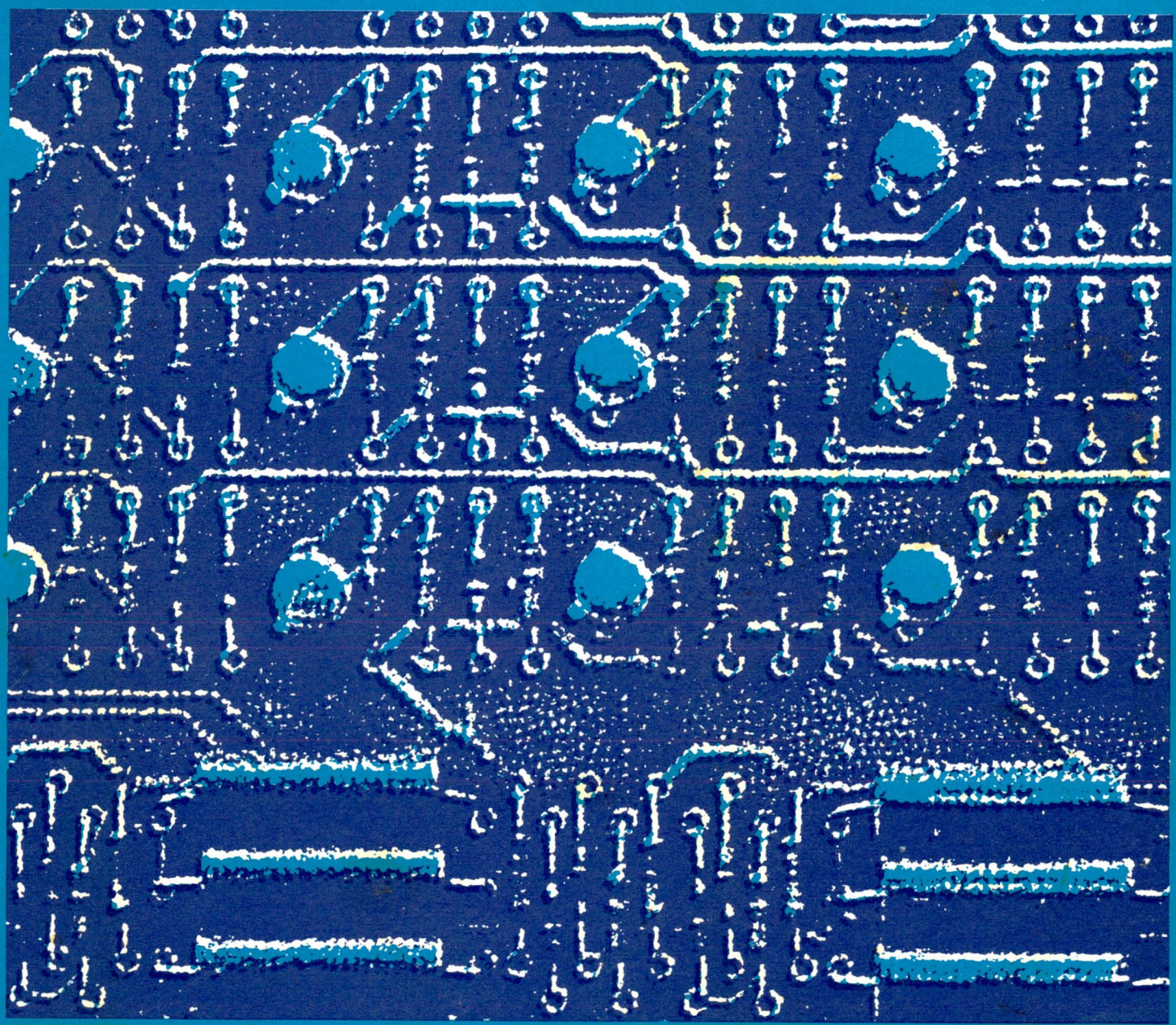
P800M Programmer's Guide 1

Volume I: Basic Operating Monitor



PHILIPS Data Systems

PHILIPS

P800M Programmer's Guide 1

Volume I

Basic Operating Monitor

A publication of
Philips Data Systems B.V.
Marketing Group Small Computers
Apeldoorn, the Netherlands

Publication number 5122 991 27322

July, 1976

Printed in the Netherlands.

This is the first of a six-volume set dealing with the Basic Operating Systems (non-real time and real time) for the P800M series. It describes the Basic Operating Monitor.

The other volumes of this set are:
Volume II: Instruction Set
Volume III: Software Processors
Volume IV: Basic Real Time Monitor
Volume V: Small Real Time Monitor
Volume VI: Cassette Operating System

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 the 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

Introduction

The Basic Operating Monitor is a punched-tape oriented system program which can load, execute and supervise user programs. It can be extended to handle other devices, such as card reader and line printer.

The monitor handles one program at a time, but some sort of multiprogramming is provided by the "scheduled label" feature, which allows for execution of user routines concurrently with the main program.

The monitor has a modular structure to permit the user to select only those modules necessary to fulfill his system requirements. At system generation time these selected features are merged into one single module. The memory space occupied by the monitor varies therefore with the number of modules included, but it can be used in the following minimum configuration:

- 4k memory
- operator's typewriter with ASR tape equipment
- for system generation, high-speed punched tape equipment is highly recommended, as the system software is delivered in 8+8 tape format.

1 Principles of operation

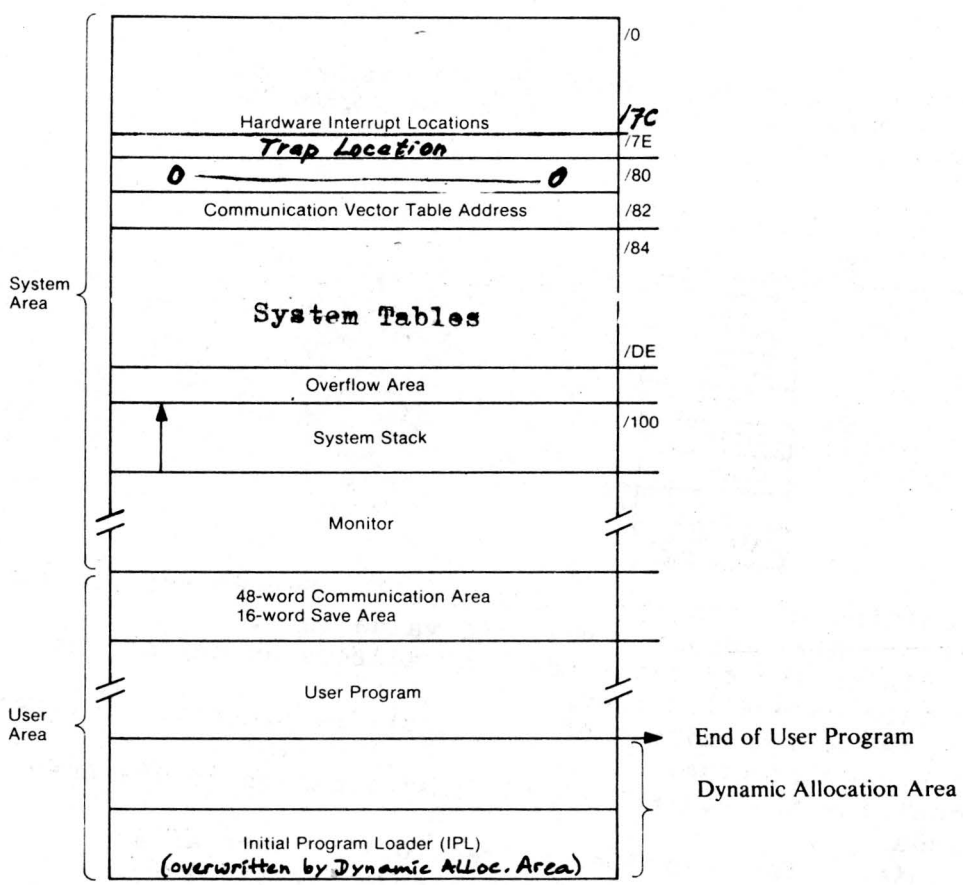
The Basic Operating Monitor (BOM) is a monitor handling one program at a time, to be applied basically as a punched-tape oriented program development tool.
Its structure is very modular, in order to allow the user to select those monitor parts that he will need for his application, so as to increase efficiency and minimize memory occupation.

The monitor is loaded by the Initial Program Loader (IPL), which itself is loaded by a bootstrap loader. Details on these procedures can be found in Chapter 9, Operation.
Then the user program is loaded. If the operator communication package is included in the monitor, it is loaded by the monitor.
If the operator communication package is not included, the IPL, which was read into the upper part of memory by the bootstrap loader, will load the user program. The area occupied by the IPL may in both cases be used by the user program as a dynamic buffer allocation area, for which purpose two monitor requests are available: Get Buffer and Release Buffer.
If the operator communication package is included in the monitor, loading is done by operator control message and the loader forms part of the monitor.

The monitor itself handles the standard interrupt signals, controls I/O operations and executes functions requested by the user in his program by means of monitor requests. The monitor modules are centered around a dispatcher, which determines on the basis of interrupt signals and priorities which routine or program must be executed. Although the BOM is designed to handle one program at a time, a form of multi-tasking can be achieved by using scheduled label routines (see Programming section). These routines are attached to the specification of a monitor request and enable a program to run concurrently with, for example, an I/O operation.

Because the BOM has a modular structure, the user must define his own system at system generation time by selecting the monitor modules he wants.

The memory layout is as follows:

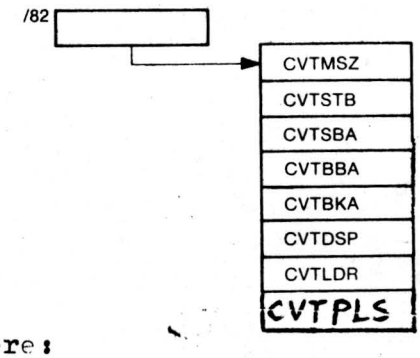


Locations /0 to /7C are *hardware interrupt locations*. They are hard-wired to internal and external interrupt lines. Each location contains the address of the interrupt routine required to service the interrupt connected to that location. The interrupt connected to location /0 has the highest priority (level 0). See chapter 4.

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 store, multiple load 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 table has the following layout:



where:

CVTMSZ contains the machine memory size in characters. If the memory size is 32k words, the value is 0.

CVTSTB contains the system stack base address as defined at system generation time.

CVTSBA contains the address of the first free location following the user program.

CVTBBA contains the address of the last location in the user area.

CVTBKA contains the beginning address of the user area.

CVTDSP contains the address of the dispatcher (M:DISP).

CVTLDR contains the option load flag:
1 if the loader was included in the monitor at SysGen
0 if it was not.

CVTPLS contains the clock pulse, which is normally 1 (20 msec. clock).

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, PSW are stored here 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 and 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 which remains after the user program area, is reserved for *dynamic memory allocation* after termination of all loading procedures. From this area, blocks of memory space can be requested by the system or by the user. The user must send a 'Get Buffer' monitor request for this purpose. When he does no longer need the buffer, he must send a 'Release Buffer' monitor request.

3 Interrupt System

Programs and routines under the BOM run on the basis of an interrupt and priority system which consists 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 levels
49	operator routines	
61	abort module	
62	user program	
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 locations /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 strongly 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
 4:    punched tape reader                    etc.
 5:    tape punch
 6:    operator's typewriter
 7:    control panel
 8 to /F:  free
/10:        disc
/11:             disc
/12:   disc
/13:   magnetic tape
/14:   cassette tape
/15:   card reader
/16:   free
/17:   line printer
/18 to /1F:  free
```

T, with the other interruptable monitor service routines

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 or monitor request handlers.

lor equal to

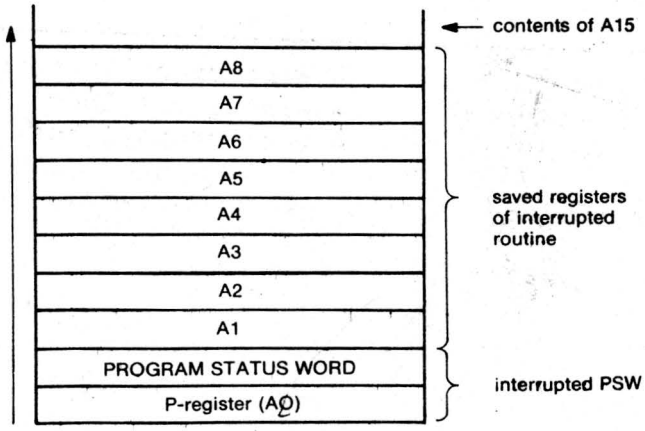
SOFTWARE PRIORITY LEVELS

Under the BOM, the user program is connected to level 62. It is activated after loading, by the operator command ST. The operator command package, abort module and idle task also operate on software priority levels.

STACK

When an interrupt occurs, certain information about the interrupted program or routine must be saved before the interrupt can be serviced.
For user programs, this is done in the system stack pointed to by register A15 or, when there is a priority change from one program to another, in a 16-word save area reserved in front of each program, where P-register, PSW and registers A1 to A14 are stored.
For interrupt routines it is done in the system stack.
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 moreover a number of registers:

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

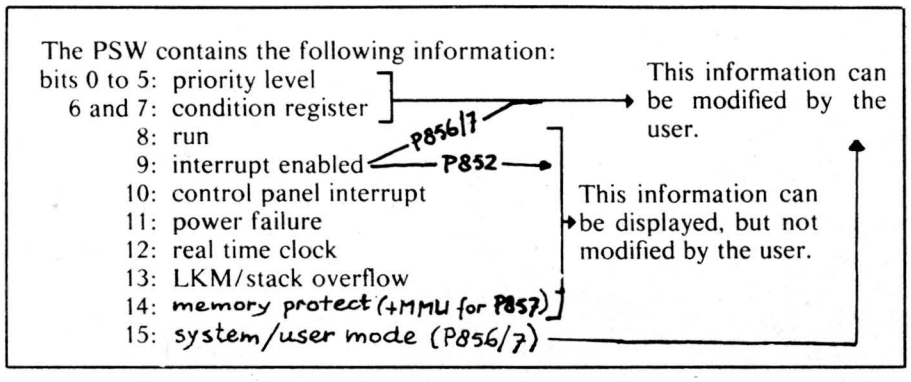


10

When the stack pointer (A15) reaches the value /100, the last location before the stack overflow area, an interrupt will be given.

Or when its value is below /100

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



The PSW can be displayed on the control panel.

4 Programs

There are three types of user-written programs:
- programs (user-written codings sequences), connected to **software** level 62
- subroutines ; they are called through a CF (Call Function) instruction and run on the level of the calling program.
- interrupt routines, servicing one of the 63 possible hardware interrupts and connected to any of the **hardware** levels 0 to 47.

SOFTWARE LEVEL PROGRAMS

These are user programs connected to any of the software priority level 62.

When a user program is loaded into memory, the system reserves a communication area of 48 words and a save area of 16 words in front of the user program. The communication area can be used by the user to store information and the save area is used by the system to store the program's register contents in case of a scheduled label interrupt in the main program. At the start of the user program, register A1 contains the address of the first location of the user program area and register A2 contains the first free location following the user program.

Monitor requests allow the user to request certain functions from the monitor by means of an LKM instruction followed by a DATA directive with a number to specify the function. Certain parameters may have to be loaded into the A7 and/or A8 registers first. Monitor requests can be combined with scheduled labels (see below).

After a program has been loaded, all relevant information about it is stored in a Program Control Table in the PCT Pool in 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 it 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.

Every Software Level Program is preceded by a 16-word save area reserved by INIMON. If any scheduled labels are used in the program, the user must reserve himself an additional save area, for interrupts occurring while the program is in a scheduled label sequence, to store the contents of its P-register, PSW and registers A1 to A14. The user must reserve 16 + (N + 1) words, where N is a system generation parameter specifying the maximum number of scheduled labels which may be queued in this table at any one time (see below):

```
        IDENT     PRNAME
        RES       16 + (N + 1)   N + 1 = FILLAB table length
START       Coding
        LKM
        DATA      3
        END
```

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:

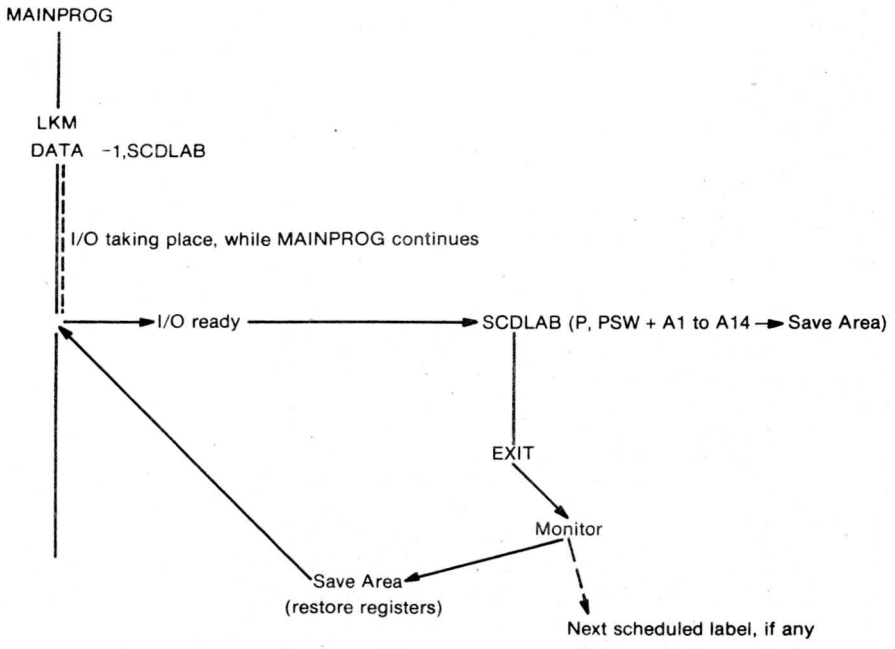
normal I/O request:

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

with sched. label:

```
        LDK     A7,CODE
        LDKL    A8,ECBADR
        LKM
        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 of the main program are stored in the 16-word save area in front of the user program. except A8(ECBADR)
When entering the SCDLAB routine, no register value is significant, so it is not possible to pass parameters to the scheduled label sequence. 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. is 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 defined at system generation time.
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 the whole system.

INTERRUPT ROUTINES

User interrupt routines must have been connected to one of the interrupt locations in memory (locations /0 to /7e. 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. Another method, providing a relocatable routine, is shown in the following 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 /20

```
        IDENT   INTRO
        :
        LDKL    A1,ROUT
        ST      A1,/20
        :
ROUT    :       }   INTERRUPT ROUTINE
        :
        END
```

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. 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
STR   A2,A15
:     :
STR   A8,A15
  coding
LDR   A8,A15
LDR   A7,A15
RTN   A15
```

For P856/7 or, with simulation routines included for P852:

```
MSR   8,A15
  {
  coding
MLR   8,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 registers 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 System Software Manual.

(Note: If an INH instruction immediately follows the ENB instruction, a dummy instruction must be inserted, because external interrupts are scanned every two instruction. This dummy instruction may, for example, be another ENB, so that then the correct sequence becomes: ENB - ENB - INH)

DYNAMIC MEMORY ALLOCATION

The user can make temporary use of the dynamic allocation area of memory if his monitor contains the module with the two monitor requests:

- Get Buffer, to reserve a user buffer in the dynamic allocation area;
- Release Buffer, to release the area created by the Get Buffer request.
  Memory allocation is in blocks, the block length being specified by the user. Preceding each block is a control block containing links and other parameters *which must not be destroyed.*
  For the syntax of these requests, refer to the section Monitor Requests.

5 Input/Output

All I/O operations are initiated by an I/O monitor request. At system generation time, the necessary modules and tables for fulfilling this request must have been filled. 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 three types of I/O request (as specified in A7):

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.

Standard Object I/O requests: by means of standard conventions facilities are provided for error control characters, checksum and data conversion from external 4+4+4+4 or 8+8 punched tape format to internal 16-bit format.

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, pointed to by register A8, the user specifies the file code (logical address) of the device 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 concerning the result of the operation in this ECB, so that it may be verified by the user program.

The **file codes** are a means of giving a logical address to a file or a peripheral unit. They are defined at system generation time and consist of 2 hexadecimal digits from /01 to /FF. The following file codes are standard:

/01:	Source Input
/02:	Listing Output
/03:	Punch Output
/04:	Object Input
/05:	Operator's Typewriter (input and output)

The following are **recommended** file codes:

/06: ASR tape reader
/07: ASR tape punch
/08: High-speed tape reader
/09: High-speed tape punch
/0A: Line printer
/0B: Card reader
/0C: Magnetic tape cassette
/0D: Magnetic tape cassette
/0E: Magnetic tape cassette
/0F: Magnetic tape cassette

These can also be re-assigned by the user, i.e. it is possible to re-assign file code 08 to, for example, the card reader.

The remaining file codes, up to /FF can be assigned by the user to any other files or devices.

6 Operator control messages

If the operator communication package has not been included in the monitor, processing is carried out according to the instructions in the program.
If the operator communication package has been included, the operator has additional control over program execution by means of a set of operator control messages which are described in the following paragraphs.

Control messages are entered via the operator's typewriter. The operator initiates communication with the system by pressing the INT button on the control panel. This causes the system to type out M: to which the operator replies with a control message, followed by LF CR.
A message consists of two characters identifying the message, followed by a space which is possibly followed by a number of parameters. These parameters may be separated by spaces or commas. Some parameters are optional.

Notes:
- The operator communication package is an optional module. If this module is not included in the monitor, the operator must take care not to press the INT button on the control panel, because then the system will issue a Halt instruction.
- Every numerical value in an operator control message is assumed to be a hexadecimal value, specified with or without the slash(/).
- If a message contains an error, the system types out the message ER. The operator may then press the INT button and type in a correct message.

In the following paragraphs, the syntax and use of the available operator messages are described, where Backus Normal Form is used as the notation for the syntax description:

|	means:	or;
[]	means:	optional component; any or all items within these brackets may be omitted, e.g. [+|−]<integer> could be +<integer>, −<integer>, or <integer>;
[]	means:	alternative components; one of the items within these brackets must be selected, e.g. [+|−]426 must be either +426 or −426;
< >	means:	these brackets contain a syntactical item;
␣	means:	space.

Table of operator control messages

AB ABORT A PROGRAM AB

syntax: AB
use: This message can be given to terminate a program before its normal end.
All buffers are deallocated.

AS ASSIGN FILE CODE AS

syntax: AS␣ < file > ␣ < device >
use: By means of this message a file code can be assigned to a device or a previously created file code can be modified.
< file >: file code (refer to Appendix C)
< device >: device name (refer to Appendix C), followed by the physical device address (2 hexadecimal characters).
example: AS 03 TP10
File code 03 is assigned to the ASR tape punch with physical address 10.

DM — DUMP MEMORY — DM

syntax: DM␣<address 1>␣<address 2>

use: This message produces a hexadecimal dump on the standard listing output device in full lines, so that <address 1> and <address 2> are included. Each address consists of up to 4 hexadecimal characters.

example: DM 0002 0004

causes a full line to be listed of the values contained at addresses 0000 to 000E inclusive.

HD — HALT DUMP — HD

syntax: HD

use: This message is used to terminate a memory dump before it reaches its specified end. This can be useful, for example, if the dump is being output on the typewriter, a slow device for large scale output, and continuation of the dump becomes unnecessary.

LD — LOAD A PROGRAM — LD

syntax: LD␣[<value>][␣M]

use: This message is used to load a program into the memory. It is loaded from the standard object input device.

<value>: displacement value (in hexadecimal), relative to the beginning address of the user area. If specified, the program is loaded <value> characters after the beginning of the user area.

M : if specified, indicates that the loaded program will run in system mode. (P856/7 only)

Whenever a program is loaded, the program identification, loading address and length are printed on the typewriter.

MC — MANUAL DEVICE CONTROL — MC

syntax: MC␣<file code>␣<software order>[␣<repeat factor>]

use: This message is given when the operator wishes to perform a manual operation on a magnetic tape device.

<file code> is the file code assigned to the device, consisting of 2 hexadecimal characters.

<software order> consists of 2 hexadecimal characters that have the following significance:

16: Skip forward to EOF mark

22: Write EOF mark

24: Write EOV mark
26: Write EOS mark
31: Rewind to load point
33: Backspace one block
34: Space one block forward (not allowed for cassette)
36: Skip backwards to EOF

38: Unlock

<repeat factor> allows the operator to have the required function performed as many times as specified here, with only one MC message.

example: MC 15 34
The device with file code 15 is to skip forward one block.

PS — PAUSE — PS

syntax: PS
use: To temporarily stop the processing of the program. To restart it the operator must type RS.

RD — RELEASE DEVICE — RD

syntax: RD␣<device address>
use: Following an unsuccessful I/O operation, the monitor may type out a PU system message.
The operator may now give the RD message if he wishes to release the operation from the device on which the error occurred.
<address> is the physical address of the device to be released, as 2 hexadecimal characters.
Control is returned to the user with the status in ECB.

RS — RESTART A PROGRAM — RS

syntax: RS[␣<new A7>]
use: To restart a program stopped temporarily by a PS message or a Pause monitor request.
If the program has been stopped by a monitor request it can be restarted with a new value in register A7.
example: RS␣081E
The program that was stopped by a Pause monitor request, is to be restarted with the value /081E in register A7.

RY — RETRY I/O OPERATION — RY

syntax: RY␣<device address>

use: Following an unsuccessful I/O operation, the monitor may type a PU system message.
The operator may now give the RY message to retry the operation, after taking any necessary steps.
< address >: the physical device address (2 hexadecimal characters) where the operation has to be retried. If the operation is now successful, control is returned to the user with the status in the ECB, otherwise, the monitor will type out a new error message if another retry is possible.

example: RY 02
Retry the last I/O operation on the device with physical address 02.

ST — START A PROGRAM — ST

syntax: ST

use To activate the last loaded program.

WM — WRITE INTO MEMORY — WM

syntax: WM␣ < address > ␣ < value 1 >[␣ < value 2 > . . .␣ < value n >]

use: This message can be used to correct one or more memory locations. < address > is the first address to be altered (up to 4 hexadecimal characters).
< value 1 > to < value n > are values to be entered in the memory locations starting at < address >, i.e.
< value 1 > is placed in < address >
< value 2 > is placed in < address > plus 2.

example: WM,4FE,44F,3FE4
The value /44F is placed in memory location /4FE.
The value /3FE4 is placed in memory location /500.

7 Monitor Requests

The user program can request the monitor to execute certain functions. A request takes the form of a 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 module is not available, A7 is always −1. The following requests are available:

Monitor Request	Register A7	Register A8	LKM DATA	page
I/O Requests	Order	ECB Address	1	28
Wait for an Event	—	ECB Address	2	32
Exit	—	—	3	33
Get Buffer	Buffer Length	—	4	34
Release Buffer	–	(A14: Buffer Address)	5	35
Pause	Message Length	Block Address	6	36
Control Abort	Control Block	Routine Address	7	37

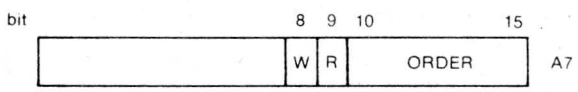
Note: It is possible to attach a scheduled label to a Monitor Request. For details, see Chapter 4.

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.
Register A7 is loaded with a CODE that specifies the details of the I/O function, as follows:



W and R specify the mode of operation:

W = 1: Return to the calling program will be made after completion of the I/O operation.

W = 0: Return to the calling program will be made as soon as the transfer has been initiated. The program will give a Wait Request later, for synchronization.

R = 1: Any abnormal conditions will be processed by the user program. The system will return the Hardware Status in ECB word 4 (see under ECB). No retry is possible. Possible with Basic Read/Write only.

R = 0: Any abnormal conditions will be processed by the system and the Software Status will be returned in ECB word 4 (see under ECB).

ORDER consists of a hexadecimal value on six bits, identifying the I/O function:

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

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. Where ASR is the only input possibility. For ease of handling, a word is divided equally and punched over four rows, 4 bits per row.

08: Object Write −8+8 format. A word is contained within 2 rows, 8 bits per row.
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 format.

30: Get information about a file code

Tape handling orders:

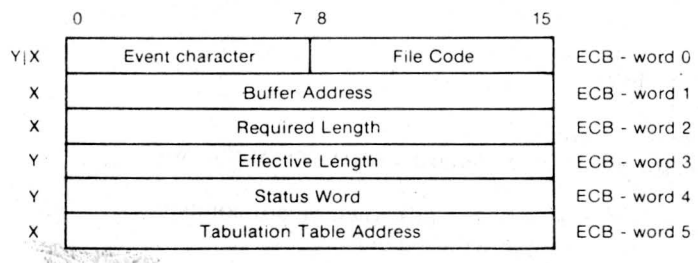
14: Skip forward to next EOS mark
16: Skip forward to next EOF mark
22: Write EOF mark
24: Write EOV mark
26: Write EOS mark
31: Rewind to load point
33: Backspace one block
34: Space one block forward (not allowed for cassette)
36: Skip backwards to EOF mark

38: Unlock

Refer to Appendix A for more information specific to the various peripheral devices.

ECBADR is the address of the Event Control Block containing the parameters for the I/O function:

Event Control Block
The ECB address must be loaded in the A8 register. ECB structure:



where: The words marked X must be filled by the user; those marked Y will be filled by the system.

word 0: event character:
bit 0 = 1: end of operation has occurred for this ECB.
file code: must be loaded before requesting the I/O. See Appendix C.

word 1: address of the user buffer.

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

word 3: effective length which has been transmitted (in words for basic read on card reader, in characters for other devices). This information is stored here by the monitor upon completion of the I/O operation.

Note: If magnetic tape or cassette tape handling orders are given in this monitor request (/16 and /31 to /38), it may be useful to make a separate ECB for these orders, as in these cases only ECB word 0 is important (file code). The contents of ECB1 and 2 are then irrelevant, *but must be unequal to zero.*

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

- Request accepted: for Basic orders this word will be filled with the hardware status (bit 0 = 1). (See Appendix D). For Standard orders the status will be:
  Status 8000: no operation (after RD message) ← bit 7=1: no data on cassette
  positive: bit 8 = 1: End-Of-Volume } mag. or cass. tape
  bit 9 = 1: End-Of-Tape }
  bit 10 = 1: beginning of tape.
  bit 11 = 1: end of input medium.
  bit 12 = 1: incorrect length requested or checksum error.
  bit 13 = 1: illegal character code
  bit 14 = 1: EOS
  bit 15 = 1: EOF
  zero: normal I/O completion.
- Request rejected: for Basic or Standard orders the status will be:
  status C000 and: bit 11 = 1: order unknown or not compatible with the device.
  bit 12 = 1: illegal buffer size.
  bit 13 = 1: illegal buffer address.
  bit 14 = 1: device attached to another program.
  bit 15 = 1: illegal file code.

word 5: for a Standard read on punched tape equipment or typewriter, the user may fill this word with the tabulation table address. This tabulation table gives the format in which the user wishes the printout to appear.

The table has the following format:

Number of Tackets	First Tacket
Second Tacket	Third Tacket

etc.

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

Note: If word 5 is filled, the required length in word 2 must contain both the characters *and* the blanks in the tackets.

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

Upon completion of one of these I/O functions, the system responds as follows:
A7 > 0: request completed.
= −1: the corresponding request module is not in memory.

WAIT FOR AN EVENT REQUEST

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 bit 0 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 must take place in the same or 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.

EXIT REQUEST

Calling Sequence

```
LKM
DATA    3
```

Use

This request is used to indicate the end of a user program or scheduled label routine. The program exit is effected after completion of all I/O operations and after all labels have been scheduled.

A scheduled label routine exit passes control to the next scheduled label routine, if one is present, otherwise control passes to the main program.

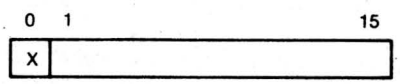
GET BUFFER REQUEST

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):

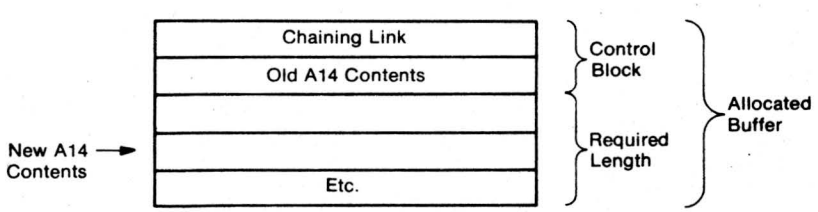


If bit 0 = 0: return to user in case of overflow.
If bit 0 = 1: implicit wait of the calling program in case of overflow.

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, behind the user program 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.

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.

PAUSE REQUEST (usable only if operator communication package is included)

Calling sequence:

```
LDK   A7,ML
LDKL  A8,MA
LKM
DATA  6
```

where:
ML is the length of the user message, in characters, including control character.
MA is the address of the block containing the message.

Use:
To print a user message on the operator's typewriter and then put the user program into a pause state. The system will enter the idle task immediately, and all current I/O operations will be finished. To restart the program, the operator must type the Restart message. Therefore, the Pause request can be given only if the monitor contains the operator communication facility.

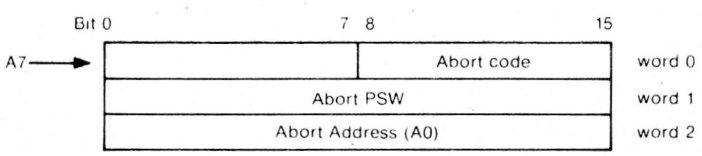
Note: - It is very useful if the user mentions, in the output message, that the program is now in pause state.
- The message must start with a control character.

KEEP CONTROL ON ABORT CONDITION REQUEST

Calling sequence:

```
LDKL  A7.UCBA
LDKL  A8.LABEL
LKM
DATA  7
```

where:
UCBA is the address of a three words long user control block which the system will load with the following information:



Use:
By means of this request the user can himself handle abort conditions, i.e. his own routine replaces the abort handling of the monitor. Thus, the program will not be declared aborted by the monitor, but control will be transferred normally to the user routine attached to this request (register A8). The parameters of the abort condition will be placed in a 3-word block and, according to the returned parameters, the user routine can take action.
This request is to be given once in a program, at the point from where the user wants control over any abort conditions, so mostly at the beginning of the user program. It is usable only once, so if an abort takes place and the program is restarted, the request has to be given again. When a user decides that, at a certain point in his program, he wants to return to regular abort handling by the monitor, he may do so by provoking an "artificial" abort.
The abort code returned in the UCBA block may be one of the following:

01: simulation routine save area overflow
02: non-available instruction used

04: buffer area destroyed or block was bigger than 32k characters.
05: label could not be scheduled

Note: An operator abort (AB) can never be blocked.

8 System messages

System messages are printed on the operator's typewriter by the system. For systems without operator communication, no messages are printed. When abort and exit conditions arise, the P-register halts at specific addresses.

Abort
format: ABORT␣ <code> ␣ <address>
meaning: an error has been encoutered and the program is aborted.
<code>: may be:
01: simulation routine save area overflow.
02: illegal instruction.

04: buffer area destroyed or block was bigger than 32k.
05: label could not be scheduled.
06: operator abort.
<address>: the address at which the abort occurred.

Erroneous Cluster
format: EC
meaning: an erroneous cluster has been encountered on the tape. Program loading stops.

End of File
format: :EOF
meaning: the loader has read the "end-of-file" mark and stops loading.

End of Segment
format: :EOS␣ <address>
meaning: the loader has read the "end-of-segment" mark.
<address>: address of the first free location after the program.

Error in an Operator Control Message
format: ER
meaning: the operator message contains an error. The operator must press the INT button and retype the message.

Program Termination
format: EXIT
meaning: This message is printed when execution of the user program is completed.

Program Identification
format: IDENT␣ < prog. id. > ␣ < address >
meaning: the loader has read the program identification.
< prog. id. > : program name.
< address > : first address of the loaded program.

No Start Address
format: NS
meaning: No start address was specified in the END/START cluster (record type 7. See Appendix B). Program cannot be executed.

Overflow
format: OVL
meaning: Insufficient memory available. Program loading stops.

Peripheral Device Error
format: PU, < device name and address >, < hardware status > [RY]
meaning: a failure has been detected on a peripheral device. RY is printed if a retry is possible.
If the operator releases the operation on the device, the system will consider the I/O operation completed and control will be returned to the user.
< hardware status > : see Appendix D.

Input/Output Error
format: I/O ERROR XXXX YYYY
meaning: An error has occured during an I/O operation, e.g. throughput error, data fault, in connection with an MC operator command.
XXXX is device name and device address.
YYYY is the hardware status of the control unit.

9 Operation

The initial loading procedure is as follows:
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

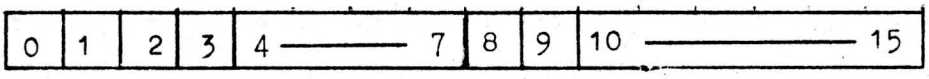
The bootstrap can be loaded in one of two ways, depending on whether the optional ROM bootstrap is included in the system or not:

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.
- push the MC button.
- put the IPL tape on the tape reader and make the reader operable.
- push the RUN button on the CPU control panel.
  Now the IPL is read and loaded into memory.

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

- switch on the CPU.
- put the IPL tape on the tape reader and make it operable.
- set up the device parameters on the toggle switches, as shown below.
- push the IPL button on the control panel. This loads the bootstrap into memory, which immediately starts reading and loading the IPL from the tape reader.

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



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: used only in case bit 1 = 1:
=1: the disc used is a moving head disc
=0: the disc used is a fixed head disc
bit 3: =0: the device is connected to the multiplex
=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 (P824-001)
bits 10 to 15 contain the device address.

Examples:
9130: ASR tape reader, format 4x4, address 30
1020: punched tape reader, format 8+8, address 20
0244: magnetic tape unit, address 04.
0795: cassette tape unit, address 15

LOADING OBJECT TAPES

The Initial Program Loader (IPL) which has now been loaded into memory, is actually divided into two parts. The bootstrap, which was located at addresses 0 to /7F, loads the first part (80 characters) into memory from location /80 and transfer control to this IPL part with start address /84. This IPL part then computes the memory size and loads the rest of the IPL into the high end of memory, where it receives control from the first IPL part.
Now object tapes such as the monitor can be loaded and the bootstrap and first IPL part can be overwritten by them:

- when the IPL has been loaded it outputs the message
  OBJECT TAPE ON READER. THINK OF BASE!
  (If it is an IPL which is not on a separate tape but on the tape in front of the object program, e.g. in front of the Assembler or Stand Alone Linkage Editor, this message is not output, but the IPL carries on loading the rest of the tape immediately after it has itself been loaded.)
- now put the monitor tape on the tape reader, ensuring that the reader is operable. If other object programs than the monitor are loaded the user has, at this point, the opportunity of changing the base address of the object program which is to be loaded. This address is contained in register A9 and can be modified via the CPU control panel when the CPU is stopped. If this register is not modified, it is assumed to contain the value 0, as in the case of monitor loading.
- push the RUN button.
  This restarts the IPL which loads and starts the monitor.
- the following message is output
  IDENT XXXXXX
  identifying the loaded object tape. When loading is finished
  :EOS
  :EOF
  are typed out.
- the monitor is now ready for use and if the INT button on the control panel is pushed, it will type out
  M:
  to ask for a command. Note that the INT button must not be pushed if the OCOM module (operator communication) has not been included at SYSGEN for this will result in abort.

For the BOM, operation from here on is different depending on whether the operator communication package is included in the monitor or not:

- with operator communication package:
  an operator control message, e.g. to load a program, can be entered via the typewriter in reply to the M: output by the monitor.
- without operator communication package:
  the monitor does not type out M:, but requests loading of the user program by typing out:
  OBJECT TAPE ON READER. THINK OF BASE!
  At this point register A9 contains the beginning address of the user program area. If the user wants to change this loading base address, he may now alter the contents of this register. Note that without Op. Comm package INT must not be pushed. Then the user program is loaded by pushing the RUN button. If an error is detected or a failure occurs, the loader halts at a specific address. When the user program has been loaded successfully, the operator starts its execution by pushing the RUN button once more.

APPENDICES

Appendix A Peripheral input/output

This appendix refers to the I/O order which the user must specify in register A7, when he gives an I/O monitor request (LKM1).

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.

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.

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". It is ignored and not transmitted to the buffer.
- 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 7 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).

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 control character (second 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 character. 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 words, excluding first character and checksum. 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.

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 and checksum. An 8-bit checksum is performed and punched.

Cassette and Magnetic Tape:
See under Basic Read (/01).

READ UP TO END-OF-SEGMENT (/14)

High-speed Tape Reader:
The tape is read until an:EOS statement has been read.

Card Reader:
The cards are read until an:EOS statement has been read.

READ UP TO END-OF-FILE (/16)

High-speed Tape Reader:
The tape is read until an :EOF statement has been read.

Card Reader:
The cards are read until an :EOF statement has been read.

WRITE EOF MARK (/22)

Operator's Typewriter:
An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

ASR Tape Punch:
An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

High-speed Tape Punch:
An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

Line Printer:
An end-of-file mark is output as follows: :EOF

WRITE 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.

WRITE EOV (/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 this file is read, the EOT (end-of-tape) status also returned in the status word of the ECB should be ingnored and only the EOV status must be taken into account.

Note: In case the EOT is detected when 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
ECB - word 2:	maximum record size.
ECB - word 3:	left character: unused. right character: device address.
ECB - word 4:	status=0. For line printer, this word contains the number of lines specified for this printer at system generation time.

If the file code in ECB word 0 is set to zero, the other words of the ECB will also contain zeros. This means no device has been assigned to the file code.

OFF LINE (/38)

Magnetic Tape:
This order switches the tape drive off-line.

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

Appendix B Object code record types

Type 0: Block data record
1: Entry point names record
2: External reference names record
3: Code words record
4: Internal modification record
5: Entry point definition record
6: Common length definition record
7: End/Start record.

Appendix C File codes and device names

File codes enable files to be identified and addressed. The file codes are assigned to devices at SYSGEN. The device addresses are fixed by hardware. The size of the file code table is also established at SYSGEN so that dummy file codes should be included if future expansion is envisaged.

Standard File Codes	**File**
01	Source input
02	Listing output
03	Punch output
04	Object input
05	Operator's typewriter (input and output)

Device Names	**Device**
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	Magnetic tape cassette
MT	Magnetic tape
NO	No device: an operation on this file will have no effect.

Appendix D Control unit status word configuration

This is the hardware status as returned in ECB word 4. See I/O monitor request.

Bit	Description	Control Unit						
		ASR	CR	LP	PTP	PTR	TK	MT
0								
1	has become ready						×	×
2	rewinding							×
3	tape mark read						×	×
4	no data						×	
5	Load point beginning of tape						×	×
6	write unable						×	×
7	A or B side (A=1, B=0)						×	
8	device address						×	×
9	device address						×	×
10	EOT tape low				×	×	×	×
11	program error						×	×
12	incorrect length		×				×	×
13	parity error						×	×
14	throughput error	×	×			×	×	×
15	not operable (only significant bit for TST)	×	×	×	×	×	×	×

```
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  3FC8                       SLC     A7,8            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 WORD:
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 DOUBLEWORD: 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-
```

To standardize system generation for all systems and to make it more flexible, a new system generation procedure has been developed, consisting of a number of sysgen processors, some of which are used for the generation of one particular type of system, such as BOS while others are used with the generation of any system. The main advantages of this system generation package are greater flexibility during sysgen, extensive logging of the operation and improved efficiency.

For the generation of a Basic Operating Monitor , the user receives three punched tapes with the following sysgen processors:

- GENMON, a generation monitor used only during the system generation process to run the generation processors. It is preceded by an IPL to load it.
- BOMGEN, which generates the monitor tables from the answers it receives from the user in a conversational process with a standard list of questions.
- IPLGEN, which generates the Initial Program Loader (IPL), the first module to be stored on the system medium. IPLGEN runs under any monitor.
- GENLKE, which runs under GENMON and scans the library of system modules, which are contained on the fourth tape the user receives: BOMLIB. GENLKE selects from this library the modules requested during the BOMGEN phase and links them with the monitor tables generated during BOMGEN.

In the following paragraphs the whole set of operations necessary for generating a BOM is described in a number of sections corresponding to the sysgen processors listed above. At the end of each section a number of notes and remarks may be given, which the user must carefully read before starting the operation.

OPERATION

The minimum configuration required for generating a BOM is:
- CPU with 16k word memory
- typewriter with address 10
- papertape reader and punch.

During the GENMON phase, the configuration, file code assignments and levels are defined, as well as the device addresses. It is during this phase that the flexibility of this system generation procedure shows most clearly, in the great number of definition possiblities, e.g. for the user's system medium.
However, the system generation will handle a set of built-in standards if these are acceptable to the user. In this case the user indicates to GENMON that he accepts the standard definitions and does not have to define anything himself. In the other case he must alter the standards during GENMON.
These standard definitions are described in the section GENMON.

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

T o start the process:
- switch on the CPU
- switch the punched tape reader on, put the tape with IPL and GENMON in the reader and make it operable
- set the data switches on the CPU control panel to allow the bootstrap to load this tape from the 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 these bits, see page 42; suffice it to mention here that bit 3 is set to 1 to indicate that the reader is connected to Programmed Channel, and bits 10 to 15 contain the device address.)

- press the MC button
- press the IPL button

Now the bootstrap is loaded, which starts loading the punched tape. The first program on the tape is the Initial Program Loader (IPL), which immediately loads the first system generation program into memory:

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

(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:   (answer NO, for no discs used in the configuration)
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 typewriter)
- 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 input)

If the user replies N or NO to this second 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 is the same for all monitors, some of the file codes given in this list are irrelevant for the generation of a BOM and the user must type in NO after those); since the user receives his sysgen processors on punched tape, it will be clear that in any case he can't answer YES to STANDARD FILE CODE ASSIGNMENT?, for it means that he will have to redefine file code /4 to the tape reader. Whether any other file codes need to be redefined depends on whether he wants his sysgen output on punched tape, cassette tape or magnetic tape and also on whether he wants his own system medium to be punched tape, cassette tape or magnetic tape. So the user types NO after this question and GENMON outputs the following list:

```
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 DEV22                    F.C./7:  (standard = TK25)
AUX. LKE INPUT DEV 3                    F.C./8:  (standard = PR20)
AUX. LKE INPUT DEV 4                    F.C./9:  (type in NO)
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: (type in NO)
```

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

- (CR) if he wants the standard assignment (see above); it is
- <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 encountered in the standard list above, e.g. TK05, it suffices to specify only the device name, e.g. 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: BOMGEN

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 (especially from cassette or magnetic tape) 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 ON/RTC or LOCK position. For PANEL LEVEL it is also mandatory.

For the question STANDARD FILE CODE ASSIGNMENTS:

- The standards imply sysgen from cassette tape. Therefore, for BOM system generation the answer to this question must be NO and the standards must be redefined.
- 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 punched tape.

- file code /A: from this file code the parameters for the BOM table generation (BOMGEN) will be read. This may be done in interactive mode (e.g. TY) or not (e.g. punched tape reader, cassette tape, magnetic tape or card reader).
- file code /B: this may be punched tape, cassette tape or magnetic tape.
- AUX. INPUT DEV.: these may be assigned in advance, especially for sysgen if libraries are to be scanned on various devices.
- file code /3: this is the main output device (sequential), i.e. punched tape, cassette tape or magnetic tape.
- file code /2: for logging of the sysgen operation: LP. If no line printer is available, type in NO.
- File codes /4, /A, /B, /3 are mandatory; file codes /6 up to /9 and /2 are optional.

- When answers to GENMON questions are given on an ASR typewriter, 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.

BOMGEN

When GENMON initialization is terminated and the message READY TO LOAD PROGRAMS has been output the user can start the second phase, BOMGEN, the building of the BOM tables. This is done by typing in replies to questions output on the typewriter by BOMGEN. From these replies BOMGEN builds the tables and records them on the medium with file code /B, i.e. if the user has not redefined this file code, on cassette drive TK15, otherwise it may be the tape punch or magnetic tape, according to the user's definition under GENMON.
(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 and then the user must, after loading BOMGEN into memory, put this pre-recorded tape on the reader.)

In any case, the BOMGEN phase is started as follows:

- put the BOMGEN tape on the tape reader and make it operable
- push the INT button on the CPU control panel
- on output of M: type in

  LD
- now BOMGEN is loaded from the tape reader into memory and its identification is output:

  IDENT BOMGEN
- when loading is terminated,

  :EOS

  :EOF

  is output on the typewriter.
- prepare the tape punch if this has been assigned to file code /B under GENMON: switch it on and feed the tape. If /B has not been redefined, put a cassette in drive TK15.
- if you are working in non-conversational mode, prepare the BOMGEN command input device, after having taken off the BOMGEN tape
- push the INT button on the CPU control panel
- on output of M: type in

  ST
- now BOMGEN is started and outputs the following messages 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 character 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.

MEMORY SIZE

Reply: Specify, with two hexadecimal characters, the size of the machine memory in 8k modules. The number must be a multiple of 8k and may not exceed /20.

Example: 10: 16k memory.

Error Messages: TG03: the specified number is not hexadecimal.
TG04: the size is not a multiple of 8k
TG05: the size exceeds 32k words.

STACK SIZE

Reply: Note, that each device requires up to 20 words in the stack for each interrupt managed for that device. Specify up to 4 hexadecimal characters, giving the size in words, of the stack which is used by the system to save registers when an interrupt occurs.

Error Message: TG03: the number specified is not hexadecimal.

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 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 in 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.

REAL TIME CLOCK LEVEL

Reply: Specify the level for the real time clock, in one or two hexadecimal digits.

LKM LEVEL

Reply: First specify the level to which the LKM interrupt is connected in the same manner as for power failure and real time clock. Then, define the standard monitor requests necessary for your system. TABGEN prints the names of these monitor requests one by one, and after each one the user can answer:
Y if the monitor request is required
N if it is not wanted.
All monitor requests are optional.
The following list of standard monitor requests is printed by TABGEN:

IORM	(Input/Output Requests)	necessary for standard system
WAIT	(Wait for an Event)	
EXIT	(Exit)	
GTBUF	(Get Buffer)	required with FORTRAN
FRBUF	Release Buffer)	

PSE (Pause) (recommended for cassette and magnetic tape handling)

ABRT (Control Abort) (required for Debug)

Note: If the user issues a request for a non-existing function, later on in a program, the value-1 will be returned in register A7.

Error Message: TG06: the reply was neither Y nor N. The user must type the correct reply.

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 consits of two hexadecimal digits defining the DATA ⟨number⟩ which follows the LKM instruction, and ENTRYi is the symbolic entry point of the routine processing the LKM DATA Ni function.
The system will extend the monitor request table in which word i contains the address of ENTRYi. Therefore, during SYSLINK, the user must provide the Linkage Editor with the module containing all entry points ENTRYi specified here.
END indicates that all user LKM functions have been declared or that none are wanted.

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.

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: parameter error.

OCOM OPTIONS

Reply: Specify if the operator communication package must be included in the system by typing:

Y if it must be included, or

N if it must not be included.

If the reply was Y, DOMGEN will print, one by one, the operator commands available. By typing Y or N after each one, the user indicates whether he wants it to be included or not.

The following list is output:

DM (Dump Memory)

MC (Manual Control) (required for magnetic tape and cassette handling)

HD (Halt Dump) (recommended if DM has been selected)

WM (Write into Memory)

LD (Load a Program) } (required for Debug)

ST (Start a Program) }

RY (Retry Device) } (see Note below)

RD (Release Device) }

AS (Assign File Code)

AB (Abort a Program)

PS (Pause)

RS (Restart) (required if PS or PSE (LKM) selected)

Note: When the commands RY and RD are not selected, the system will stop when an I/O error occurs which requires operator's intervention. In a configuration with only a typewriter these two commands are not needed, but if more devices are used, they are highly recommended.

USER COMMANDS

Reply: Specify any user-made operator commands which must be included in the system, as follows:

```
<CM1>,<ENTRY1>
  |       |
  |       |
<CMn>,<ENTRYn>
    END
```

where CMi consists of 2 characters defining the operator command and ENTRYi specifies the entry point of the module which processes command CMi.
END indicates that all commands have been declared or that none are required.

Note: CMi must not be equal to any of the standard operator command mnemonics.

Error Message: TG03: syntax error.

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 opertor's typewriter, three devices (TY, TP, TR) must be declared with the same address, if they will all be 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.
TG06: missing parameter.

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:
  LPDA,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.

Device names used:

TR : ASR tape reader
TP : ASR tape punch
PR : high-speed reader
PP : high-speed punch
TY : operator's typewriter
CR : card reader
LP : line printer
TK : cassette tape unit
MT : magnetic tape unit

HIGHEST FILE CODE NUMBER

Reply: Specify the highest file code number (from /01 to /FF) used in any of the programs. This will indicate how many words the system must reserve for the File Code Table.

Error Message: TG03: syntax error.

SPECIFY FILE CODES

Reply: Specify all file codes used by the programs. If system processors are used, their standard file codes must be declared. The standard file codes are: 01:source input, 02:listing output, 03:punch output, 04:object input, 05:operator's typewriter. 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 code 05 is used by the system for input/output to/from the operator's typewriter. It cannot be assigned or reassigned by AS operator commands.
So, if used, it is necessary to declare at least this file code. All other file codes may be assigned at execution time by means of the operator command AS.

Error Message: TG08: syntax error
TG10: device has not been declared.

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 SYSLINK 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. 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.

Error Messages: TG03: syntax error, e.g. the file code is not a hexadecimal number.

TG07: the first character of the DWT is not an alphabetical character.

ABORT MODULE (This question is printed only if N was replied for AB under OCOM OPTIONS and ABRT under LKM LEVEL)

Reply: If neither the monitor request 'Keep Control on Abort' nor the operator command AB have been selected, the user may include the system abort module by typing Y. If the reply is N, the abort module will not be included. Abort of the user program will, if the module has been selected, result in an output message and the user may enter a new command to run the next job. If the module has not been selected, an abort will stop the machine, through a Halt instruction.

Error Message: TG06: invalid reply.

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 and P857 the user must type N.

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.

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.

Note: This is not the maximum number of scheduled labels used in the program, which may be a higher number.

TABGEN ENDED

IPLGEN

During this phase an Initial Program Loader for the user's system will be generated. This implies, that at this point he must prepare his syetem medium. In standard cases, the output from IPLGEN would be onto the cassette in drive TK05 (file code /3 under question STANDARD FILE CODE ASSIGNMENTS of GENMON phase), but the user may have reassigned it to the tape punch during that phase. In any case he must put the tape containing IPLGEN on the tape reader.

Now IPLGEN can be loaded and started:

- push the INT button on the CPU control panel
- on output of M: type in LD
- now the next sysgen processor is loaded from the Paper tape reader and its identification typed out:
  IDENT IPLGEN
  when loading is terminated,
  :EOS
  :EOF
  is output
- push the INT button
- on output of M: type in ST
- Now the IPLGEN processor is started and an Initial Program Loader (IPL) is written at the beginning of the user's system medium.

Note:

- When the IPL is generated onto cassette or magnetic tape, the positioning of this tape must remain stable until the user's monitor which must follow it is recorded onto it. Remember that a cassette is rewound when taken out of the drive and put back in again!

GENLKE

During this phase the final user monitor is obtained by linking the tables generated under BOMGEN with the monitor modules required from the BOM Library on the BOMLIB tape, or a User Library Tape or Extension Tape (see Note at end of section).
First GENLKE must be loaded into memory:

- take the tape containing the output from the BOMGEN phase from the tape punch, if it was punched out on paper tape
- put the GENLKE tape on the tape reader and make it operable
- push the INT button on the CPU control panel
- on output of M: type in
  LD
- now GENLKE is loaded into memory from the tape reader and its identification is output on the typewriter:
  IDENT GENLKE
- when loading is terminated
  :EOS
  :EOF
  is output on the typewriter.

- now put the tape generated under BOMGEN on the tape reader and make it operable (or, if the tables were generated on the cassette in TK15, take the cassette out of TK15, put it in TK05 and wait for it to be rewound; all this depends on the file codes assigned under GENMON)
- prepare the tape punch and feed the tape if paper tape is designated as the system medium (file code /3, see GENMON) or prepare the relevant device if the system medium is cassette or magnetic tape
- push the INT button on the CPU control panel
- on output of M: type in
  ST
- now 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 3 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 Basic Operating Monitor

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

If the file codes are going to be used as defined under GENMON (file codes /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 produced under BOMGEN are now recorded on the system medium, i.e. punched tape, cassette tape or magnetic tape and <module name> is output on the print file.

- when this is finished, take the tape with the tables generated under BOMGEN from the reader and put the BOMLIB tape on
- in response to the

L:

output by GENLKE, now type in

L

upon which GENLKE will start scanning the BOMLIB tape to select the required modules and record them onto the user's system medium (either the punched tape or the cassette in TK15 or the magnetic tape; see above). The names of the selected modules are output on the print file, 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. The last module to be included must be INIMON, so if GENLKE types out

INIMON
after the user has typed his first U, it is correct. The GENLKE processor then types out
L:
and the user can type
L
to solve this last unsatisfied reference.
(If there were any 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 BOM on the print file and on the typewriter it outputs monitor length (L=XXXX), monitor start address (S=XXXX) and the first free location after the monitor in memory (E=XXXX).

Note:
In response to an L: output by GENLKE the user may type in an H to put this processor in pause state. This is very useful if the input file code for the GENLKE processor must be changed, because to restart GENLKE the message RS XX may be given, where XX is the new input file code.

Note:
If modules from other tapes beside the BOMLIB tape must be link-edited during this phase, the tapes must be csanned in a defined order, which is:

- User Library Tapes
- Extension Tape(s)
- BOMLIB tape

When more references than INIMON remain unsatisfied, rescanning must start at the first step in this sequence.

Now the system generation process is finished and the user has his own monitor on punched tape, cassette tape or magnetic tape, depending on which one he had chosen as his system medium during the GENMON phase.

Programming Rules

With BOMGEN, the user can combine his own routines with the standard system routines. This entails that the interfaces must not be modified and that the routines must be written in accordance with these interfaces.
With respect to this, some guidelines are given below.

- *Interrupt Routines*

- If the routine does not reset any event and does not have to run in enable mode, any registers can be saved in the A15 stack. They will have to be restored at the end of processing, before returning to the interrupted routine via A15.
  Example:
  An interrupt routine running in inhibit mode, using 3 registers:
  Save 3 registers in A15 stack:
  Reset interrupt: RIT,INR,OTR....
  Process.
  Restore 3 registers from A15 stack:
  Return to interrupted context: RTN A15

  For P856/7 or for P852 with simulation routines:
  MSR 8,A15
  MLR 8,A15
- If the routine resets an event or has to run in enable mode, it is mandatory to have stored in the A15 stack the P-register, PSW and the first 8 registers before switching to enable mode. Moreover, at the end of processing a branch must be made to the dispatcher without restoring the registers.

If no label is scheduled, A6 must be 0 and A5 contents is irrelevant.
If a label is scheduled, A6 must contain the scheduled label address and A5 the Program Control Table address.

For example:
An interrupt routine resetting an event:
Save 8 registers in the A15 stack:
Reset interrupt: RIT,INR....
Process.

(For P856/7 or P852 with simulation routines:
MSR 8,A15

```
                    LDKL   A6,SCLAB
Do not restore registers.
                    LDKL   A5,PCTAD
Branch to dispatcher:
                    ABL DISPAT
```

Note:
- DISPAT,SCLAB,PCTAD must have been defined as an external. The dispatcher address can be found in the Communication Vector Table.
- It is always mandatory to reset the interrupt signal by executing a special instruction.

- *I/O Drivers*

When an I/O request is given, the monitor will call the IORM module to process the request. The control will be transferred to the driver, the start address of which is stored in the Device Work Table (DWT).
- The driver runs at level 48.
- Register A6 contains the DWT address.
  Register A7 the I/O order.
  Register A8 the ECB address.
- Having initialized the I/O, the driver must return to the dispatcher by means of:
  ABL C:WAIT
  with A7 containing the I/O order and A8 the ECB address.

- Regarding I/O interrupts, the driver must respect the rules laid down for interrupt routines.
- When the I/O has been completed (after the SST command), the driver can switch to level 48 as follows:
  A1 must contain the return address, then:
  AB L:VCH
- The driver can return to the ENDIO module by branching to the appropriate entry point:
  R:TUR2: first post-edit, then return to calling program.
  Status of the I/O operation can be found in register A2.
  R:TUR4: return to calling program.
  Status of the I/O operation can be found in register A2.
  R:TURN: return to the interrupted program.

  M:RETR: output an error message. If:
  A1=0: print the message with RY.
  A1≠0: print the message without RY.

- *User-written Monitor Requests*

The module processing the user monitor request must be included in the system at SYSLINK time; thus, the entry point is added to the monitor table. When the request is made, it is processed as follows:
- the monitor saves register A1 to A8 of the calling program in the A15 stack.
- the LKM interrupt is reset.
- the module processing the monitor request is called via a branch instruction.
- when the processing routine is called:
  A6=0: if no scheduled label is used.
  A6≠0: then it contains the address to be scheduled when the event occurs.
  A5 must then contain the PCT address.
- the processing routine has to return to the dispatcher by ABL DISPAT where DISPAT must have been declared as an external.

The processing routine is thus called by the system at the level of the LKM. If the user wants to switch to a lower level, i.e. that of the monitor, he must give the following instructions:
LDKL A1,48 or 49 (monitor level)
CF A15,CHLEV
Where CHLEV must have been declared as an external.

- *User-written Operator Commands*

Operator commands are activated by control panel interrupt. The control panel routine processes the interrupt, saves registers and decodes the first two characters of the command (i.e. the mnemonic). Then control is transferred to the processing routine, which must follow these rules:
- the routine runs at level 49.
- the message is read into buffer BUFCP, consisting of 80 characters.
- A5 contains the address of a flag, used as follows:
  If 0: accept new operator command.
  If 1: do not accept DM and MC commands.
  This flag is reset to zero when the routine is called, so the routine must set the flag to 1, by means of IMR A5.
- the routine must return to the label CPTRN which must have been declared as an external.

  Note: All input/output must be performed without wait.

Comment Sheet

P800M Programmer's Guide 1, Volume I (5122 991 27322)

Name _____

Company _____

Department _____

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