected to interrupt level 10, so the starting address of the routine has to be loaded into memory location X'54'. This can be done as follows, providing a relocatable routine:

```
        IDENT
          |
          |
        AORG /54
        DATA INTRO
        RORG
          |
          |
INTRO     |   _____
          |  |     |
          |  |     |   } interrupt routine
          |  |_____|
          |
        END
```

Interrupt routines must be resident in memory (partition 0) and be written in master mode, using only the registers A1 through A8. This implies, that, before making any operations, the routine must provide for saving the contents of these registers of the interrupted program in the stack (MSR 8,A15) and restore them at the end of the routine (MSR 8,A15).
The stack is handled automatically by the monitor, with A15 as stack pointer. The user may not operate on this register. Any other registers or parameters which the user wants to save must be stored in a save area, the address of which is given in the Program Control Table (see below).
If this interrupt routine is interrupted, the registers A1 to A8 of the old interrupt program are taken from the stack and stored in the save area of the old program. Then the registers A1 to A8 of the interrupted routine are placed in the stack. At the end of the interrupt routine the reverse process takes place and all registers are restored.

Software Level Programs

These are programs belonging to user tasks connected to any of the levels 50 to 63. They are activated through a monitor request or an operator control message, after which the dispatcher starts them according to their priority.
After loading of a program, all the relevant information about it is stored in a Program Control Table (PCT; see below) and the program remains under control of the monitor until it is terminated. During this time it passes through various states, which are 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, but is not yet terminated.
- wait for execution: the program is ready to use central processor time when it has priority.
- wait for an event: the program has given up control volunt-

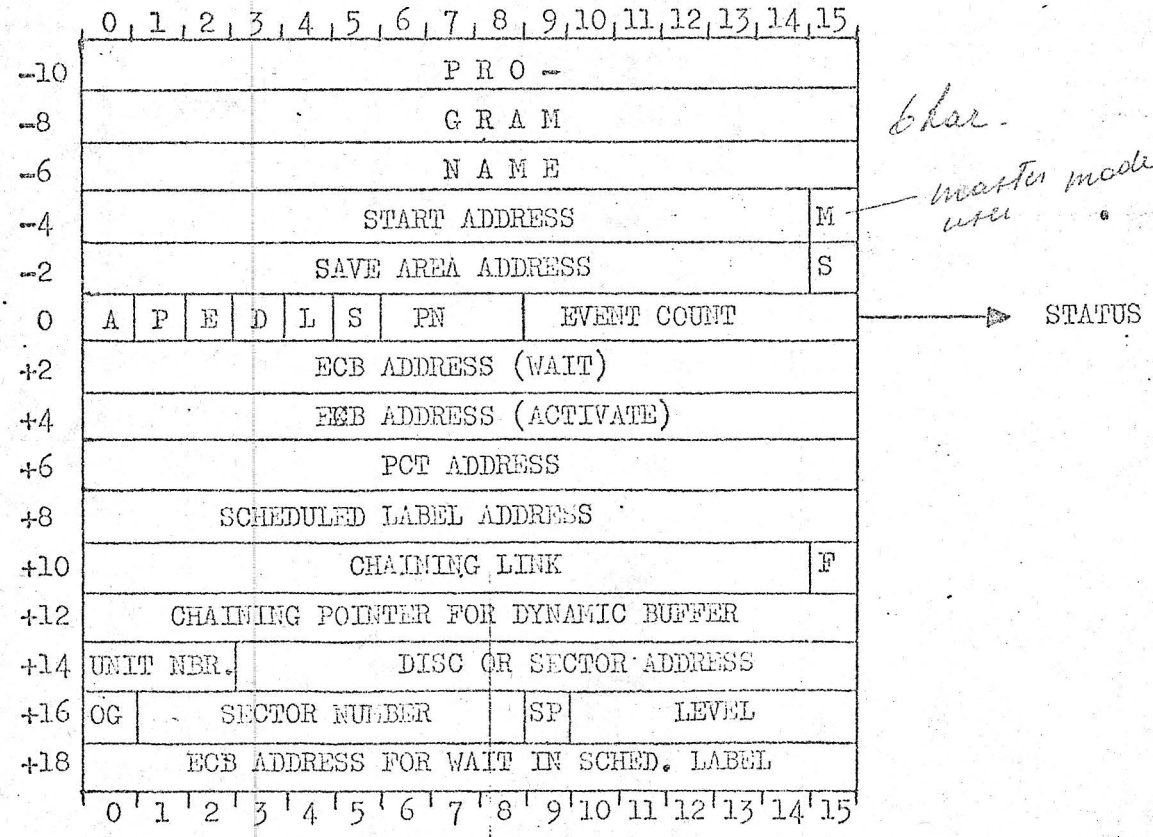
arily. It has requested an external event, e.g. input or output, and is waiting for that event to occur.

(N.B.: when a program is in Pause state, it is waiting to be restarted by the operator).

Programs, residing on disc, can exist in two versions: an original and a swapped version. When the original version was written onto the disc initially, it must then have been declared swappable. In that case, an extra area is reserved behind it on the disc, of the same length. Such programs must be loaded in the dynamic area (P6) and may expect to be swapped back onto disc during execution, in favor of a higher priority program. Thus, for these programs, two additional states are possible:

- loaded: When a disc resident program has been loaded into memory (either the original or the swapped version).
- swapped: when the loaded disc resident program is swapped out of memory and recorded onto disc as a swapped version, different from the original one.

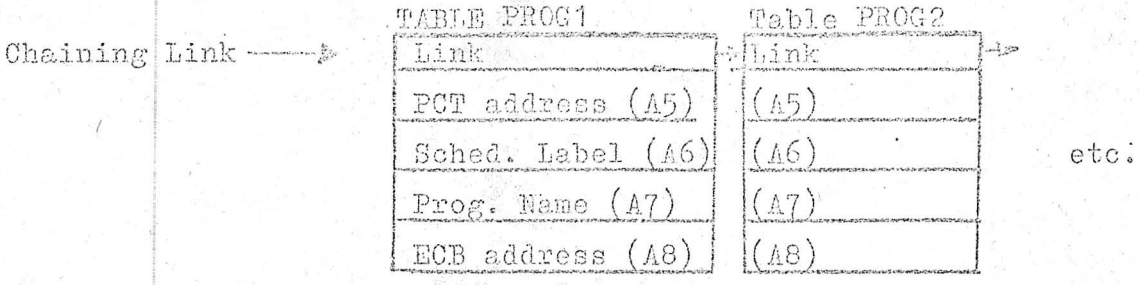
The Program Control Table, built by the monitor for each program loaded, contains the following information:



where:

- word -10, -8 and -6 contain the name of the related program.
- word -4 contains the start address of the program.
  bit 15 (M) = 0, if the program is in master mode.
  1, if the program is in user mode.
- word -2 contains the address of a save area where the monitor saves registers A0 to A14 and PSW in case of an interrupt.
  If S = 0, this is the main program
  1, this is a scheduled label.
- word 0 is the Status Word. At the same time, it is the address of this Program Control Table.
  Bit 0: A=0, if the program is active, 1 if inactive
  Bit 1: P=1, if program is in pause state, 0 if not
  Bit 2: E=1, if main program has exited, 0 if scheduled label exit
  Bit 3: D=1, if program is disc resident, 0 if core resident
  Bit 4: L=1, if the program has been loaded (disc-resident prog), 0 if not
  Bit 5: S =1, if the program has been swapped, 0 if not
  Bits 6 - 8: PN= partition number for this program (from 0 to 7)
  Bits 9-15: event count
- word +2 contains the ECB address on which this program is waiting if it has given a 'Wait for an event' monitor request.
- word +4 contains the ECB address for an 'Activate' monitor request for another program, to enable that program to wait for this one's exit.
- word +6 contains the address of the Program Control Table of the activated program.
- word +8 contains the address of the first scheduled label in this program.
- word +10 is a chaining link. If there are more programs on the same level, this word points to the PCT (Status word) of the next program and so on. The chaining link of the last program points back to this PCT. In this case, F=0.
  It is also possible that several programs have made an 'Activate' monitor request for this one. In that case these requests are queued, link-tables are built for each program in the queue and the chaining-link points to the first word of the link-

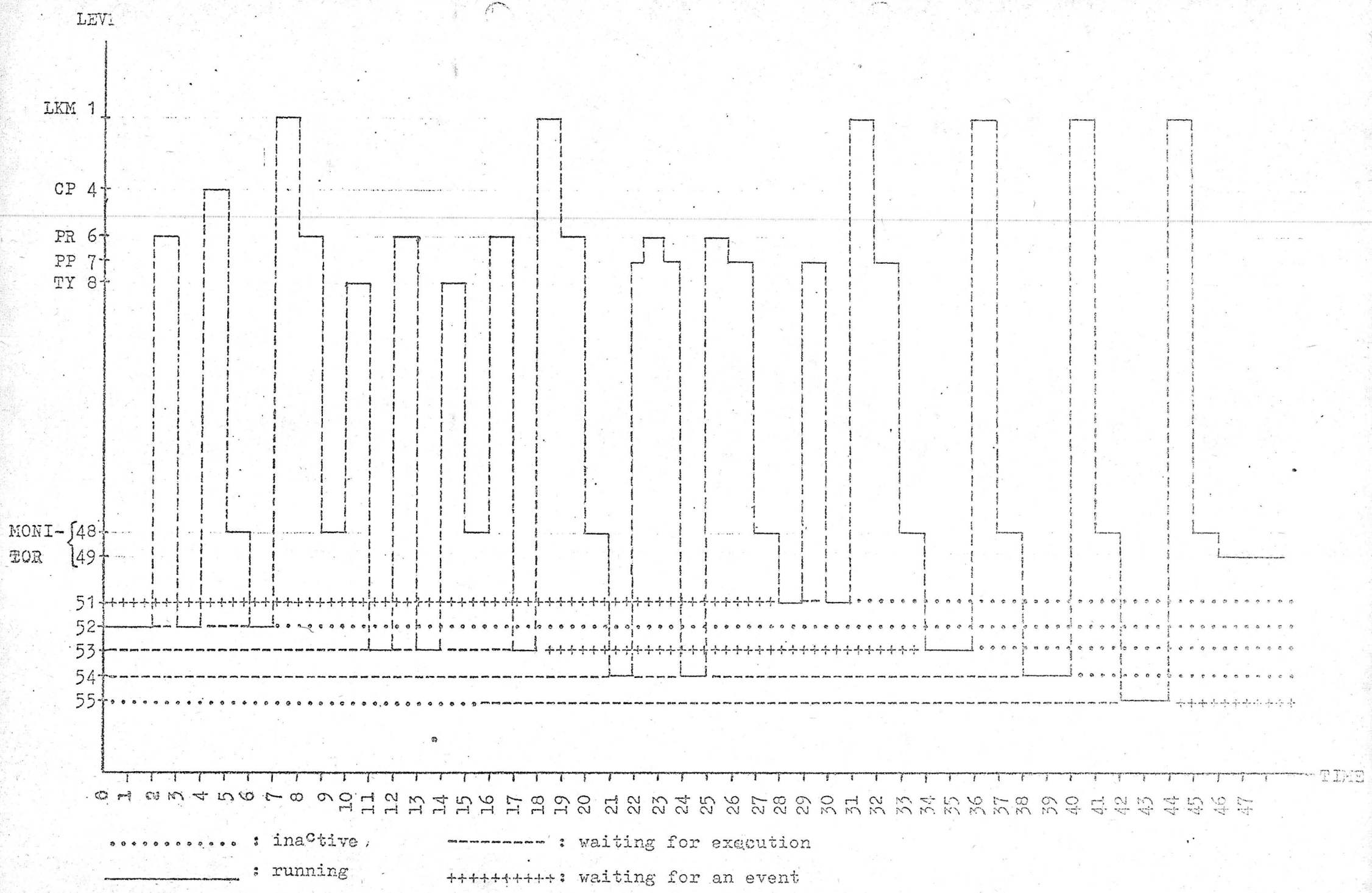
table belonging to the first program in the queue. As follows:



The last link points back to the chaining link in the PCT.

In case of such a stacked activate request, F=1.

- word +12 contains a chaining pointer for a dynamic buffer, i.e. the address of a buffer reserved by a 'Get Buffer' request.
- word +14: Bits 0-2: disc unit number (from 0 to 7)
  Bits 3-15: disc or sector address
- word +16: Bit 0: this bit is set to 0 in case a reserved buffer overflows. The program is put in wait state until a new, large enough, buffer has been reserved.
  Bits 1 to 8: disc sector number .
  Bit 9: SP=1, if this program is swappable
  0, if not.
  Bits 10-15: contain the priority level of this program.
- word +18: It is not allowed for the user to specify a 'Wait' request in a scheduled label. In emergency cases, the monitor will do this and this word is then used for the ECB address.



Example of operation of interrupt system

Time

0 Level 51 is waiting for the PR, 52 is running, 53 and 54 are waiting for execution, 55 is inactive.

1

2 PR interrupt, for level 51.

3 Level 52 is restarted.

4 Control panel interrupt by operator. This results in:

5 Monitor starts typewriter for communication with the operator.

6 Level 52 is restarted.

7 Level 52 gives monitor request (LKM) Exit: 52 becomes inactive.

8 PR interrupt, for level 51.

9 Monitor starts level 53, but

10 Typewriter interrupt, for the monitor.

11 Level 53 is started.

12 PR interrupt, for level 51.

13 Level 53 is restarted.

14 Last typewriterinterrupt, for the monitor.

15 Monitor examines typewriter messages, finds: ST 55, so level 55 becomes active.

16 PR interrupt, for level 51.

17 Level 53 is restarted.

18 Level 53 gives I/O monitor request (LKM) for PP, so 53 from here on is 'waiting for an event'.

19 PR interrupt. (It was already waiting, but the LKM interrupt has a higher priority).

20 Monitor starts level 54.

21 Level 54 starts running.

22 PP interrupt for level 53, interrupted by a PR interrupt.

23 PR interrupt, then PP interrupt routine is restarted.

24 Level 54 is restarted.

25 Last PR interrupt for level 51.

26 PP interrupt, for level 53.

27 Monitor decides to restart level 51 (its PR input is ready.)

28 Level 51 is restarted.

29 PP interrupt, for level 53.

30 Level 51 is restarted.

31 Level 51 gives monitor request(LKM) Exit: 51 becomes inactive.

32 Last PP interrupt for level 53.

33 Monitor decides to restart level 53.

34 Level 53 is restarted.

35 Level 53 is running.

36 Level 53 gives monitor request (LKM) Exit: 53 becomes inactive.

37 Monitor decides to start level 54.
38 Level 54 is restarted.
39 Level 54 is running.
40 Level 54 gives monitor request (LKM) Exit: 54 becomes inactive.
41 Monitor decides to start level 55.
42 Level 55 is started.
43 Level 55 is running.
44 Level 55 gives a monitor request (LKM) Pause: 55 now is in the state of 'waiting for an event'.
45 Monitor decides to start the idle task, which is on level 49.
46 The idle task is started.
47 The idle task is running.

Real-Time Capabilities

For foreground programs, i.e. those programs making use of real-time facilities, it is necessary that they be connected to a level.
For interrupt levels, this is done by the loader according to the AORG directives given by the user to fill the associated hardware locations with the starting addresses of the interrupt routines.
For software levels the connection must be made by
- operator control message CL; or
- a monitor request from another program ('Connect a Program to a Level')

For the background level the connection is made by the loader.

Program Swapping

In order to use processor time more efficiently, program swapping has been included as a feature of this system.
During a program run, it is possible that the execution of the program is halted temporarily at certain points (e.g. pause, I/O). To allow parallel processing of other disc resident programs, the monitor can then swap the suspended program back onto disc to make room for another one.
This feature entails certain restrictions:
- swapping is possible only to and from the dynamic area of the foreground (P6)
- Any internal buffers, e.g. for I/O and Event Control Blocks of a swappable program must be reserved and used in the buffer pool (P1). For this purpose, there are two monitor requests: 'Get Buffer' to reserve a buffer and 'Release Buffer' to release the buffer space after use.
- When these programs are stored on the disc for the first time, the user must define whether a program is swappable or not, in the operator control message SA. If a program is declared swappable, the monitor will reserve an area behind the program on the disc. This area will have the same length as the swappable program and will be used to store the program in case of a swapping procedure.

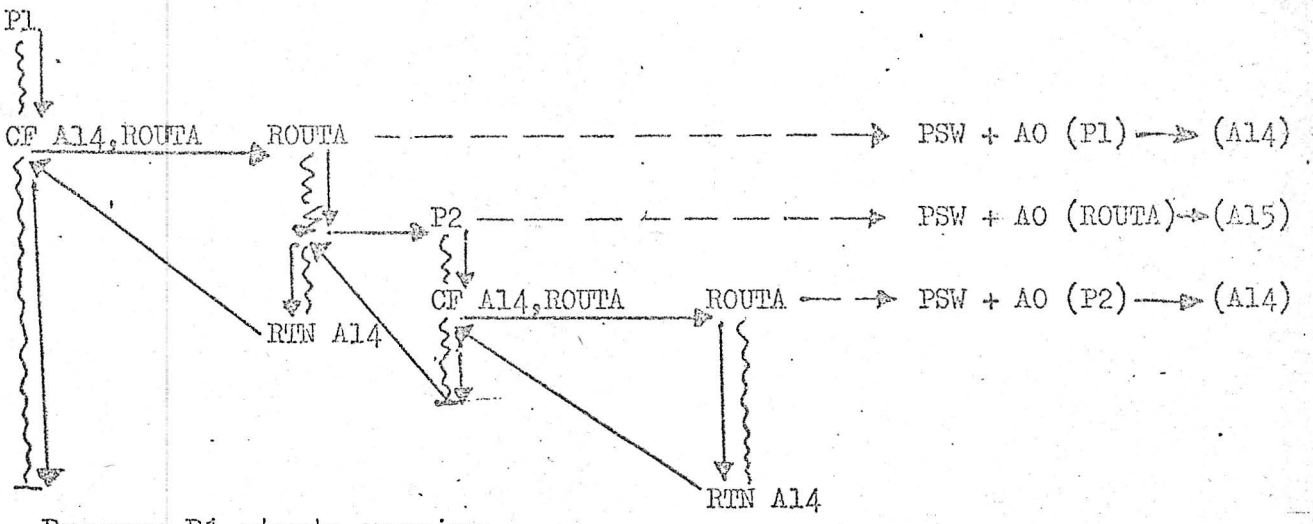
Reentrant subroutines

Regular software level programs are not reentrant, i.e. they cannot be interrupted at any point on request from another program. It is possible, however, to store subroutines, which are to be used by several programs, in the partition P0 (monitor and common subroutines). These subroutines must be loaded into this partition at monitor loading time and it is not possible to add new subroutines dynamically later on. Programs can call a common subroutine by means of the 'Call Function' instruction, as follows:

CF A14,<subroutine name>

If a reentrant subroutine itself needs memory space, it must get this in the buffer pool (P1) through the monitor request 'Get Buffer'. If this routine wants to call another, also reentrant subroutine, it must reserve two extra words in this buffer.

Example:

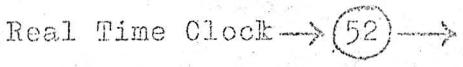


- Program P1 starts running.
- CF is given, PSW + A0 of P1 are stored in stack, of which address is given in A14 and subroutine ROUT is started.
- Program P2 interrupts, PSW and A0 of ROUT are stored in stack specified in A15 and P2 starts running.
- CF is given, PSW + A0 of P2 are added to the stack 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 in P0; the stack indicated by A15 is reserved and handled by the monitor.

Time-slicing

For software level programs time-slicing is possible, i.e. allotting equal amounts of processor time to several programs connected to one level, under control of a timer and by turns.
This functions as follows:
Let us assume that level 52 is connected to a timer, e.g. the real time clock, and also to a program containing the monitor request which activates the switching procedure:




```
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. Provided level 53 or 54 have not interrupted, the dispatcher then activates one of the programs A, B or C on level 55, to whichever one it was pointing.
This program then starts running:
- until it is interrupted by 53 or 54 or a level of higher priority than 52, or
- 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 much as possible, the switch request and the clock can be connected to level 50 and the programs to be switched to level 51. In this case, only hardware interrupts can interfere.

Real Time Clock - Timers

The optional real time clock is a pulse timer which generates an interrupt signal every 20 milliseconds. It has a fixed device address: X'3F' for all configurations and a fixed interrupt level: 2. Turning the clock on and off is done through CIO start and CIO stop instructions.
It is possible, through operator control message or monitor request, to connect a program, which has first been connected to a level, to the clock also. In this way, a program can be activated through clock interrupts.

Input/Output

Input/Output operations are initiated through a monitor request for peripheral units, not directly from the user program through I/O instructions. At system generation time the I/O modules inside the monitor are filled with the necessary data.
In the I/O requests in his program the user indicates the input or output functions he wants performed. These functions are of three types:

- Basic I/O requests: these requests are meant for binary I/O, as the monitor does not in this case provide for data conversion or character checking, only for control command initialization and end-of-operation signals.
- Standard ASCII I/O requests: for these requests some facilities are available, such as error control characters and data conversion from external code to internal ASCII code and vice versa. The characters are checked for end-of-data and they are stored seven by seven bits, two characters to a word.
- Standard object I/O requests: for these requests the following facilities are available: error control characters, checksum and data conversion from external 4-4-4-4 or 8-8 punched tape format to internal 16-bit format.

  If a user has a non-standard device in his configuration or adds a new device, the interrupt routine he writes for this device allows him to have direct control of it.
  Any I/O operations in such a routine must be made through this type of request.

In his I/O requests the user not only specifies the I/O function he wants performed, but also gives the address of an Event Control Block (ECB). This ECB contains further information relevant to the I/O operation such as the address of the buffer to be exchanged, its length and the file code belonging to the file or device on which the I/O operation is to be performed (see - I/O monitor request).

File Codes

File codes are used for the purpose of referencing files and devices. They consist of two hexadecimal characters from X'01' tp X'FF'. Some of these file codes are standard, i.e. the user is bound to specify them, as they have been incorporated in the system:

X'01' = Standard Source Input
X'02' = Standard Listing Output
X'03' = Standard Punch Output
X'04' = Standard Object Input
X'05' = Operator's Typewriter (Input and Output)
X'0A', X'0B', X'0C', X'0D' are reserved for disc units.
X'0E' = Standard Library Input
X'0F' = Standard Program Input

File codes are assigned by means of the operator control message AS.
User file codes, i.e. the file codes that are not standard but can be assigned by the user as desired, are all codes from X'10' through X'7F'.
The file codes from X'F0' through X'FF' must be defined at disc initialization time, to be used for data files on disc.

Device Addresses

When assigning a file code, it is necessary to specify the address of the device to which the assignment is made. It should be noted, that some of these device addresses are standard. They are the following:

X'01'/X'02': discs
X'03': magnetic tape
X'04': cassette tape
X'05': card reader
X'06': line printer
X'10': operator's typewriter
X'20': punched tape reader
X'30': tape punch
X'0F': adapter
X'1F': Direct Memory Access Channel (DMAC)
X'2F': Memory Increment Data Break (MIDB)
X'3F': Real time clock

It is not allowed to use X'00' as an address.
X'2F' and X'3F' are fixed by hardware.
X'20', X'30' and X'10' are fixed by software to have standard configurations.
The DMAC and adapter must always have addresses X'1F' and X'0F' because of their priority.

Device Names

The following device names are used:
TY: operator's typewriter
TR: ASR tape reader
TP: ASR tape punch
PR: punched tape reader
PP: tape punch
LP: line printer
CR: card reader
CP: card punch
MT: magnetic tape
TC: magnetic tape cassette