27.5 [illegible]

February 1976

A publication of:

Philips Data Systems B.V.
Marketing Group Small Computers
P.O. Box 245 Apeldoorn, The Netherlands

5122 991 27431

Printed in The Netherlands

PHILIPS

0.25

P800M

Programmer's Reference Data

PHILIPS Data Systems

CONTENTS

Page

IPL control information 2
Standard device addresses 2
I/O functions 2
File codes 3
Status word (ECB word 4) 3
I/O Functions 4
LKM monitor requests 5
C.U. status word configuration 6
Data Commmunication 7
I/O requests DATEM 7
DATEM ECB 7
Hexadecimal format of characters 9
Monitor operator messages 10
CCI commands 11
SCL commands DRTM 21
Disc processor messages 25
Basic processor messages 27
Directives 32
Instruction set 33
System messages 53
Standard interrupt levels 64
Program status word 64
Format of RER/WER instructions 65
Powers of 16 65
Powers of 2 65
AND, OR, XOR 66
Internal hexa 66
Hexadecimal-decimal conversion 67
ASCII code 68

INFORMATION ON DATA SWITCHES BEFORE PRESSING IPL BUTTON

bit 0	= 1	IPL is read from ASR reader (4 x 4)
	0	IPL is loaded from other device
1	= 1	IPL is loaded from disc
	0	IPL is loaded from other device
2	= 1	moving head disc
	0	fixed head disc
3	= 1	programmed channel
	0	I/O processor

4 to 7		control information for control unit during execution of CIO Start sent by the bootstrap:
		TY = 0001 MT = 0010
		TK = 0111 DK = 0011

8	= 1	multiple device control unit
	0	single device control unit
9	= 1	P824-001 CDD disc
10-15		device address of device from which IPL is loaded

STANDARD DEVICE ADDRESSES

/00 = reserved	/02 = disc 1	/03 = disc 1	/06 = CR
/01 = disc 1	/12 = disc 2	/13 = disc 2	/07 = LP
/11 = disc 2	/22 = disc 3	/23 = disc 3	/10 = ASR
/21 = disc 3	/32 = disc 4	/33 = disc 4	/20 = PTR
/31 = disc 4			/30 = PTP
			/0E = Adapter
			/0F = Adapter
/04 = MT1	/05 = CAS 1	/08 = SLCU2/4	/16 = AMA
/14 = MT2	/15 = CAS 2	/09 = SLCU2/4	/26 = AMA
/24 = MT3	/25 = CAS 3	/0A = SLCU2/4	/36 = AMA
/34 = MT4		/0B = SLCU2/4	
		/0C = SLCU2/4	

DUMP PROGRAM

The IPL is loaded by pushing the IPL button on the control panel.
The user may now specify in register A9 the loading address of the dump program. Default = 0000.
Next push the RUN button to load the dump program. When the reading stops the user must load registers A8, A9 and A10 with the following information:

A8 to be loaded with the address of the device onto which the dump will take place, e.g. /10 for the operator's typewriter or /07 for the line printer. If the device is connected to the programmed channel, bit 0 of register A8 must be 1
A9 first address of area to be dumped
A10 1st address of this area.

Press RUN button to activate the dumping.

FILE CODES

Standard file codes – basic system

/01	= standard source input
/02	= standard listing output
/03	= standard punch output
/04	= standard object input
/05	= operator's keyboard

Standard file codes – disc system

/01	= operator's typewriter
/02	= print unit
/03	= punch output
/04-/09	= reserved for peripheral devices
/D0	= catalogued procedure input
/D3	= reserved for system use
/D4	= /S or library source file (Line Editor output)
/D5	= /O (ASM output, LKE input)
/D6	= /L (LKE output)
/D7	= system object file (library)
/D8	= user object file (library)
/D9-/DF	= reserved for system use
/E0	= control command input
/E1	= source input
/E2	= object input
/EE	= catalogued procedure output
/EF	= system typewriter
/F0-/FF	= disc units' logical addresses

STATUS WORD (ECB word 4)

For Basic orders: If /0000 normal I/O completion
If bit 0 = 0 remaining bits give C.U. status
1 remaining bits give software status
(see below)

For Standard orders: If /0000 normal I/O completion
If bit 0 = 0

bit 6	= 1	unknown file header label	CFM
bit 7	= 1	no data on cassette	
bit 8	= 1	wrong labelling	
bit 9	= 1	end-of-file set	
bit 8	= 1	End-Of-Volume mark	
bit 9	= 1	End-Of-Tape mark	
bit 10	= 1	Beginning of tape	
bit 11	= 1	End of input medium (disc only)	
bit 12	= 1	Requested length is incorrect	
bit 13	= 1	Illegal character code or checksum error	
bit 14	= 1	EOS mark	
bit 15	= 1	EOF mark	

If bit 0 = 1
1 = 0
bits 2 thru 15 give C.U. status

If bit 0 = 1
1 = 1
bits 2 thru 15 give software status

DRTM		bit 2	= 1	power failure
		5	= 1	disc overflow (no more granules available)
		6	= 1	no disc buffer available (dynamic allocation area overflow)
		7	= 1	disc queue overflow
		10	= 1	file is write-protected
	BOM DOM BRTM	11	= 1	function unknown or not compatible with device
		12	= 1	buffer size is illegal
		13	= 1	buffer address is illegal
		14	= 1	device attached to other program
		15	= 1	illegal file code or non-existing

I/O FUNCTIONS	BOM	BRTM	COM	CFM		DOM	DRTM	MAM
				basic	ext cmp			
/01 Basic Read	X	X	X	X	X	X	X	
/05 Basic Write	X	X	X	X	X	X	X	
/02 Standard Read	X	X	X	X	X	X	X	
/06 Standard Write	X	X	X	X	X	X	X	
/07 Object Write (4x4)	X	X	X			X	X	
/08 Object Write (8+8)	X	X	X	X	X	X	X	
/0A Random Read						X	X	
/0B Random Write						X	X	
/14 Skip forward to EOS	X	X				X	X	
/16 Skip forward to EOF	X	X	X	X		X	X	
/21 Get type of labelling					X			
/22 Write EOF mark	X	X	X	X	X	X	X	
/23 Write file header label					X			
/24 Write EOV mark	X	X	X			X	X	
/26 Write EOS mark	X	X	X	X	X	X	X	
/27 Search file header label					X			
/29 Enable access for labels					X			
/2A Inhibit access for labels					X			
/30 Get information about file codes	X	X				X	X	
/31 Rewind to load point	X	X	X	X	X	X	X	
/33 Backspace one block	X	X	X	X	X	X	X	
/34 Space one block forward	X	X				X	X	
/36 Skip backwards to EOF	X	X				X	X	
Search backwards to tape mark			X	X				
Search for first file					X			
/37 Search for next file header label					X			
/38 Unlock (TL, TK)	X	X	X	X	X	X	X	
Off-line (MT)	X	X						

LKM MONITOR REQUESTS	BOM	DOM	SRTM	BRTM	DRTM	COM
1 I/O	X	X	X	X	X	X
2 Wait for an event	X	X	X	X	X	X
3 Exit	X	X	X	X	X	X
4 Get Buffer	X	X		X	X	X
5 Release Buffer	X	X		X	X	X
6 Pause	X	X				X
7 Keep control on abort condition	X	X				X
8 DATEM	X	X		X	X	
9 Load a segment		X				
10 Connect a program to a timer			X	X	X	
11 Disconnect a program from a timer			X	X	X	
12 Activate program			X	X	X	
13 Switch inside a software level				X	X	
14 Attach a device to a program			X	X	X	
15 Detach a device from a program			X	X	X	
17 Get time				X	X	
18 Reset an event				X	X	
20 Connect a program to a level				X	X	
21 Disconnect a program from a level				X	X	
22 Wait for a given time				X	X	
23 Assign file code					X	
24 Delete file code					X	
25 Read unsollicited operator message					X	
26 Cancel LKM 25					X	

C.U. STATUS WORD CONFIGURATION

Bit	Description	CU								
		ASR	CR	MHD	LP	TP	PTR	CASS Tape	FHD	MT
0	–									
1	ready			x				x		x
2	rewind									x
	tape mark has been read							x		x
4	no data							x		
	on cylinder load point			x				x		x
6	seek error			x						
	write unable							x		x
7	A or B side							x		
8	Device Address							x	x	x
9	Device address			x				x	x	x
10	EOT						x	x		
	tape low					x				
11	Program error			x				x	x	x
12	Incorrect length		x	x				x	x	x
13	Parity error							x		
	Data fault		x	x					x	x
14	throughput error	x	x	x			x	x	x	x
15	not operable	x	x	x	x	x	x	x	x	x

ASR	I/O typewriter	TP	tape punch	FHD	fixed head disc
CR	card reader	PTR	punched tape reader	MT	magnetic tape
MHD	moving head disc	CASS	cassette tape		
LP	line printer				

DATA COMMUNICATION

LKM = 8 (- 8, <sched lab>)

I/O requests DATEM

/1 read with echo, without time-out
/2 read without echo, without time-out
/3 read with echo, with time-out
/4 read without echo, with time-out
/6 write a block, without time-out
/7 write a block with time-out
/D change line definition
/E get line definition
/10 stop the exchange
/11 disconnect the line
/12 search pattern
/13 wait for call
/14 accept data
/15 set time-out

DATEM ECB

The ECB address must be loaded in register A8.
The structure of the ECB is:

	0	1 – 7	8 – 15
WORD 0	E	RESERVED	LOG.LINE NO
WORD 1	BUFFER ADDRESS		
WORD 2	BUFFER LENGTH		
WORD 3	TRANSMITTED LENGTH		
WORD 4	SERVICE STATUS		
WORD 5	TERMINATOR TABLE ADDRESS		
WORD 6	TIME-OUT VALUE		

The status word has the following format:

bit 0 = the specified transmission line is busy
1 = the transmission line is not connected (on switched network)
2 = the logical line number specified in word 0 is not correct
3 = illigal request: register A7 contains a wrong service number
negative block length
4 = character(s) lost. this bit is set in an asynchronous transmission when the user has not provided a receive buffer in time. see also request /14.
5 = end-of-carrier detection
6 = the time-out request cannot be serviced as the time control table declared at system generation is too short.
7 = buffer overflow, the reserved buffer is too small
8 = the transmission stopped
9 = power failure in the data communication equipment
10 = the time-out as specified in word 6 has elapsed
11 = break detection (interruption of input stream by remote station). only valid for asynchronous transmissions
12 = the service is not accepted
13 = parity error (hardware detection)
14 = throughput error. this bit is only set in input
15 = modem not operable

/D Change line definition

ECB word 3: bits 0 trough 4 may not be changed
bit 5 1 = IBM CRC
0 = CCIT CRC or no CRC
bit 6 1 = odd parity
0 = even or no parity
bit 7 1 = hardware parity check
0 = no hardware parity check
bit 8 1 = CRC computed by hardware (SLCU2S)
0 = CRC not computed by hardware
bits 9 and 10 may not be changed
bit 11 1 = automatic SYN generation
0 = no automatic SYN generation
bit 12 1 = the line is always connected (leased line)
0 = the line is not always connected
bit 13 1 = hardware character check inhibited (SLCU2S)
0 = hardware character check not inhibited
bits 14, 15 not significant

ECB word 5: bits 0 through 4. A number as given to the special character table at system generation time. All zeroes if no special table requested
bits 5 through 7. Pointer to the table in which the different SYN values are kept (see SYNTAB description)
bit 8 1 = higher transfer rate of the modem
0 = lower rate of the modem
bit 9 1 = character synchronization requested
0 = character synchronization inhibited
bits 10 through 15 may not be changed

/E GET line definition

ECB word 2: bit 0 = 1 EBCDIC code
0 ASCII code
this bit is only of importance when working with the SLCU2S
bits 1 through 3 Number of lines connected ot the AMA8A/C
bits 4 through 7 Number of bits per character
bits 8 through 15 Interrupt level of the line control unit (0 through /3F)

ECB word 3: bit 0 = 1 the line is busy
0 the line is not busy
bits 1 through 4 are not used
bit 5 = IBM CRC
0 CCITT CRC or no CRC
bit 6 = 1 odd parity
0 even or no parity
bit 7 = 1 hardware parity generation
0 no hardware parity generation
bit 8 = 1 CRC computed by hardware
0 CRC not computed by hardware
bit 9 = 1 this is a full duplex line
0 this is a half duplex line
bit 10 = 1 control unit is conneted to the I/O processor
0 control unit is connected to the Programmed Channel
bit 11 = 1 automatic SYN generation
0 no automatic SYN generation
bit 12 = 1 the line is always connected (leased line)
0 the line is not always connected
bit 13 = 1 hardware character check inhibited (SLCU2S)
0 hardware character check not inhibited

bits 14 and 15 00 SLCU2S
01 SLCU4
10 ALCU2 or ALCU4
11 AMA8A or AMA8C

ECB word 5: bits 0 through 4. Number as given to a special character table at SYSGEN time and used by this line
bits 5 through 7. Pointer in the SYNTAB table for the SYN value used on this line
bit 8 = 1 higher transfer rate of the modem
0 lower transfer rate of the modem
bit 9 = 1 character synchronization requested
0 character synchronization inhibited
bits 10 through 15. Device address of the line control unit (0 through /3F)

HEXADECIMAL FORMAT OF FRAMING CHARACTERS AND SPECIAL CHARACTERS

	ASCII	**EBCDIC**
SYN	/0016	/0032
STX (TTD)	/0002	
SOH	/0001	
DLE-STX	/1002	

ASCII non transparent - without parity

RVI (DLE-<)	/103C	ETB	/0017
TTD (STX-ENQ)	/0205	ETX	/0003
ENQ	/0005	NAK	/0015
EOT	/0004	ACK0 (DLE-0)	/1030
WACK (DLE-;)	/103B	ACK1 (DLE-1)	/1031
		ITB	/001F

ASCII transparent - without parity

RVI (DLE-<)	/103C	ETB (DLE-ETB)	/1017
TTD (STX-ENQ)	/0205	ETX (DLE-ETX)	/1003
ENQ (DLE-ENQ)	/1005	NAK (DLE-NAK)	/1015
EOT (DLE-EOT)	/1004	ACK0 (DLE-0)	/1030
WACK (DLE-;)	/103B	ACK1 (DLE-1)	/1031
		ITB (DLE-ITB)	/101F

EBCDIC non transparent

RVI (DLE-@)	/107C	ETB	/0026
TTD (STX-ENQ)	/022D	ETX	/0003
ENQ	/002D	NAK	/003D
EOT	/0037	ACK0 (DLE-)	/1070
WACK (DLE-’)	/106B	ACK1 (DLE-/)	/1061
		ITB	/001F

EBCDIC transparent

RVI (DLE-@)	/107C	ETB (DLE-ETB)	/1026
TTD (STX-ENQ)	/022D	ETX (DLE-ETX)	/1003
ENQ (DLE-ENQ)	/102D	NAK (DLE-NAK)	/103D
EOT (DLE-EOT)	/1037	ACK0 (DLE-)	/1070
WACK (DLE-')	/106B	ACK1 (DLE-/)	/1061
		ITB (DLE-ITB)	/001F

MONITOR OPERATOR MESSAGES

BOM - DOM - COM

Message	Meaning
AB	Abort a program
AS⊔<file code>⊔<device name><address>	Assign a file code
DM⊔<address1>⊔<address2>	Dump memory
HD	Halt dump
LD⊔⌈<value>⌉ ⌈⊔M⌉	Load a program (BOM only)
MC⊔<file code>⊔<order>⌈⊔<rept fact>⌉	Manual device control
PS	Pause
RD⊔<device address>	Release an operation from device
RS⊔⌈<new A7>⌉	Restart a program
RY⊔<device address>	Retry an I/O operation
ST	Start a program (BOM only)
WM⊔<address>⊔<value1>⌈⊔<value2>..<value n>⌉	Write into memory

DRTM

CC	Activate SCL
CR⊔<file code>	enter correction records from this file code. Format correction record: <address>,<value1>⌈,<value2>.. ..,<value n>⌉
DD⊔<disc number>,<sector1>⌈,<sector2>⌉	Dump disc
DM⊔<address1>⌈,<address2>⌉	Dump memory
HD	Halt dump
HT	Stop CPU activity
RD⊔<device address>	Release I/O operation
RY⊔<device address>	Retry I/O operation
WM⊔<address>,<value1>⌈,<value2>..<value n>⌉	Write memory

COM

WF⊔⌈</file code>,⌉⌈<file identifier> ⌈,<option>⌉⌉	Write first header
WH⊔⌈</file code>,⌉ <file indentifier> ⌈,<option>⌉	Write header
RN⊔⌈</file code>⌉⌈,<name>⌉	Run program
SH⊔⌈</file code>⌉⌈,<name>⌉	Search header
CF⊔⌈</file code>⌉	Clear file

Note: *For other monitor/operator messages, see also BOM, DOM and COM*

CCI COMMANDS

Assign

use: assign a file code to a peripheral unit, a disc file or a temporary area on disc.

syntax: ASG⊔/<file code 1>⌊,/<file code 2 >|< device name >< device address>⌋ ⌈,<name>⌈,<userid>⌈,<disc number>⌈, NP⌉⌉, NP⌉⌉⌉

where:
<file code 1> = file code to be assigned
<file code 2> = an assignment previously made for this file code is also to be made for <file code 1>
<device name><device address> = <file code 1> is to be assigned to this peripheral unit (e.g. /E1,CR05). DK does not require an address
<name> = only used when DK is specified in <device name>. It specifies the name of the library file to which <file code 1> must be assigned. If <name> is not specified in this case, <file code 1> is assigned to a temporary file.
<userid> = only used when <name> is specified. With <disc number> a file code may be assigned to a file in an other user's library on the specified disc.
<disc number> = file code /F0 to /FF
NP = the assigned file is write protected unless NP is specified

End of session

use: in conversational mode: command to be given to terminate a user's session unless the parameter BYE is specified which will switch the system to batch processing mode.
in batch processing mode: command to terminate a job. The system looks for a new job. If the parameter BYE is specified the system will switch to conversational mode.

syntax: BYE⊔⌈BYE⌈,<DNDA>⌉⌉

Declare User

use: add a new user identification. This command can only be used in a system session.

syntax: DCU⊔<userid>,/<disc number>
where:
<userid> = new user identification
<disc number> = file code from /F0 to /FF

Delete file

use: delete a file or object module from a library

syntax: DEL␣⌊<name>⌉/OB⌋⌈L/S|,/O|,/L⌋⌉

where:
name = name of file or module to be deleted
/OB = the whole object file of the library must be deleted. In this case /S, /O or /L may not be used.
/S = source file
/O = object file
/L = load file

Delete user

use: can only be used in system session. The specified user is deleted from the catalogue.

syntax: DLU␣<userid>,/<disc number>

where:
<userid> = user identification to be deleted
<disc number> = file code /F0 to /FF

Dump file

use: have a hexadecimal dump on the print unit

syntax: DUF␣⌊/<file code>|/O|/L|<name>| ⌈,<sect nb a>⌈,<sect nb b>⌉⌉

where:
<file code> = file code of file to be dumped
<name> = name of library file to be dumped (type UF)
/O = object file
/L = load file
<sect nb a>⌈,<sect nb b>⌉ = dump of sectors in the specified range

End catalogued procedure

use: this command terminates the catalogued procedure

syntax: END

Include object module

use: select an object module from a library and copy it into the temporary /O file

syntax: INC␣⌊/OBJCT|<name>⌋ ⌈,<userid>⌈,<disc number>⌉⌉

where:
/OBJCT = the whole object library must be copied into /O
<name> = name of object module to be included
<userid> = the object module is to be found in a library of the specified userid
<disc number> = the object module can be found on the disc with the specified number (/F0 to /FF)

Start job

use: start batch processing mode or start a new batch after BYE

syntax: JOB␣⌊<userid>|/<disc file code>,<userid>⌋

where:
<userid> = the system scans each on-line disc catalogue for the specified userid
<disc file code>,<userid> = the system looks for the userid in the catalogue of the specified disc.

Keep file

use: make a file or module permanent by placing it in the library

syntax: KPF␣⌊/S|/O|/L|/<file code>⌋ ⌈,<name>⌉

where:
/S,/O,/L = type of file to be kept
<file code> = file code of file to be kept
<name> = the file or module is placed in the library under this name
Specifying the name is obligatory for /L or <file code>
For /O, and name specified, the specified object module is kept. Otherwise all object modules on the /O file are kept.

List catalogue

use: accepted only in a system session. The catalogue of the specified disc is printed on the typewriter log

syntax: LIC␣/<disc number>

where:
<disc number> = file code /F0 to /FF

List directory

use: the user library is listed on the typewriter log

syntax: LSD␣⌈/OB⌉

where:
/OB = only the names on the object file are listed

List file codes

use: a list of file codes and corresponding devices is output on file code 1

syntax: LSF

List file

use: list the specified disc file on the typewriter log. The file must be sequential and consist of ASCII records

syntax: LST␣⌊/<file code>|/S|/S,<name>|<name>⌋ ⌈,<line nb a>⌈,<line nb b>⌉⌉

where:
<file code> = a temporary data file
/S = a temporary source file
/S, <name> = a catalogued source file
<name> = a catalogued user data file
<line nb a>⌈,<line nb b>⌉ = all lines in the specified range are listed, the both specified included

Send message

use: command especially used in batch processing. It allows to send the specified message to the operator

syntax: MES␣<message to the operator>

Move a file

use: move a file from a library to a temporary /S or /L file or to a file indicated by the file code

syntax: MOV⊔<name>,⌊/S|/L|/<file code>⌋ ⌈,<userid>⌈,<disc number>⌉⌉

where:
<name> = name of the library file to be moved
/S = the file must be moved to /S file
/L = the file must be moved to /L file
<file code> = file code of the temporary file to which the file of name <name> must be moved
<userid> = user identification of the user whose file must be moved to a file of the current user
<disc number> = parameter used together with <userid> if the file to be moved is on an other disc

Define node

use: this command defines a node in a segmented program

syntax: NOD⊔<name>
where:
<name> = up to 6 alphanumeric characters of the name to be given to a node.

Punch a file

use: punch a sequential file on the punch unit. The maximum record length = 132 characters

syntax: PCH⊔⌊/<file code>|/S|<name>|<name>,/S⌋⌈,<new file codes>⌉

where:
<file code> = file code of the temporary user file which must be punched
/S = the source file must be punched
<name> = name of the catalogued user data file to be punched
<name>, /S = name of the catalogued source program
<new file code>= output file code (default = /03)

Punch load

use: punch a load file present in the library or on /L

syntax: PLD⊔<name>⌈,/L,<file code>⌉,<file code>⌉

where:
<name> = name of the load file
/L = the temporary /L file is punched
<file code> = output file code (default = /03)

Punch object

use: punch the temporary object file or a specific module in the library

syntax: POB⊔⌈<name>⌉⌈<name>|<file code>|<name>,<file code>⌉

where:
<name> = name of the library object module to be punched
If <name> is not specified the whole /O file is punched
<file code> = output file code (default = /03)

Print object directory

use: all names in the object library are printed
syntax: POD

Print catalogue

use: can only be used in a system session. The catalogue of the specified disc is printed on the print unit

syntax: PRC⊔ /<disc number>

where:
<disc number> = address of disc (F0 to /FF) whose catalogue must be printed

Print directory

use: print the user's library directory on the print unit

syntax: PRD⊔⌈/OB⌉

where:
/OB = if specified, only the names of the object modules in the object file are printed
If /OB is not specified the user's library directory is printed

Print file

use: print the specified disc file on the print unit. The file must be sequential and consist of ASCII characters

syntax: PRT⊔⌊/<file code>|/S|/S, <name>|<name>⌋ ⌈,<line nb a>
⌈,<line nb b>⌉⌉

where:
<file code> = temporary data file
/S = source file
/S, <name> = catalogued source file
<name> = catalogued user data file
<line nb a>⌈,<line nb b>⌉ = the lines in the specified range are printed, the two specified lines included

Pause

use: send a message to the operator and go to Pause state. To restart the user has to press the INT button and type in RS⊔⌈<new A7>⌉

syntax: PSE⊔<message to operator>

Read data

use: read data and transfer to a temporary user file. The file codes in this command must be in the range from /01 to /EF

syntax: RDA⊔/<disc file code>⌈,/<input file code>⌉

where:
<disc file code> = temporary user file to which the data are transferred
<input file code> = file code from which the date are read.
Default = /E1 source input

Read object

use: copy an object file from an input unit onto the disc as a /O file or as a complement to the /O file (i.e. the /O file was not closed by an EOF). No EOF record is written onto the disc

syntax: RDO⊔⌈/<file code>⌉

where:
<file code> = file code of the input unit from which the object file is to be read. If not specified the object file is read from the standard object input unit

Read source

use: copy a sequential source program file or sequential data file from the source input unit or other sequential input unit onto the disc as a /S file

syntax: RDS⊔⌈/<file code>⌉

where:
<file code> = file code of the input unit from which the file is to be read. If the parameter is not specified the source file is read from the standard source input unit

Replace supervisor

use: can only be used in a system session. The command copies the monitor of one disc onto the disc specified in this command

syntax: RSU⊔<disc number>

where:
<disc number> = file code of disc receiving the new monitor (/F0 to /FF)

Run a program

use: this command starts the execution of a program

syntax: RUN ⊔⌈<name>⌉

where:
<name> = name of the program

Scratch

use: release the user assignments not made permanent

syntax: SCR⊔⌈⌊/S|/O|/L|/<file code>⌋⌉

If no parameters are specified all user assignments are released. If a parameter is specified the user assignments on the specified file are released

Save disc onto magnetic tape

use: copy the content of a disc onto magnetic tape

syntax: SDM⊔<disc number>,/<file code>⌈,CK⌉

where:
<disc number> = file code of the disc to be copied (/F0 to /FF)
<file code> = file code of the magnetic tape
CK = the magnetic tape is rewound and compared to the disc

Define segments

use: define the library program names of program parts used as a segment by a root program. This command must be followed by RUN

syntax: SEG⊔<name list>

where:
<name list> = one or more library program names, separated by commas. The list may not contain more than 15 names

Skip form

use: a number of pages may be skipped on the file code /02.
syntax: SKF⊔⌈<number>⌉
where:
<number> = the number of pages to be skipped (default = 1)

Save disc onto an other disc

use: can only be used in a system session. This command causes the copying of one disc onto an other disc.
The volume label of the disc to which is copied is not destroyed

syntax: SVD⊔<disc number a>,<disc number b>

where:
disc number = file code from /F0 to /FF
SVD⊔/Fx, /F0 is not allowed

Save user files

use: copy all files of the specified user and present on the specified disc into the library of the current user

syntax: SVU⊔<userid>,<disc number>

where:
<userid> = user identification of user whose files are to be copied
<disc number> = /F0 to /FF

PROCESSOR CALLS

Assembler

syntax: ASM⊔⌊/S|<name>⌋⌈, NL⌉

where:
/S = the source program must be assembled from the /S file
<name> = name of source program in library
NL = if specified, no assembly listing

Linkage Editor

syntax: LKE⊔⌈N|S|U⌉ ⌈,M⌉ ⌈⌊,DE|,DS⌋⌉ ⌈,/<address>⌉ ⌈,<start address>⌉

where:
N = no library scanning is desired
S = only the standard library has to be scanned

U = only the user library has to be scanned
(default: both libraries are scanned, the user library first)
M = the map is printed. Default: no map
DE = entry point and internal symbols are saved
DS = only the internal symbols are saved
<address> hexa displacement value of blank common from beginning of load module
<start address> = name of start address defined as an entry in one of the module in the /O file

Line Editor

syntax: LED␣<name>⌊⌈,<file code 1>⌈,<file code 2>⌉⌉ | ⌈,/S⌈,</file code 2>⌉⌉⌋⌈,XX⌉

where:
<name> = name of source module or user data file to be edited
<file code 1> = output file code for edited file
<file code 2> = file code from which the input commands are read
XX = the specified characters must preceed the LED message

Debugging Package

syntax: DEB␣⌈<name>⌉

where:
<name> = name of module to be debugged

Full FORTRAN Compiler

syntax: FOR␣⌊/S|<name>⌋ ⌈, NL⌉

where:
/S = the program must be compiled from the temporary /S file
<name> = name of the program to be compiled
NL = if specified, no listing is given of the compiled program

Disc FORTRAN Transcoder

syntax: TCD

High speed FORTRAN

syntax: HSF␣⌊/S|<name>⌋⌈,NL⌉
see FOR

User processor

syntax: UPR␣<proc. name>,⌊/S|<file name>⌋⌈,NL⌉
where:
<proc. name> = user-made processor see also ASM.

Overlay Linkage Editor

syntax: OLE␣⌈N|S|U⌉⌈,M⌉⌈⌊,DE|,DS⌋⌉⌈,/<address>⌉⌈,<entry point>⌉
where:
N = no library scanning is desired
S = the standard library must be scanned
U = the user library must be scanned (default: scan both libraries)

M = map is printed. (default: no map)
DE = entry point and internal symbols are saved
DS = internal symbols are saved
/<address> = absolute hexa address of blank common
<entry point> = start address defined as entry point in the root.
For a non-segmented program it is the entry name of a start address.

MAGNETIC TAPE AND CASSETTE TAPE COMMANDS (CCI)

File backwards space

use: space tape backwards across the previous tape mark or across a number of tape marks

syntax: FBS␣/<file code>⌈,<number>⌉

where:
<file code> = file code of device
<number> = decimal or hexa number indicating the number of marks to be spaced back
Default = 1

File forward space

use: position the device after a tape mark

syntax: FFS␣/<file code>⌈,<number>|ALL⌉

where:
<file code> = file code of device
<number> = decimal or hexa number indicating the number of tape marks to be skipped
ALL = the device is positioned to two consecutive tape marks (:EOS:EOF)
Default = 1

Print label

use: print label on typewriter log and position tape at first record after label

syntax: PLB␣/<file code>

where:
<file code> = file code of device

Space backwards

use: space backward one or more records

syntax: RBS␣/<file code>⌈,<number>⌉

where:
<file code> = file code of device
<number> = decimal or hexa number indicating the number of records to be backspaced
Default = 1

Position file to first record

use: position the file to the first record

syntax: REF␣/<file code>

where:
<file code> = file code of file to be positioned

Rewind tape

use: rewind the specified tape

syntax: REW⌴/<file code>

where:

<file code> = file code of tape unit to be rewound

Record forward space

use: space forward until next physical record or the number of records specified

syntax: RFS⌴/<file code>⌈,<number>⌉

where:

<file code> = file code of device

<number> = decimal number indicating the number of records to space forward
Default = 1

Unlock device

use: switch the specified device to 'unlock' or 'switch off' state.

syntax: ULD⌴/<file code>

where:

<file code> = file code of device

Write EOF record

use: write one or a number of EOF records or tape marks

syntax: WEF⌴/<file code>⌈,<number>⌉

where:

<file code> = file code of device

<number> = decimal or hexa number indicating the number of EOF or tape marks to be written
Default = 1

Write EOS record

use: write one or a number of EOS records or tape marks

syntax: WES⌴/<file code>⌈,<number>⌉

where:

<file code> = file code of device

<number> = decimal or hexa number indicating the number of EOF or tape marks to be written
Default = 1

Write End-Of-Volume mark

use: write one EOV mark on the specified tape

syntax: WEV⌴/<file code>

where:

<file code> = file code of device

Write label

use: write a volume label on the specified tape

syntax: WLB⌴/<file code>, <number>,/<sec. code>,<owner>

where:

<file code> = file code of tape on which the volume label has to be written

<number> = volume serial number consisting of up to 6 characters

<sec. code> = security code consisting of one hexa character

<owner> = user identification which may consist of up to 39 characters

SCL COMMANDS DRTM

Assign a file code

syntax: AS⌴/<file code 1>,⌊/<file code 2>|DN⌈DA⌉|DKFX⌈<name>|,<no of granules>⌉⌋

where:

<file code 1> = file code which must be assigned

<file code 2> = file code to which <file code 1> must be assigned

DN⌈DA⌉ = device name and, if specified, device address to which file code 1 must be assigned

DKFX = must be used when <file code 1> must be assigned to a file on disc. (FX = /0 to /F)

<name> = file code 1 is assigned to a catalogued file

<no of granules> = the system allocates the specified number of granules to the file

Connect a program to a software level

syntax: CN⌴<name>, <level>

where:

<name> = name of program

<level> = level to which the program must be connected (49 to 61)

Connect a program to the clock or timer

syntax: CT⌴<name>,<NTIM>,<PR>⌊,<NC>|<HH>,<MM>,<SS>⌋

where:

<name> = name of program to be connected

<NTIM> = timer number. 0 if to be connected to RTC

<PR> = pulse rate (from 0 to 127). If 0 only one program activation

<NC> = no of timer cycles (0 to 9999 or /o to /7FFF)

<HH>,<MM>,<SS> = time in hours, minutes, seconds

Delete a file

syntax: DF⌴<disc number>,<file name>

where:

<disc number> = file code of disc on which the file is catalogued (/F0 to /FF)

<file name> = name of file to be deleted

Delete a file code

syntax: DL␣/<file code>

where:
<file code> = file code to be deleted

Disconnect a program from a level

syntax: DN␣<name>,<level>
where:
<name> = name of program to be disconnected
<level> = level to which the program was connected

Disconnect a program from a timer

syntax: DT␣<name>,<NTIM>

where:
<name> = the program of this name is to be disconnected
<NTIM> = timer number

End of commands

syntax: EN

Halt clock

syntax: HC

Keep file

syntax: KF␣/<file code>,<file name>

where:
<file code> = the file receives this file code
<file name> = name of file to be kept

Load a memory resident program

syntax: LD␣<name>,<disc number>⌈,<level>|, SL, <number>⌉

where:
<name> = name of program to be loaded
<disc number> = file code (/F0 to /FF) of disc from which the program is loaded
<level> = the program is an interrupt routine and must be connected to this level
SL, <number> = the program uses scheduled labels when <number> specifies the maximum number of labels to be scheduled at the same time

Declare a Read Only program

syntax: RO␣<name>,<disc number> ⌈, SL,<number>⌉

where:
<name> = name of the Read Only program
<disc number> = file of disc on which the program can be found (/F0 to /FF)
SL,<number> = the program use scheduled labels, <number> specifies the number of scheduled labels for which a save area must be reserved

Set clock

syntax: SC␣⌈<HH>⌈,<MM>⌈,<SS>⌉⌉⌉
This command indicates the time on which the clock will be started. Default = 0

Set date

syntax: SD␣<DD>,<MM>,<YY>

Start a program

syntax: ST␣<name>

where:
<name> = name of the program declared previously. The program must have been connected to a software level

Declare a swappable program

syntax: SW␣<name>,<disc number>⌊<time slice>|S⌋,⌊I|E⌋ ⌈,SL,<number>⌉

where:
<name> = name of program
<disc number> = file code of disc on which the program is stored (/F0 to /FF)
<time slice>|S = value of time slice for this program
It must be a multiple of 100 milliseconds
If S is specified the time slice as defined at SYSGEN
I = the program can be swapped immediately
E = the program is swapped after termination of all current I/O operations
SL, <number> = specifies the number of scheduled labels for which space must be reserved in the program's save area

Define time slice

syntax: TS␣<number>

where:
<number> = length of time slice in tenths of seconds (max = 256)

Save disc onto magnetic tape

syntax: SM␣/<disc file code>,/<mag. tape file code>

where:
<disc file code> = file code of disc to be copied on magnetic tape
<mag. tape file code> = file code of magnetic tape onto which the disc is copied

MAGNETIC TAPE AND CASSETTE TAPE COMMANDS (SCL)

Skip forward

syntax: FF␣/<file code>⌈,<number>|ALL⌉

where:
<file code> = file code of device
<number> = decimal number indicating the number of files to be skipped forward.
Default = 1

Skip backward

syntax: BF␣/<file code>⌈,<number>⌉

where:
<file code> = file code of device
<number> = decimal number indicating the number of files to be skipped backwards
Default = 1

Skip backward record

syntax: BR␣/<file code>⌈,<number>⌉

where:
<file code> = file code of device
<number> = decimal number indicating the number of records to be skipped backwards
Default = 1

Skip forward record

syntax: FR␣/<file code>⌈,<number>⌉

where:
<file code> = file code of device
<number> = decimal number indicating the number of records to be skipped forward
Default = 1

Rewind

syntax: RW␣/<file code>

where:
<file code> = file code of tape to be rewound

Unload

syntax: UN␣/<file code>

where:

<file code> = file code of tape to be unloaded

Write EOF

syntax: WF␣/<file code>

where:
<file code> = file code of file after which EOF must be written

Write EOS

syntax: WS␣/<file code>

where:
<file code> = file code of file after which EOS must be written

Write EOV

syntax: WV␣/<file code>

where:
<file code> = file code of file on which the EOV mark must be written

DISC PROCESSOR MESSAGES

Line Editor

!!CH␣$$<char string a>$$<char string b>$$
Replace<char string a> by <char string b> where ever in the program

!!LS␣$$<char string>$$
List all lines containing this character string

!!JN␣⌈<line no>,⌉<name>,<line no a>,<line no b>
Insert the lines <line no a> to <line no b> inclusive of the module named <name> after <line no> of the current input.
If <line no> is not specified the lines are inserted behind the current line of the main input file

!!RE␣<line no>,$$<char string a>$$<char string b>$$
Replace <char string a> by <char string b> in the line with the specified line number <line no>

!!DL␣<line no a>⌈,<line no b>⌉
Delete the line specified or the lines in the specified range

!!IL␣⌈<line no>⌉
Insert line(s) after the specified line number or after the current statement if no parameter is specified

!!AB
Abort the update

!!EN
This command terminates the updating session

TERMINATE THE UPDATING WITH A KPF COMMAND

Debugging Package

<memory reference>:: =	absolute address =	/<up to 4 hexa digits>. In IF command: M␣<hexa number>
	relative address =	@ <up to 4 hexa digits>
	symbolic address =	1. when DS option is specified: – $<symb table name>& <label> ±<dec no> 2. when DE option is specified – $<symb table name>& <label>±<dec no> – $<entry point> ± <decimal number>

<register>::= R<2-digit decimal number>
<constant>::= /<up to 4-digit hexa number>

AT␣<memory reference>
Define a breakpoint

RT
Return to interactive mode

IF␣⌈<memory ref>|<register>⌉>|=|<⌈<memory ref>|<register>|<constant>⌉
    Conditional execution of the attached breakpoint

GO␣⌈<memory reference>⌉
    Continue execution of user program

DB␣<memory reference>
    Delete a breakpoint

DM␣<memory ref a>,<memory ref b>
    Dump the memory specified

DR␣⌈<register a>⌈,<register b>⌉⌉

    Dump register. The registers may be A1 to A14

WM␣<memory ref>,<constant 1>⌈,<constant 2>......,<constant n>⌉
    Write memory

WR␣<register>,<constant 1>⌈,<constant 2>......,<constant n>⌉
    Write register

CI␣/<file code>
    Change the device from which the debug commands are read

CO␣/<file code>
    Change the output device

RE␣/<file code>,<memory reference>,/<no of char>
    Read a number of characters from the specified device

TR␣<2 ASCII char>
    Trace

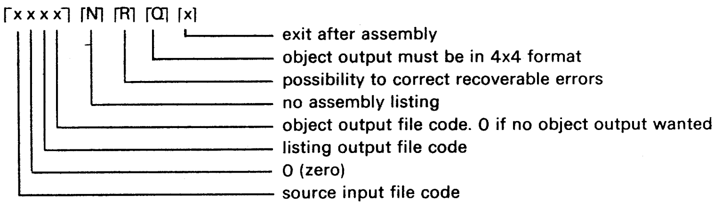
//
    Start the execution of the user program

RX
    Exit

BASIC PROCESSOR MESSAGE

Assembler

Option message:

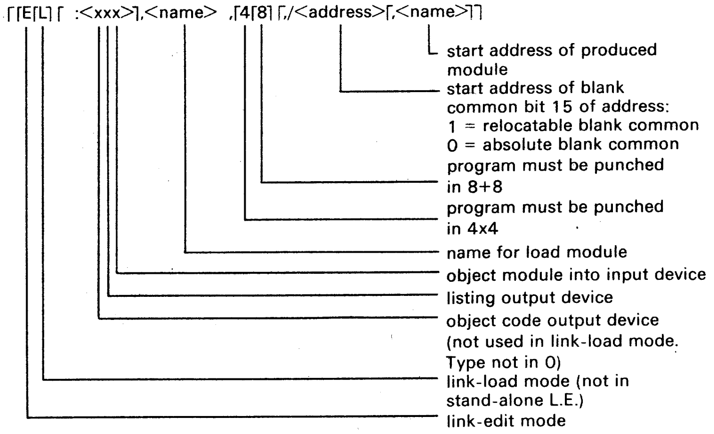

⌈x x x x⌉ ⌈N⌉ ⌈R⌉ ⌈Q⌉ ⌈x⌉
exit after assembly
object output must be in 4x4 format
possibility to correct recoverable errors
no assembly listing
object output file code. 0 if no object output wanted
listing output file code
0 (zero)
source input file code

Other messages:

LF CR	1. No option message required. Default: 1023 2. Resume processing after :EOS 3. Concludes typed-in statement
:EOF	Terminate processing after A:

Linkage Editor

Option message


⌈⌈E⌈L⌉⌈ :<xxx>⌉,<name> ,⌈4⌈8⌉⌈,/<address>⌈,<name>⌉⌉
start address of produced module
start address of blank common bit 15 of address:
1 = relocatable blank common
0 = absolute blank common
program must be punched in 8+8
program must be punched in 4x4
name for load module
object module into input device
listing output device
object code output device (not used in link-load mode. Type not in 0)
link-load mode (not in stand-alone L.E.)
link-edit mode

Operator Messages

LF CR	1. No option message required. Default = L:21 2. Concludes typed-in message

A␣<address>	Define absolute address
E␣<entry points name list>	Define entry point (link-edit only)
L	Use library to solve unsat. external ref.
P	Process input file up to EOF
R␣<address>	Define relative base address
S␣<symbol>	Select named module
T	Terminate processing
U	List undefined external references
X␣<external reference name list>	Define external reference names (link-edit only)

Update

Option message

```
<xxxxx>⌈,8⌈,4⌉
 |||||  |  |
 |||||  |  └─── object program to be updated is punched in 4x4
 |||||  └────── object program to be updated is punched in 8+8
 ||||└───────── file code onto which the updated program is punched or written
 |||└────────── file code of the peripheral onto which the input or output
 |||            program can be listed
 ||└─────────── file code from which data can be added to master file
 |└──────────── file code from which the control messages are read
 └───────────── master file code
```

Operator Messages

D:<name>	Delete object or source module of this name
D:<line no a>⌈␣<line no b>⌉	Delete line or lines (the ones specified included)
:EOF	Punch :EOF on Punch File
:EOS	Punch :EOS on Punch File
I: <name>	Insert object or source module of this name
I:	Correction statements may be input
I:<line no>	Insert line or lines on this place
L	List all modules of the input file
M:<name>	Start updating at t e module specified
S:<name>	All records deleted up to source module <name>
S	All records deleted up to :EOF mark
;	No updating until next :EOS
.	1. No opdating until next :EOF 2. After U: terminate processing
LF CR	Concludes typed-in statement or message

Debugging Package

<memory reference>: : = absolute address = /<up to 4 hexa digits>. In IF command
M␣<hexa number>
relative address = @ <up to 4 hexa digits>

<register>: : =R <2-digit decimal number>

<constant>: : =/<up to 4-digit hexa number>

AT␣<memory reference>
Define a breakpoint

RT
Return to interactive mode

IF␣⌈<memory ref>|<register>⌉>|=|<⌈<memory ref>|<register>|<constant>⌉
Conditional execution of the attached breakpoint

GO␣⌈<memory reference>⌉
Continue execution of user program

DB␣<memory reference>
Delete a breakpoint

DM␣<memory ref a>,<memory ref b>
Dump the memory specified

DR␣⌈<register a>⌈,<register b>⌉⌉
Dump register. The registers may be A1 to A14

WM␣<memory ref>,<constant 1>⌈,<constant 2>.....,<constant n>⌉
Write memory

WR␣<register>,<constant 1>⌈,<constant 2>.....,<constant n>⌉
Write register

CI␣/<file code>
Change the device from which the debug commands are read

CO␣/<file code>
Change the output device

RE␣/<file code>,<memory reference>,/<no of char>
Read a number of characters from the specified device

TR␣<2 ASCII char>
Trace

//
Start the execution of the user program

RX
Exit

Cassette Update Package

Define file codes

AS⊔I=<xx>, O=<xx>, A=<xx>, L=<xx>, P=<xx>, C=<xx>

where:
I = input file code
O = output file code
A = auxiliary file code
L = listing file code
P = punch file code
C = command input file code

Copy up to file

CF⊔[⌊<file name>|:EOL⌋]

where:
<file name> = file header name preceding the file up to which a copy must be made
:EOL = copy up to end of library

Skip to file

SF⊔[<file name>]

where:
<file name> = file header name of the file up to which must be skipped

Delete file

DF⊔[<file name>]

where:
<file name> = file header name of the file to be deleted

Insert file

IF⊔[⌊<file name>|:EOL⌋]

where:
<file name> = file header name of the last file to be inserted
:EOL = all files up to the end of library are inserted

Write header

WH⊔<name>

where:
the specified name is written onto the output tape

Search auxiliary file

SA⊔[<file name>]

where:
<file name> = file header name to be searched. The move may be forward as well as backward.

End of file

EF

Copy up to module

CM⊔[⌊<module name>|:EOF⌋]

where:
<module name> = ident of the module up to which a copy must be made
:EOF = all modules up to :EOF are copied

Skip to module

SM⊔[<module name>]

where:
<module name> = ident of the module up to which must be skipped

Delete module

DM⊔[<module name>]

where:
<module name> = ident of the module to be deleted

Insert module

IM⊔[⌊<module name>|:EOF⌋]

where:
<module name> = name of last module which must be inserted
:EOF = all modules up to :EOF are inserted

Delete line

!!DL⊔<number 1>[,<number 2>]
all the lines between number 1 (included) and number 2 (included) are deleted

Insert line

!!IL⊔[<number>]
if no number is specified the insertion will be after the current line.
If <number> is specified the insertion will be after the specified number

End of the line modification

!!EN

terminate line updating. The user may continue on line or module level

Exit

EN

control is returned to the monitor

Search header

SH⊔<header name>

where:
<header name> = file header name to be looked for

List headers

LH⊔[<header name>]

where:
<header name> = all idents under this name will be listed.
If not specified all fileheaders are listed

List file

LF⊔[<file name>][,<ident>]

where:
<file name> = the file of this name is listed.
If the parameter is not specified current file is listed
<ident> = all identifiers are listed.

List module

LM⊔[<module name>]

where:
<module name> = the module of this name will be listed.
If the parameter is not specified the current module is listed

Punch file

PF⊔[<file name>]

where:
<file name> = the file of this name is punched except for the file header. If the parameter is not specified the current file is punched

Punch module

PM⊔[<module name>]

where:
< module name> = the module of this name is punched.
If the parameter is not specified the current module is punched

DIRECTIVES

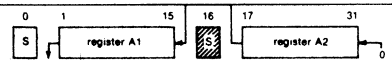
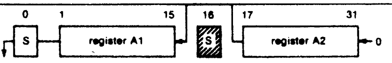
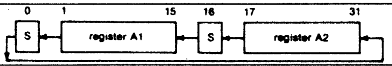
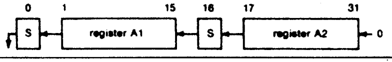
```
[<ident>]  ⊔DATA⊔<data expression>[,<data expression>, ....]
           (up to 16 words generated)
[<ident>]  ⊔EQU⊔<predefined expression>
           ⊔IDENT⊔<module name>
[<ident>]  ⊔END⊔[<predefined expression>] [,<symbol>]
[<ident>]  ⊔RES⊔<predefined absolute expression>
           ⊔AORG⊔<predefined absolute expression>
           ⊔RORG⊔[<predefined relocatable expression>]
           ⊔ENTRY⊔<entry point name>[,<entry point name>, ....,
           <entry point name>]
           ⊔EXTRN⊔<external name>    external name>, ....,
           <external name>]
           ⊔STAB⊔[<internal symbol
           <internal symbol>]
           ⊔NLIST⊔
           ⊔LIST⊔
           ⊔EJECT⊔
           ⊔[IFT|IFF]⊔<predefined absolute expression>=
           <predefined absolute expression>
           ⊔XIF⊔
[<ident>]  ⊔COMN⊔<common field definition list>
 <ident>   ⊔FORM⊔<field definition>[,<field definition>, ....,
           <field definition>] [/<field number list>]
 <ident>   ⊔XFORM⊔<FORM defined pseudo-mnemonic>,<field list>
           ⊔GEN⊔

           <data expression>::=<expression>|<character string>
           <field definition>::=<field length definition>
           [=<field value definition>]
```

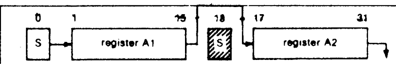
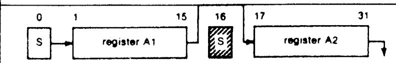
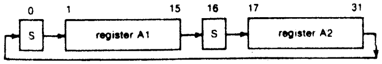
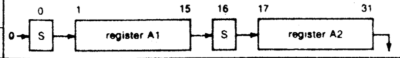
name (in alphabetical order)	mnemonic P852M	mnemonic P856M P857M	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in μ sec. for memory P852M 1.2	P856M / P857M 0.7	P856M / P857M 1.2	remarks
Absolute branch	ABI	ABI	1	0001	10	0	(M) → P		4.4	2.5	4.1	bits 5-7: condition
					10	0	(M + (R2)) → P		4.6	2.7	4.4	bit 8: n.s.
					11	0	((M)) → P		6.0	3.1	5.8	
					11	0	((M + (R2))) → P		6.2	3.4	5.9	
						0	no branch: (P) + 4 → P		1.8	1.0	1.6	
Absolute conditional	AB	AB	0	0001	-	n.s.	K → P		1.6	1.1	1.3	short format
branch (with constant)							no branch: (P) + 2 → P	3)	1.6	0.9	1.0	
Absolute conditional	ABL	ABL	1	0001	01	0	KL → P		2.8	1.8	2.6	long format
branch (with constant)							no branch: (P) + 2 → P		1.8	0.9	1.8	
Absolute conditional	ABR	ABR	1	0001	00	n.s.	(R2) → P		2.1	1.4	1.8	bits 5-7: condition
branch to register					01	0	((R2)) → P		3.3	2.0	2.3	bit 8: n.s.
							no branch: (P) + 2 → P		1.6	0.9	1.3	
Add constant	ADK	ADK	0	0010	-	-	(R3) + K → R3		1.8	0.9	1.3	short; 6)
	ADKL	ADKL	1	0010	01	0	(R1) + KL → R1		3.0	1.6	2.6	long 6)
Addition	AD	AD	1	0010	10	0	(R1) + (M) → R1		4.6	2.3	3.8	6)
					10	1	(R1) + (M) → M	2)	5.8	3.2	5.1	when l/s bit = 1,
					10	0	(R1) + (M + (R2)) → R1		4.8	2.3	3.8	R1 must be ≠ 0
					10	1	(R1) + (M + (R2)) → M + (R2)		6.0	3.4	5.1	

					11	0	(R1) + ((M)) → R1		6.2	2.9	5.1	
					11	1	(R1) + ((M)) → M		7.4	3.8	6.3	
					11	0	(R1) + ((M + (R2))) → R1		6.4	3.2	5.1	
					11	1	(R1) + ((M + (R2))) → (M + (R2))	2)	7.6	4.1	6.3	
Addition/register	ADR	ADR	1	0010	00	n.s.	(R1) + (R2) → R1		2.3	1.2	1.4	when l/s bit = 1,
					01	0	(R1) + ((R2)) → R1		3.5	1.8	3.0	R1 must be ≠ 0;
					01	1	(R1) + ((R2)) → (R2)		4.7	2.7	3.8	6)
Call function	CFI	CFI	1	1110			(P) → (R1), (R1) - 2 → R1					6) 7)
							(PSW) → (R1), (R1) - 2 → R1					
							then:					
					10	1	(M) → P		8.3 7.9	4.7 4.7	4.4 4.9	
					10	1	(M + (R2)) → P		8.5 8.1	4.9 4.9	5.1 5.1	
					11	1	((M)) → P		9.9 9.5	5.3 5.6	5.5 5.8	
					11	1	((M + (R2))) → P		10.1 9.7	5.6 5.3	5.8 5.5	
Call function/constant	CF	CF	1	1110	01	1	(P) → (R1), (R1) - 2 → R1	3)	6.7	4.0	4.2	
							(PSW) → (R1), (R1) - 2 → R1		6.3	4.0	4.2	7)
							KL → P		6.3	4.0	4.2	if <r1> ≠ A15
Call function/register	CFR	CFR	1	1110			(P) → (R1), (R1) - 2 → R1		6.0 5.6	3.7 3.7	3.9 3.9	7)
							(PSW) → (R1), (R1) - 2 → R1	3)	7.2 6.8	4.2 4.2	4.4 4.4	
							then:					6)
					00	1	(R2) → P					
					01	1	((R2)) → P					

name (in alphabetical order)	mnemonic		for- mat	OP- code	mode	L/S (0/1) bit	function	condition register	execution time in μ sec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
Clear memory	CM	CM	1	0100	10	1	0 → M		5.6	2.5	3.8	bits 5-8: 0000
					10	1	0 → M + (R2)		5.8	2.8	3.9	
					11	1	0 → (M)	3)	7.2	3.1	5.1	
					11	1	0 → (M + (R2))		7.4	3.4	5.1	
Clear memory/register	CMR	CMR	1	0100	01	1	0 → (R2)		4.5	2.2	2.8	bits 5-8: 0000
Compare characters	CC	CC	1	1101	10	1	(R1)r ÷ (M) l/r → CR		4.6	2.7	3.8	6)
					10	1	(R1)r ÷ (M + (R2)) l/r → CR		4.8	3.0	3.9	
					11	1	(R1)r ÷ ((M)) l/r → CR		6.2	3.3	5.1	
					11	1	(R1)r ÷ ((M + (R2))) l/r → CR		6.4	3.6	5.1	
Compare characters	CCR	CCR	1	1101	01	1	(R1)r ÷ ((R2)) l/r → CR		3.5	2.3	2.8	6)
register/register												
Compare character	CCK	CCK	1	1101	01	1	(R1)r ÷ KLl → CR	4)	3.0	2.3	2.9	6)
with constant												
Compare words	CW	CW	1	1101	10	0	(R1) ÷ (M) → CR		4.6	2.3	3.8	6)
					10	0	(R1) ÷ (M + (R2)) → CR		4.8	2.5	3.8	
					11	0	(R1) ÷ ((M)) → CR		6.2	2.9	5.9	
					11	0	(R1) ÷ ((M + (R2))) → CR		6.4	3.2	5.1	
Compare words	CWR	CWR	1	1101	00	n.s.	(R1) ÷ (R2) → CR		2.3	1.2	1.3	6)

register/register					01	0	(R1) ÷ ((R2)) → CR				3.5	1.8	2.6	
Compare word	CWK	CWK	1	1101	01	0	(R1) ÷ KL → CR			4)	3.0	1.6	2.6	6)
with constant														
Control Input/Output	CIO	CIO	0	1000	–	–	Start (bit 9=1) or stop (bit 9=0)			5)	4.8	3.4	4.4	bit 8 = 1
							any I/O operation							
								quotient	remainder					
Divide	DV	DV	1	1001	10	0	(A1, A2) / (M) →	A2	A1			8.5	10.0	
					10	0	(A1, A2) / (M + (R2)) →	A2	A1			8.6	10.0	
					11	0	(A1, A2) / ((M)) →	A2	A1			9.1	11.3	
					11	0	(A1, A2) / ((M + (R2))) →	A2	A1			9.4	11.3	
								quotient	remainder					
Divide by constant	DVK	DVK	1	1001	01	0	(A1, A2) / KL →	A2	A1			7.7	8.8	
								quotient	remainder					
Divide registers/registers	DVR	DVR	1	1001	00	0	(A1, A2) / (R2) →	A2	A1	2)		7.4	7.7	
					01	0	(A1, A2) / ((R2)) →	A2	A1			8.0	8.7	
Double Add	DA	DA	1	1010	10	0	(M, M + 1) + (A1, A2) → A1, A2					3.8	5.6	
					10	0	(M + (R2), M + (R2) + 2) + (A1, A2) → A1, A2					4.2	5.6	
					11	0	((M), (M) + 2) + (A1, A2) → A1, A2					4.5	6.9	
					11	0	((M + (R2)), ((M + (R2) + 2) + (A1, A2) → A1, A2					4.7	6.9	
Double add registers/	DAR	DAR	1	1010	00	0	(R2, R2 + 2) + (A1, A2) → A1, A2					2.9	3.1	
registers					01	0	((R2), (R2 + 2)) + (A1, A2) → A1, A2					3.5	4.4	

name (in alphabetical order)	mnemonic		for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in µsec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
Double add with constant	DAK	DAK	1	1010	01	0	KL + (A1, A2) → A1, A2			3.2	4.4	
Double subtract	DS	DS	1	1011	10	0	(A1, A2) – (M, M + 2) → A1, A2			3.8	5.6	
					10	0	(A1, A2) – (M + (R2), M + (R2) + 2) → A1, A2			4.2	5.6	
					11	0	(A1, A2) – ((M), (M) + 2) → A1, A2			4.5	6.9	
					11	0	(A1, A2) – ((M + (R2)), ((M + (R2)) + 2) → A1, A2	2)		4.7	6.9	
Double subtract	DSR	DSR	1	1011	00	0	(A1, A2) – (R2, R2 + 2) → A1, A2			2.9	3.1	
registers/registers					01	0	(A1, A2) – ((R2), (R2 + 2)) → A1, A2			3.5	4.4	
Double subtract	DSK	DSK	1	1011	01	0	(A1, A2) – KL1, KL2 → A1, A2			3.2	4.4	
with constant												
Double left and	DLN	DLN	1	0111	–	n.s.	 0 1 15 16 17 31 S register A1 S register A2 0	3)		nx 0.5 + 4.2	nx 0.5 + 4.2	bits 8-10: 100
normalize shift												bits 11-14: R2;
												bit 15: n.s.
Double left arithmetic shift	DLA	DLA	0	0111	–	–	 0 1 15 16 17 31 S register A1 S register A2 0	2)		nx 0.3 + 3.2	nx 0.3 + 3.1	bits 8-10: 000
Double left circular shift	DLC	DLC	0	0111	–	–	 0 1 15 16 17 31 S register A1 S register A2	1)		nx 0.3 + 2.3	nx 0.3 + 2.4	bits 8-10: 110
Double left logical shift	DLL	DLL	0	0111	–	–	 0 1 15 16 17 31 S register A1 S register A2 0	3)		nx 0.3 + 2.3	nx 0.3 + 2.4	bits 8-10: 010

Double right and	DRN	DRN	0	0111	–	–	 0 1 15 18 17 31 S register A1 S register A2	3)		nx 0.5 + 5.0	nx 0.5 + 4.9	bits 8-10: 101
normalize shift												bits 11-14: R2;
												bit 15: n.s.
Double right arithmetic	DRA	DRA	0	0111	–	–	 0 1 15 16 17 31 S register A1 S register A2	1)		nx 0.3 + 2.8	nx 0.3 + 3.1	bits 8-10: 001
shift												
Double right circular shift	DRC	DRC	0	0111	–	–	 0 1 15 16 17 31 S register A1 S register A2			nx 0.2 + 2.6	nx 0.3 + 2.4	bits 8-10: 111
Double right logical shift	DRL	DRL	0	0111	–	–	 0 0 1 15 16 17 31 S register A1 S register A2			nx 0.3 + 2.6	nx 0.3 + 2.4	bits 8-10: 011
Enable interrupt	ENB	ENB	0	0101	–	–	machine status = 'permit interrupt'	3)	2.1	3.5	3.5	bits 8-15: 01000000
Execute	EX	EX	1	1110	10	1	(M) is executed		+11.5	+6.0	+7.0	bits 5-8: 0000
					10	1	(M + (R2)) is executed		+11.7	+6.2	+6.4	EXK also bits 11-14.0000
					11	1	((M)) is executed		+14.1	+6.7	+7.6	The executed instr. may
					11	1	((M + (R2))) is executed		+14.3	+7.0	+8.2	**not** be another EX, EXK,
Execute constant	EXK	EXK	1	1110	01	1	KL is executed		+ 7.6	+5.9	+6.3	EXR, RTN, CF, or double
Execute register	EXR	EXR	1	1110	00	1	(R2) is executed		+ 4.8	+5.0	+5.8	format Execution time =
					01	1	((R2)) is executed		+ 8.1	+5.5	+6.5	instr. in eff. mem. addr. +
Exchange characters	ECR	ECR	1	1100	00	n.s.	(R2) ⌊ → R1 r; (R2)r → R1⌊	3)				spec. times
register/register									5.3	1.2	1.3	6)
Exclusive OR	XR	XR		0110	10	0	(R1) ⊻ (M) → R1	1)	4.6	2.3	3.8	6)
					10	1	(R1) ⊻ (M) → M		5.8	3.2	5.1	
					10	0	(R1) ⊻ (M + (R2)) → R1		4.8	2.5	3.8	

name (in alphabetical order)	mnemonic		for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in µ sec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
					10	1	(R1) ⊻ (M + (R2)) → M + (R2)	1)	6.0	3.4	5.1	
					11	0	(R1) ⊻ ((M)) → R1		6.2	2.9	5.1	
					11	1	(R1) ⊻ ((M)) → (M)		7.4	3.8	6.3	
					11	0	(R1) ⊻ ((M + (R2))) → R1		6.4	3.2	5.1	
					11	1	(R1) ⊻ ((M + (R2))) → (M + (R2))		7.6	4.1	6.3	
Exclusive OR	XRR	XRR	1	0110	00	0	(R1) ⊻ (R2) → R1		2.3	1.2	1.3	6)
register/register					01	0	(R1) ⊻ ((R2)) → R1		3.5	1.8	2.6	
					01	1	(R1) ⊻ ((R2)) → R2		4.7	2.7	3.8	
Exclusive OR	XRK	XRK	0	0110	–	–	(R3)$_{8-15}$ ⊻ K → R3$_{8-15}$		1.8	0.9	1.3	short; 6)
with constant												
Exclusive OR	XRKL	XRKL	1	0110	01	0	(R1) ⊻ KL → R1		3.0	1.6	2.6	long; 6)
with constant												
Halt	HLT	HLT	0	0100	–	–	machine → 'halt' mode	3)	2.1	1.1	1.7	bits 8-15: 01111111;
Increment Memory	IM	IM	1	0010	10	1	(M) + 1 → M	2)	5.8	3.2	5.1	bits 5-8: 0000
					10	1	(M + (R2)) + 1 → M + (R2)		6.0	3.4	5.1	
					11	1	((M +)) + 1 → (M)		7.4	3.8	6.3	
					11	1	((M) + (R2))) + 1 → (M + (R2))		7.6	4.1	6.3	

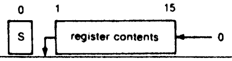
Increment memory/	IMR	IMR	1	0010	01	1	((R2)) + 1 → (R2)	2)	4.7	2.7	3.8	bits 5-8: 0000
register												
Inhibit interrupt	INH	INH	0	0100	–	–	machine status = 'prohibit all interrupts'	3)	2.1	1.1	1.7	bits 8-15: 10111111
Input to register	INR	INR	0	1001	–	–	word/character from device → R3	5)	4.7	5.2	5.3	bit 8 = 0;
Link to monitor	LKM	LKM	0	0101	–	–	user mode → system mode (P855M)	3)	2.1	3.5	3.5	bits 8-15; 00000100
Load character	LC	LC	1	1100	10	0	(M) l/r → R1r		4.4	2.6	3.8	6)
					10	0	(M + (R2)) l/r → R1r		4.6	3.0	3.9	R1 must be ≠ 0
					11	0	((M)) l/r → R1r		6.0	3.2	5.1	
					11	0	((M + (R2))) l/r → R1r		6.2	3.5	5.1	
Load character/constant	LCK	LCK	1	1100	01	0	KL l → R1r		2.8	2.3	2.9	6)
Load character/register	LCR	LCR	1	1100	01	0	((R2)) l/r → R1r		3.3	2.3	2.8	6)
Load constant	LDK	LDK	0	0000	–	–	K → $R3_{8\text{-}15}$, 0 → $R3_{0\text{-}7}$		1.6	0.9	1.3	short; 6)
	LDKL	LDKL	1	0000	01	0	KL → R1		3.0	1.6	2.6	long; 6)
Load register	LD	LD	1	0000	10	0	(M) → R1	1)	4.6	2.2	3.7	6)
					10	0	(M + (R2)) → R1		4.8	2.2	4.0	
					11	0	((M)) → R1		6.2	2.9	4.6	
					11	0	((M + (R2))) → R1		6.4	3.2	5.0	

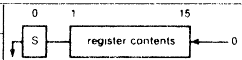
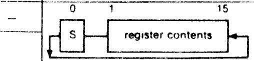
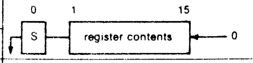
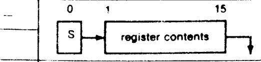
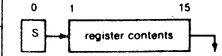
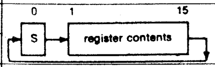
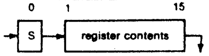
name	mnemonic		for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in µsec. for memory			remarks
(in alphabetical order)	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
Load register/register	LDR	LDR	1	0000	00	n.s.	(R2) → R1	1)	3.3	1.2	1.4	6)
					01	0	((R2)) → R1		3.5	1.8	2.2	
					01	0	(A15) + 2 → A15, ((A15)) → R1		4.6	2.2	2.5	
Logical AND	AN	AN	1	0100	10	0	(R1) ∧ (M) → R1		4.6	2.3	3.8	6)
					10	1	(R1) ∧ (M) → M		5.8	3.2	5.1	
					10	0	(R1) ∧ (M + (R2)) → R1		4.8	2.5	3.8	
					10	1	(R1) ∧ (M + R2)) → M + (R2)		6.0	3.4	5.1	
					11	0	(R1) ∧ ((M)) → R1		6.2	2.9	5.1	
					11	1	(R1) ∧ ((M)) → (M)		7.4	3.8	5.1	
					11	0	(R1) ∧ ((M + (R2))) → R1		6.4	3.2	5.1	
					11	1	(R1) ∧ ((M + (R2))) → (M + (R2))		7.6	4.1	5.1	
Logical AND	ANR	ANR	1	0100	00	0	(R1) ∧ (R2) → R1		2.3	1.2	1.3	6)
register/register					01	0	(R1) ∧ ((R2)) → R1		3.5	1.8	2.6	
					01	1	(R1) ∧ ((R2)) → (R2)		4.7	2.7	3.8	
Logical AND	ANK	ANK	0	0100	–	–	$(R3)_{8\text{-}15}$ ∧ K → $R3_{8\text{-}15}$		1.8	0.9	1.3	short; 6)
with constant	ANKL	ANKL	1	0100	01	0	(R1) ∧ KL → R1		3.0	1.6	2.6	long; 6)
Logical OR	OR	OR	1	0101	10	0	(R1) ∨ (M) → R1		4.6	2.3	3.8	6)

					10	1	(R1) ∨ (M) → M		5.8	3.2	5.1	6)
					10	0	(R1) ∨ (M + (R2)) → R1		4.8	2.5	3.8	
					10	1	(R1) ∨ (M + (R2)) → M + (R2)		6.0	3.4	5.1	
					11	0	(R1) ∨ ((M)) → R1		6.2	2.9	5.1	
					11	1	(R1) ∨ ((M)) → (M)		7.4	3.8	6.3	
					11	0	(R1) ∨ ((M + (R2))) → R1		6.4	3.2	5.1	
					11	1	(R1) ∨ ((M + (R2))) → (M + (R2))		7.6	4.1	6.3	
Logical OR	ORR	ORR	1	0101	00	0	(R1) ∨ (R2) → R1		2.3	1.2	1.3	6)
register/register					01	0	(R1) ∨ ((R2)) → R1		3.5	1.8	2.6	
					01	1	(R1) ∨ ((R2)) → (R2)		4.7	2.7	3.8	
Logical OR	ORK	ORK	0	0101	–	–	$(R3)_{8-15} \nleftrightarrow K \rightarrow R3_{8-15}$		1.8	0.9	1.3	short; 6)
with constant								1)				
Logical OR	ORKL	ORKL	1	0101	01	0	(R1) ∨ KL → R1		3.0	1.6	2.6	long; 6)
with constant												
Multiple load	ML	ML	1	0111	10	0	(M) ... (M + n) → A1 ... An			nx 0.8 + 2.7	nx 1.3 + 2.8	
					10	0	(M + (R2)) ... (M + (R2) + n) → A1 ... An			nx 0.8 + 3.2	nx 1.3 + 3.3	
					11	0	((M)) ... ((M) + n) → A1 ... An			nx 0.8 + 3.4	nx 0.8 + 3.5	
					11	0	((M + (R2))) ... ((M + (R2)) + n) → A1 ... An			nx 0.8 + 3.6	nx 0.8 + 4.1	6)
Multiple load/constant	MLK	MLK	1	0111	01	0	KL1, KL2, ... KLn → A1, A2, ... An			nx 0.8 + 2.8	nx 1.3 + 2.9	6)
Multiple load/register	MLR	MLR	1	0111	01	0	((R2)) → A1; ((R2) + 2) → A2; ..... ;	8)				6)
							((R2) + 2n-2) → An			nx 0.8 + 2.4	nx 1.3 + 2.4	bits 5-8; n.

name (in alphabetical order)	mnemonic		for- mat	OP- code	mode	L/S (0/1) bit	function		execution time in µ sec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
						–	(A15) + 2n → A15; ((A15)) → A1;					
							((A15) - 2) → A2; .... ;	8)				bits 5-8: n
							((A15) - 2n + 2) → An			nx 0.8 + 2.0	nx 1.3 + 2.0	
Multiple store	MS	MS	1	0111	10	1	A1 ... An → M ... M + n			nx 0.8 + 2.7	nx 1.3 + 2.8	6)
					10	1	A1 ... An → M + (R2) ... M + (R2) + n	3)		nx 0.8 + 3.2	nx 1.3 + 3.3	bits 5-8: n
					11	1	A1 ... An → (M) ... (M) + n			nx 0.8 + 3.4	nx 1.3 + 3.5	
					11	1	A1 ... An → (M + (R2)) ... (M + (R2)) + n			nx 0.8 + 3.6	nx 1.3 + 4.1	
Multiple store/register	MSR	MSR	1	0111	01	1	(A1) → (R2) ; (A2) → (R2) + 2 ;			nx 0.8 + 2.4	nx 1.3 + 2.5	6)
							....; (An) → (R2) + 2n - 2					bits 5-8: n
							(A1) → (A15) ; (A2) → (A15) - 2n + 2	3)		nx 0.8 + 2.8	nx 0.8 + 3.1	7)
							.... ; (An, → (A15) - 2n + 2 ;					bits 5-8: n;
							(A15) - n → (A15)					
Multiply	MU	MU	1	1000	10	0	(A2) x (M) → A1, A2			7.8	9.7	
					10	0	(A2) x (M + (R2)) → A1, A2	2)		8.0	9.7	
					11	0	(A2) x ((M)) → A1, A2			8.4	10.9	
					11	0	(A2) x ((M + (R2))) → A1, A2			8.7	10.9	

Multiply registers/registers	–	MUR	1	1000	00	0	(A2) x (R2) → A1, A2			6.8	7.7	
					01	0	(A2) x ((R2)) → A1, A2	2)		7.3	8.4	
Multiply with constant	–	MUK	1	1000	01	0	(A2) x KL → A1, A2			7.1	8.4	
Negate register	NGR	–	1	0011	00	1	O – (R2) → R1	2)	2.1	1.9	2.0	R1 ≠ 0
One's complement	C1	C1	1	1111	10	0	$\overline{(M)}$ → R1		4.6	2.3	3.8	6)
					10	1	$\overline{(M)}$ → M		5.8	3.2	5.1	when l/s bit = 0
					10	0	$\overline{(M + (R2))}$ → R1		4.8	2.5	3.8	R1 must be = 0
					10	1	$\overline{(M + (R2))}$ → M + (R2)		6.0	3.4	5.1	
					11	0	$\overline{((M))}$ → R1		6.2	2.9	5.1	
					11	1	$\overline{((M))}$ → (M)	1)	7.4	3.8	6.4	
					11	0	$\overline{((M + (R2)))}$ → R1		6.4	3.2	5.1	
					11	1	$\overline{((M + (R2)))}$ → (M + (R2))		7.6	4.1	6.4	
One's Complement	C1R	C1R	1	1111	00	n.s.	$\overline{(R2)}$ → R1		2.3	1.2	1.3	when l/s bit = 0,
register/register					01	0	$\overline{((R2))}$ → R1		3.5	1.8	2.5	R1 must be = 0;
					01	1	$\overline{((R2))}$ → (R2)		4.7	2.7	3.8	6)
Output from register	OTR	OTR	0	1000	–	–	word/character from R3 → device	5)	4.8	4.4	4.4	bit 8 = 0;

name (in alphabetical order)	mnemonic		for-mat	OP-code	mode	L/S (0/1) bit	function		execution time in µsec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
Relative backwards	RB	RB	0	0111	–	0	(P) + 2 + displ. → P (branch effective)		1.8	1.1	1.3	bits 5-7: condition
conditional branch							(P) + 2 → P (no branch)		1.6	0.9	1.0	bits 8-14: displ.;
												bit 15: n.s.
Relative forward	RF	RF	0	1010	–	0	(P) + 2 + displ. → P (branch effective)		1.8	1.1	1.3	bits 5-7: condition
						–	(P) + 2 → P (no branch)		1.6	0.9	1.0	bits 8-14: displ.;
								3)				bit 15: n.s.
Read external register	RER	RER	0	1111			(external register) → R3		4.2	4.6	5.1	bits 8-15: ext. reg. address
conditional branch												bits 8-15: ext. reg.
Reset internal	RIT	RIT	0	0100	–	1	to clear internal interrupt bits		2.1	1.1	1.7	bits 8, 9, 15: 1,
interrupt												bits 10-14: D.A.;
Return from Function	RTN	RTN	1	1110	01	0	(R2) + 4 → R2 ; ((R2)) → P ;	*	5.5	–	–	*reloaded from stack;
							((R2) –2) → CR					
Send status	SST	SST	0	1001	–	–	status character/word from device → R3	5)	5.3	5.2	5.3	bits 8-9: 11;
Set mode	–	SMD	0	0101	–	–	system mode → user mode	3)	–	3.5	1.7	bits 8-15: 00000001
Single left and	SLN	SLN	0	0111	–	–		2)	nx 0.4 + 3.2	nx 0.5 + 3.9	nx 0.5 + 4.2	bits 8-10: 100;
normalize shift												bits 11-14: R2; bit 15:

Single left arithmetic shift	SLA	SLA	0	0111	–	–		2)	nx 0.4 + 3.2	nx 0.5 + 2.1	nx 0.5 + 2.0	bits 8-10: 000
Single left circular shift	SLC	SLC	0	0111	–	–		1)	nx 0.4 + 3.2	nx 0.5 + 1.8	nx 0.5 + 1.9	bits 8-10: 110
Single left logical shift	SLL	SLL	0	0111	–	–			nx 0.4 + 3.2	nx 0.5 + 1.8	nx 0.5 + 1.9	bits 8-10: 010
Single right and normalize shift	SRN	SRN	0	0111	–	–		3)	nx 0.6 + 3.6	nx 0.5 + 3.9	nx 0.5 + 4.1	bits 8-10: 101 bits 11-14: R2: bit 15:N.S.
Single right arithmetic shift	SRA	SRA	0	0111	–	–			nx 0.4 + 3.2	nx 0.3 + 1.8	nx 0.3 + 1.9	bits 8-10: 001
Single right circular shift	SRC	SRC	0	0111	–	–		1)	nx 0.4 + 3.2	nx 0.3 + 1.7	nx 0.3 + 1.8	bits 8-10: 111
Single right logical shift	SRL	SRL	0	0111	–	–			nx 0.4 + 3.2	nx 0.3 + 1.7	nx 0.3 + 1.8	bits 8-10: 011
Store character	SC	SC	1	1100	10	1	(R1)r → (M) r/l		4.5	2.3	3.8	6)
					10	1	(R1)r → (M + (R2)) r/l		4.7	2.5	3.9	R1 must be ≠ 0
					11	1	(R1)r → ((M)) r/l	3)	6.1	2.9	5.1	
					11	1	(R1)r→ ((M + (R2))) r/l		6.3	3.2	5.1	
Store character/register	SCR	SCR	1	1100	01	1	(R1)r → (R2) r/l		3.4	2.1	2.8	6)

name (in alphabetical order)	mnemonic		for-mat	OP-code	mode	L/S (0/1) bit	function		execution time in µ sec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
Store register	ST	ST	1	0000	10	1	(R1) → M		4.5	2.5	3.8	6)
					10	1	(R1) → M + (R2)		4.7	2.8	4.1	
					11	1	(R1) → (M)		6.1	2.0	4.4	
					11	1	(R1) → ((M + (R2)))	3)	6.3	3.4	5.0	
Store register/register	STR	STR	1	0000	01	1	(R1) → (R2)		3.4	2.2	2.8	6)
					01	1	(R1) → (A15) ; (A15) - 2 → A15		4.5	2.9	3.1	7)
Subtract constant	SUK	SUK	0	0011	–	–	(R3) - K → R3					short; 6)
Substract constant	SUKL	SUKL	1	0011	01	0	(R1) - KL → R1		3.0	1.6	2.6	long; 6)
Subtract register/register	SUR	SUR	1	0011	00	n.s.	(R1) - (R2) → R1		2.3	1.2	1.4	when l/s bit = 1, R1 must
					01	0	(R1) - ((R2)) → R1	2)	3.5	1.8	2.6	be ≠ 0; 6)
					01	1	(R1) - ((R2)) → (R2)		4.7	2.7	3.8	
Subtract word	SU	SU	1	0011	10	0	(R1) - (M) → R1		4.6	2.3	3.8	6)
					10	1	(R1) - (M) → M		5.8	3.2	5.1	when l/s bit = 1,
					10	0	(R1) - (M + (R2)) → R1		4.8	2.5	3.8	R3 must be ≠ 0
					10	1	(R1) - (M + (R2)) → M + (R2)		6.0	3.4	5.1	

					11	0	(R1)–((M)) → R1	2)	6.2	2.9	5.1	
					11	1	(R1) – ((M)) → (M)		7.4	3.8	6.3	
					11	0	(R1) – ((M + (R2))) → R1		6.4	3.2	5.1	
					11	1	(R1 – ((M + (R2))) → (M + (R2))		7.6	4.1	6.3	
Test mask	TM	TM	1	0100	00	1	[(R1) ∧ (R2)] ÷ 0 → CR	1)	3.3	1.2	1.3	6)
Test not mask	TNM	TNM	1	0110	00	1	[(R1) ⊽ (R2)] ÷ 0 → CR		3.3	1.2	1.3	6)
Test status	TST	TST	0	1001	–	–	test DCU 'ready state'	5)	4.7	5.2	5.3	bits 8-9: 10;
Two's complement	C2	C2	1	0011	10	1	0 – (M) → M	2)	5.8	3.7	5.3	bits 5-8: 0000
					10	1	0 – (M + (R2)) → M + (R2)		6.0	3.8	5.3	
					11	1	0 – ((M)) → (M)		7.4	4.3	6.5	
					11	1	0 – ((M + (R2))) → (M + (R2))		7.6	4.6	6.5	
Two's complement/register	C2R	C2R	1	0011	01	1	0 – ((R2)) → (R2)	3)	4.7	3.2	4.0	bits 5-8: 0000
Write external register	WER	WER	0	1110			(R3) → external register		4.3	4.2	4.7	bits 8-15: ext. reg.
Extended load	–	EL	1	1010	10	0	(<m>) extended → <r1>	9)	–	2.5	3.0	MMU option system mode only
					10	0	(<m> + (r2>)) extended → <r1>		–	2.8	3.3	
					11	0	((<m>)) extended → <r1>		–	3.2	3.7	
					11	0	((<m> + (<r2))) extended → <r1>		–	3.4	4.1	
Extended load/register	–	ELR	1	1010	01	0	((<r2>)) extended → <r1>		–	2.2	2.5	
Extended store	–	ES	1	1010	10	1	(<r1>) → <m> extended	3)	–	2.5	3.0	
					10	1	(<r1>) → <m> + (<r2>) extended		–	2.9	3.4	
					11	1	(<r1>) → (<m>) extended		–	3.2	3.7	

name (in alphabetical order)	mnemonic		for- mat	OP- code	mode	L/S (0/1) bit	function	condition register	execution time in µ sec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
					11	1	(<r1>) ∸ (<m> + (<r2>)) extended	3)	–	3.4	4.1	
Extended store/register	–	ESR	1	1010	01	1	(<r1>) → (<r2>) extended		–	2.2	2.5	
Move table backward	–	MVB	0	1111	–	0	((A1)) → (A2)	3)	–	nx 1.8 + 4.1	nx 2.0 + 4.3	P857M only 6)
							(<r2>) – 2 → <r2>; (A1) + 2 → A1; (A2) + 2 →					
							A2; ((A1)) → (A2)					
							0 → <r2>; (A1) + 2 → A1; (A2) + 2 → A2					
Move table forward	–	MVF	0	1110	–	0	(<r2>) – 2 → <r2> ((A1 + <r2>)) → (A2 + <r2>)		–	nx 1.8 + 4.5	nx 2.0 + 4.7	
							0 → <r2> ((A1)) → (A2)					
Move table from user area	–	MVUS	0	1111	–	0	((A1)) → (A2); (<r2>) – 2 → <r2>	3)	–	nx 1.8 + 4.1	nx 2.0 + 4.3	P857M only 6)
to system area							((A1)) + 2) → (A2) + 2; (<r2>) – 2 → <r2>					
							((A1)) + 2n – 2 → A2 + 2n – 2; 0 → <r2>					
							n = number of transfers.					
Move table from system	–	MVSU	0	1110	–	0	(A1) and (A2) unchanged, (<r2>) updated		–	nx 1.8 + 4.1	nx 2.0 + 4.7	
area to user area							(<r2>) – 2 → ((<r2>)); ((A1 +<r2>)) →					
							(A2 + <r2>)					
							0 → ((<r2>)); ((A1)) → (A2)					
Segment table load	–	TL	1	0111	10	0	(<m>)...(<m> + 15.2) → TR0..TR15	3)	–	12.0	15.4	
					10	0	(<m>+(<r2>))...(<m>+(<r2>)+15.2)→TR0..TR15			12.4	15.8	

					11	0	(<m>))...((<m>) + 15.2) → TR0..TR15	3)		12.7	16.2	MMU option system mode only
					11	0	(<m> + (<r2>))...((<m> +(<r2>)) + 15.2)→			13.0	16.8	
							TR0..TR15					
Segment table	–	TLR	1	0111	01	1	((<r2>)) → TR0		–	11.8	15.1	
load/register							((<r2>) + 2) → TR1					
							((<r2>) + 15.2) → TR15					
Segment table	–	TS	1	0111	10	1	(TR0) → <m>; (TR1) → <m> + 2;...(TR15) →		–	12.1	15.4	
Store							<m> + 15.2					
					10	1	(TR0) → <m>+ (<r2>); (TR1) → <m> + (<r2>)		–	12.4	15.8	
							+ 2;...					
							(TR15) → <m> + (<r2>) + 15.2					
					11	1	(TR0) → (<m>); (TR1) → (<m> + 2);..(TR15)		–	12.7	16.2	
							(<m> + 15.2)					
					11	1	(TR0) → (<m> + (<r2>)); (TR1) → (<m> +		–	13.0	16.8	
							(<r2>) + 2);...					
							(TR15) → (<m> + (<r2>) + 15.2)					
Segment table	–	TSR	1	0111	01	1	(TR0) → (<r2>)		–	11.8	15.1	
store/register							(TR1) → (<r2>) + 2					
							(TR15) → (<r2>) + 15.2					
Return A1..A14	–	RTN	1	1110	01	0	(<r2>) + 4 → <r2>	•	–	2.7	3.2	reloaded from stack bits
							((<r2>)) → P					6, 7 of PSW → CR
							((<r2>)) – 2→ CR					

name (in alphabetical order)	mnemonic		for- mat	OP- code	mode	L/S (0/1) bit	function	condition register	execution time in µ sec. for memory			remarks
	P852M	P856M P857M							P852M	P856M / P857M		
									1.2	0.7	1.2	
Return A15	–	RTN	1	1110	01	0	(A15) + 4 → A15		–	4.1	4.6	PSW $_{6,7}$ → CR
							((A15)) → P					
							((A15) – 2) $_{6-7}$ → CR					
							((A15) – 2) $_{0-5}$ → PLR					
							((A15) – 2) $_{9}$ → ENB					
							((A15) – 2) $_{15}$ → SU					

NOTES FOR INSTRUCTION SET

1) CR = 0 if result = 0
1 if result > 0
2 if result < 0

2) CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 if overflow

3) CR unchanged

4) CR = 0 if a = b
1 if a > b
2 if a < b

5) CR = 0 if command accepted
1 if command not accepted
3 if device address unknown

6) For P856M/ R1 = 1111 system mode
P857M or
n = 1111 system mode

7) Stackpointer P850M: overflow
if contents < 128_{10}

8) CR = 0 if(A1) = 0
1 if(A2) > 0
2 if(A1) < 0

9) CR = 0 if (A1) = 0
1 if (A1) > 0
2 if (A1) < 0

SYSTEM MESSAGES

Message	Program	Meaning
A:	ASM	Request for option/control message.
A:	IPLRT	Request for activate message. Answer with A or LF CR
ABORT <code><address>	BOM	Program aborted
	DOM	<code> 1 power failure 2 non-available-instruction 3 memory protect error 4 buffer area destroyed or block> 16k 5 label could not be scheduled 7 buffer overflow 8 disc overflow 9 disc queue overflow A memory overflow during loading
ABORT␣<code>␣<address>	COS	Program aborted at the specified address. code: 01 simulation routine save area overflow 02 illegal instruction 04 buffer area destroyed or block bigger than 32k 05 label could not be scheduled 06 operator abort
ABS.ADR.	LKE	Disk Linkage Editor does not accept absolute addresses
ABS.STR.	LKE	Absolute start address (ignored)
ASS.ERR. <number>	ASM	Number of assembly errors
ASSIGN ERROR	CCI	KPF erroneous assignment
AUX. INPUT CANNOT BE ASSIGNED, [TRY AGAIN]	LE	Auxiliary file used in JN command cannot be assigned

Message	Program	Meaning
<dev addr>B::<volume name>	COS	Cassette with basic labelling has been loaded
BATCH PROCESSING?	CCI	Type in Y ⌈ES⌉ or N ⌈O⌉
BLK.COM	LKE	Erroneous optional blank common address
BLK DAT <name>	LKE	Unknown common block name used in a Block Data subprogram
BLK DATA ER	LKE	Data error encountered
BP CANNOT BE DELETED	DBG	A breakpoint must be terminated by GO or RT before it can be deleted
BP DOUBLE DEFINED	DBG	The breakpoint specified is present in the BP table
BP TABLE OVERFLOW	DBG	8 breakpoints may be present in the BP table
BYE MORE CORE	DRTM	Insufficient memory size
C:	UPD	Request for control message. See page
C?	UPD	Parameter error. Request for correct control message
*C	ASM	Illegal constant
<dev addr>C::SYST	COS	The system cassette has been loaded
CATALOG OVERFLOW	CCI	Too many userids catalogued
<dev addr>C::<volume name>	COS	Cassette with compact labelling has been loaded
CE:	Casupd	Erroneous input for cassette update. Retype command.
CMND NOT ALLOWED IN EXE MODE ⌈TRY AGAIN⌉	LE	The command input was a definition mode command
COMMAND NOT ALLOWED	CCI	This is not a system userid
COMMAND NOT ALLOWED IN A CAT. PROC	DOM	The command SCR without parameter is not allowed
COMMAND UNKNOWN	CCI	Erroneous CCI command
C ER	LKE	Labeled common error or error in base address of blank common
CORE OVERFLOW	LKE	Insufficient core available for user program
CORE RESIDENT AREA LENGTH:	DRTM	Type in the length (4 hexa char) of this area
D:	DBG	Request for control message.
DATE:	DOM/ DRTM	Disc system asking for date. DD MM YY or YY MM DD

Message	Program	Meaning
D.D.	LKE	Double definition error
DBL.DEF <name>	LKE	Name is defined more than once as an entry point or in the name of a common block
D:CI TOO BIG	DRTM	Not enough consecutive granules for D:CI file
D:CI TOO SMALL	DTRM	Not enough room in the file for system read only program
DEBUG OPTION REDUNDANT	CCI	Debug option redundant
DEVICE ADDRESS ERROR	CCI	Erroneous device address specified
DEVICE NAME ERROR	CCI	Erroneous device name specified
DEVICE NAME MISSING	CCI	2nd parameter missing in command
DEVICE UNKNOWN	CCI	Unknown device address
DIRECTORY OVERFLOW ON XXXXXFT	CCI	Directory overflow. File FT is catalogued
DISK ADDRESS MISSING	CCI	Disc address not specified
DISK <address> UNKNOWN	DRTM	Disc unknown by CPU
DISK ASSIGN ERROR	CCI	System cannot assign a temporary work file
DISK FILE CODE ABSENT	CCI	Disc file code not specified in command
DISK FILE CODE ERROR	CCI	2nd parameter not numeric / 1st parameter not a file code
DISK FILE CODE UNKNOWN	CCI	File code not declared at sysgen
DISK FILE CODE MISSING	CCI	No file code specified
DISK NOT OPERATIONAL	CCI	Disc unit not ready
DISK I/O ERROR	CCI	I/O error on disc
DISK OVERFLOW	CCI	No free granule available to allocate to temporary disc file
DISK UNIT <dev addr> UNKNOWN	DRTM	Non-wired unit
DISK UNKNOWN	CCI/DRTM	Disc not specified at sysgen
DKER␣<address><sect number> <status>	DOM/ DTRM	Disc not ready to be used/physical error on disc (sector destroyed).
DSK INPUT ERR, UPD ABORTED	CCI	Erroneous output from Disc Update
DSK INIT ERR	DOM/ DRTM	Disc not ready to be used
DSK OUTPUT ERR, UPD ABORTED	CCI	Erroneous output from Disc Update
DYN AREA LENGTH	DRTM	Length of dynamic area requested (4 hexa char.)

Message	Program	Meaning
* * DRTM * * yy * *	DRTM	yy=release number
* E	ASM	Address is not even
E = <absolute address>	LKE	Address is the highest absolute address in the generated module
* * * * * * E	ASM	END directive missing
<dev addr>E::<volume name>	COS	Cassette with extended labelling has been loaded
EC	BOM	Erroneous cluster or input error
EC TYPE	LKE	Erroneous cluster encountered
<dev addr> END	CFM COS	End of track or volume on address
END MIS	LKE	END cluster missing
EOF	BOM/ASM	End-of-file mark encountered
EOF IN AUXI INPUT	LE	An EOF has been read on the auxiliary input file but the operation continues
EOF, UPD TERMINATED	LE	EOF encountered before reaching specified line
EOS <address>	BOM/ASM	End of segment encountered. Address is the first free location
EOV ON INPUT FILE, MOUNT NEW TAPE THEN RESTART	DOM	EOV mark detected before EOF. Place new reel and restart
EOV ON OUTPUT FILE, MOUNT NEW TAPE THEN RESTART	DOM	The EOV mark detected on output device (magnetic tape or cassette tape). Mount a new reel and restart
ER	BOM/DOM	Operator message error
ER	COS	Loading impossible (RN command) or operator command. Push INT button and retype command.
ER 00	SCL	Command unknown
01	SCL	Syntax error.
02	SCL	Disc not operational
03	SCL	File code unknown
04	SCL	No PCT available
05	SCL	Read only save area overflow
06	SCL	Memory resident area overflow
07	SCL	Level error
08	SCL	Level already connected
09	SCL	Program unknown
10	SCL	Too many scheduled labels

Message	Program	Meaning
11	SCL	I/O error or too many scheduled labels
12	SCL	Program already declared previously
13	SCL	Program too long
14	SCL	Program has not been connected
15	SCL	Parameter error
16	SCL	The specified timer has not been assigned or program not connected to a timer
18	SCL	Program does not exist on disc
19	SCL	No 'activate block' can be built to activate background
21	SCL	Unknown file name
22	SCL	Non-disc file
23	SCL	File has already been catalogued.
24	SCL	No entry available in the library directory
26	SCL	D: CI File overflow
51	SCL	I/O error on disc
52	SCL	No spare entry available in FCT
53	SCL	No disc file description table free
54	SCL	Device unknown or disc file code unknown
55	SCL	Disc overflow or too many granules requested
56	SCL	File unknown
57	SCL	File code 2 unknown
58	SCL	More than 7 file codes assigned to the same disc file
ER 01	CFM COS	Cassette tape not assigned
ER 02	CFM COS	Dynamic catalogue overflow
ER 03	CFM COS	Bad volume or track loaded
ER 04	CFM COS	Incorrect labelling
ER 05	CFM COS	File already catalogued or previous file not yet closed with EOF
ER 06	CFM COS	I/O error on tape

Message	Program	Meaning
ER 07	CFM COS	Incompatible tape system
ER 08	CFM COS	Unknown names
ER 09	CFM COS	Wrong command syntax
ER 0A	COS	Unknown type of labelling
ER.MOD	LKE	Erroneous input module
ERR.MOD.	CCI	Error in assembly or compilation
ERR.LKE	LKE	A non-fatal error has occurred during this link-edit run
ERROR ASSIGN	CCI	Erroneous file code specified
ERROR IN PROCEDURE DEFINITION	DOM	Syntax error in catalogued procedure commands
ERROR IN PROCEDURE GENERATION	DOM	Error during execution of a procedure
EXIT	BOM	User program run completed
*F	ASM	Illegal FORM or XFORM directive
FATAL ERROR HAS OCCURRED. NO OBJECT CODE PRODUCED	ASM	A fatal error has occurred during assembly
FCT OVERFLOW	CCI	File code table overflow
FILE ALREADY CATALOGUED	CCI	This file was already kept
FILE CODE ABSENT	CCI	File code not specified in command
FILE CODE ERROR	CCI	Erroneous file code specified
FILE CODE MISSING	CCI	Parameter is not specified
FILE CODE NOT ASSIGNED	CCI	File code assigned to NO device or not yet assigned
FILE CODE UNKNOWN	CCI	Wrong file code specified
FILE NAME ERROR	CCI	First parameter is neither /S, nor a file code nor a character string
FILE NAME UNKNOWN	CCI	CCI did not recognise this file name
FILE NOT CATALOGUED	CCI	File to be deleted not catalogued
FILE OVERFLOW	CCI	File cannot accept more modules
FILE TYPE MISSING	CCI	Parameter missing
FILE TYPE ERROR	CCI	Parameter following <name> is not /S.
FIRST FILE CODE ERROR	CCI	Erroneous file code
FIRST FILE CODE MISSING	CCI	File code of disc to be copied not specified in command
FIRST FILE CODE UNKNOWN	CCI	File code not known by system

Message	Program	Meaning
FOR O/R <address>	ASM	Forward reference contained error: - value>255 for 8 least sign. bits - value specified was not absolute
I:	IPLRT	Initialization complete request for control message. Answer with WM or LD or ST
*I	ASM	Illegal identifier
******I	ASM	IDENT directive missing
IDENT <prog id><address>	BOM/IPL	Name and first address of loaded program
IDENT MISSING	CCI	IDENT record missing
IDENT TOO LONG	LKE	IDENT name too long
IDT.MIS	LKE	IDENT record missing
ILLEGAL EOS IN INPUT FILE	CCI	First record of the module is EOS
INPUT I/O ERROR	CCI	Input I/O error
INPUT COMMAND I/O ERROR	DOM	I/O error in user identification
INPUT DISK I/O ERROR	CCI	Error from the specified disc
INPUT FILE ASSIGN ERROR	CCI	Wrong assignment
INPUT FILE I/O ERROR	CCI	Input file I/O error
INPUT FILE CANNOT BE ASSIGNED	CCI	Temporary work file cannot be assiqned. The message is followed by the reason.
INVALID DISK FILE CODE	CCI	Second parameter not in range /F0 to /FF
INVALID DISK ADDRESS	CCI	Wrong address specified
INVALID DISK TYPE	CCI	Disc not supported by system
INVALID FILE CODE	CCI	Wrong file code specified
INVALID NAME	CCI	Module name not accepted
INVALID PARAM	CCI	Parameter not in range /01 to /EF
INVALID USERID	CCI	User identification does not begin with letter
INV.LGH name	CCI	Common block of this name too long
INV.IDT	LKE	Invalid IDENT record
I/O ER	LKE	I/O error encountered
I/O ERROR	CCI	I/O error. This message may follow DKER
I/O ERROR <file><status>	ASM/LKE	I/O error encountered
I/O ERROR IN CATALOG	CCI	I/O error during this operation
I/O ERROR ON LAST RECORD, [TRY AGAIN]	CCI	Type a new command from /01

Message	Program	Meaning
L:	IPLRT	Request for level to be used. Answer with LF CR or 2 digit hexa level number
*L	ASM	Illegal label
L:	LKE	Request for option/control message. See page
L?	LKE	Erroneous option/control message
L=<hexa value>	LKE	Length of relocatable program section
/L ASSIGN ERROR	CCI	/L cannot be assigned
/L EMPTY	CCI	/L file empty
LFT OVERFLOW	CCI	Disc locigal file table overflow
NO BP ON LKM/MLK	DBG	The breakpoint may not refer to these instructions
LIBRARY OPTION REDUNDANT	CCI	Library option is redundant
LINE NUMBER ERROR	CCI	Wrong line number specified
*M	ASM	Unknown mnemonic
M:	MON	Request control message after INT button pressed
MAP OPTION REDUNDANT	CCI	MAP option is redundant
MISSING PARAMETER	CCI	Parameter not specified
MODULE UNKNOWN	CCI	Object module unknown
M: PROC NOT CATALOGUED	DOM	No M: PROC file created
NS	BOM	No start address defined
NL OPTION ERROR	CCI	NL more than once declared
NO LABEL	CCI	Label on tape absent
NO LOAD MODULE	CCI	No load module in file
NO OBJECT LIBRARY	CCI	Object library not found
NO STRT.	LKE	Invalid IDENT record
*O	ASM	Erroneous displacement value
******O	ASM	Core overflow
/O ASSIGN ERROR	ASM	/O file cannot be assigned
/O CLOSE ERROR	CCI	Error during writing of EOF or rewinding of /O file
/O EMPTY	CCI	/O file empty
/O INPUT ERROR	CCI	Error during reading of /O
OBJECT LIBRARY ASSIGN ERROR	CCI	Work area for user object library cannot be assigned

Message	Program	Meaning
OBJECT MODULE NAME ERROR	CCI	Wrong module name specified
OBJECT MODULE NOT CATALOGUED	CCI	Object module not catalogued
OBJECT TAPE ON READER THINK OF BASE	IPL	Place object tape to be loaded on the paper tape reader Change program's base address if necessary
OUTPUT DISK I/O ERROR	CCI	I/O error on specified disc
OUTPUT FILE I/O ERROR	CCI	Output file I/O error
OUTPUT I/O ERROR	CCI	Output I/O error
OUTPUT NOT ASSIGNED	CCI	The /02 file is assigned to NO device or not assigned
OV	BOM	Insufficient memory available
OVL	IPLRT	Insufficient memory available
OVT	IPLRT	Insufficient table area remaining
*P	ASM	Illegal parameter
PARAM ABSENT	CCI	Parameter not specified
PARAM ERROR	CCI	Parameter error or error in sector number
PCT POOL SIZE?	IPLRT	Request for program control table size (type in 4 digit hexa number)
PARAMETER ERROR	DBG	Illegal parameter specified
PARAM MISSING	CCI	Type in all required parameters
PARTITIONING?	DRTM	Type in Y or N
PRG.OVL	CCI	Generated load module exceeds 32k
PROCEDURE IS NOT CATALOGUED	DOM	Procedure not catalogued in M: PROC file
PROCESSOR NOT CATALOGUED	CCI	A segment of processor or compiler not catalogued
PROG ABORTED AT <address>	DOM	Program aborted at this address. The message is followed by the reason of the abort, the contents of PSW and the contents of registers. The reason may be: POWER FAILURE NOT WIRED INSTRUCTION MEMORY PROTECT BUFFER AREA DESTROYED TOO MANY SCHEDULED LABELS OPERATOR ABORTED BUFFER ALLOCATION OVERFLOW DISK OVERFLOW DISK QUEUE OVERFLOW MEMORY OVERFLOW DURING LOADING PHASE

Message	Program	Meaning
PROGRAM ELAPSED TIME:	MON	The monitor types out the processing time for a specified program since the time specified by TIME:
PROGRAM NAME ERROR	CCI	Erroneous name specified
PROGRAM NOT CATALOGUED	CCI	Program not catalogued
PROGRAM SAVE AREA	DRTM	Type in length (4 hexa char.)
PU,<device name and address> <hardware status>,⌈RY⌉	MON	Peripheral error. For hardware status see control unit status word configuration
.R	ASM	Illegal relocation
R:	ASM	Request correct assembly statement
READ ONLY LENGTH:	DRTM	Define (4 hexa char.) Read Only area length. Min. /800.
READ ONLY SAVE AREA:	DRTM	Define (4 hexa char.) length of Read Only save area
REFUSED IN OFF-LINE MODE	DBG	The previous breakpoint must be terminated by GO or RT before a new breakpoint can be defined
REFUSED IN ON-LINE MODE	DBG	The IF command was given not immediately following an AT command
.S	ASM	Illegal statement
S:	CCI	Request control command
S=<address>	LKE	Start address of module
/S CANNOT BE ASSIGNED	CCI	/S cannot be assigned
/S ASSIGN ERROR:	CCI	/D4 cannot be assigned to source file
/S EMPTY	CCI	The file to be punched is empty
SC:	IPLRT	Request time for initialization. HH MM SS or LF CR
SD:	IPLRT	Request date for initialization. DD MM YY or LF CR
SECTOR DELETED	CCI	The sector was deleted previously
SECOND FILE CODE ERROR	CCI	Erroneous second file code specified
SECOND FILE CODE MISSING	CCI	File code of disc onto which is to be copied is missing
SECOND FILE CODE UNKNOWN	CCI	File code not known by system
SEGMENT NBR.01 MISSING	CCI	This parameter not specified
SEGMENT NBR NOT CATALOGUED	CCI	This segment was not catalogued or it is declared more than once
SEGMENT NBR ERROR	CCI	Erroneous segment

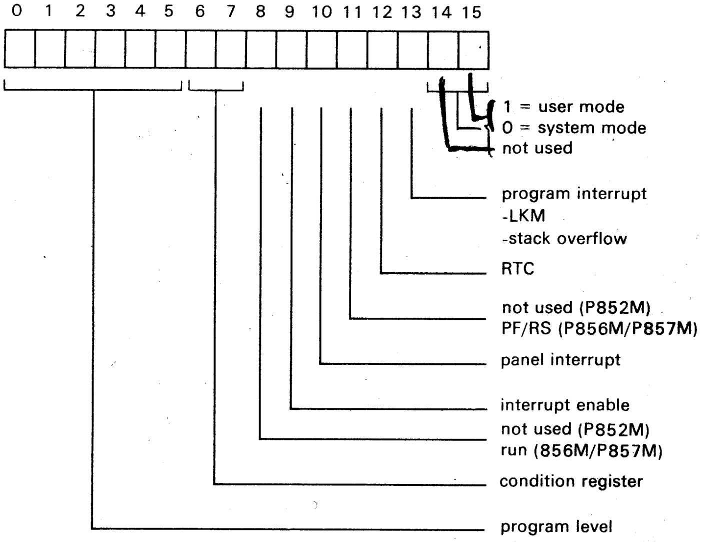
Message	Program	Meaning
SEQUENCE ERR, ⌈TRY AGAIN⌉	CCI	Update the lines in ascending order
START ADDR. REDUNDANT	CCI	Start address is redundant
SWAP AREA LENGTH	DRTM	Type in length (4 hexa char.)
SYMB. REF ERROR	DBG	Reference to entry point not valid
SYNTAX ERR, ⌈TRY AGAIN⌉	LE	Error in Line Editor command
SYNTAX ERROR	DBG	Erroneous syntax in command
SYSTEM DISK I/O ERROR	CCI	I/O error
SYSTEM LIB ASSIGN ERROR	CCI	Assign error in system library
SYSTEM SESSION COMMAND	CCI	LIC (List Catalogue) may only be used in system session
TABLE O'FLOW, TRY AGAIN	LE	Character string table overflow
TBL.OVL	LKE	Not enough space to link-edit modules
TIME:	DOM/ DRTM	Disc system asking for time (h, m, s, or LF CR)
T.O.	LKE	Insufficient table area remaining
TOO MANY FILE CODE EQU	CCI	More than 7 file codes assigned to one disc file
TOO MANY MODULES TO BE ASSIGNED	CCI	More than 18 modules to be assigned
TOO MANY PARAM	CCI	Too many parameters specified
U:	UPD	Request control message. See page
U?	UPD	Erroneous control message
UND.ENT <number>	ASM	Number of undefined entry points
UND.LAB <references>	ASM	Missing labels in module
UNKNOWN COMMAND ⌈TRY AGAIN⌉	CCI	Erroneous update command given
UNKNOWN USERID	CCI	Specified userid not recognised by system
UNS.EXT.	LKE	One or more unsatisfied external references
USER DISK I/O ERROR	CCI	I/O error
USER LIB ASSIGN ERROR	CCI	Erroneous assignment
USERID:	DOM	Request for identification. Reply with <disc no>,<userid> or <userid>
USERID ABSENT	CCI	No userid given in command
USERID ALREADY CATALOGUED	CCI	CCI specified userid was not catalogued

Message	Program	Meaning
USERID ERROR	CCI	Error in userid or the first parameter is not a userid
USERID MISSING	CCI	No parameter given
USERID NOT CATALOGUED	CCI	The userid was not catalogued
USERID UNKNOWN	CCI/DOM	Userid not found on the disc
*X	ASM	Illegal expression
XN	LKE	Unsatisfied external reference

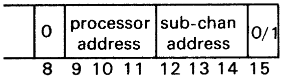
STANDARD INTERRUPT LEVELS

level	0	Power Failure/Automatic Restart
	1	LKM/Stack Overflow
	2	Real Time Clock
	3	Reserved
	4	PTR
	5	PTP
	6	ASR
	7	Control Panel
	8 to /F	Free
	/10	Disc
	/11	Disc
	/12	Disc
	/13	MT
	/14	TK, TL
	/15	CR
	/16	PL
	/17	LP
	/18 to/1F	Free

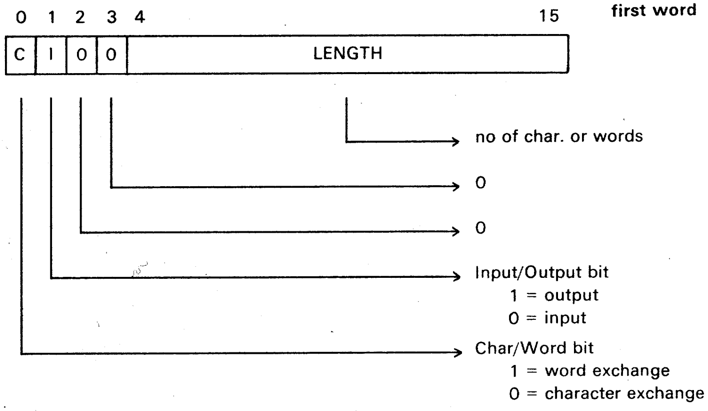
PROGRAM STATUS WORD

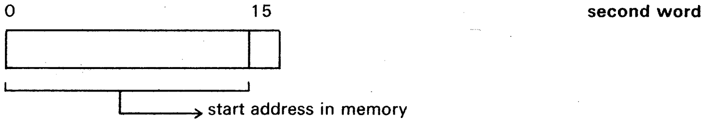


FORMAT OF RER/WER INSTRUCTIONS



FORMAT OF CONTROL WORDS (FOR WER)





bit 15 = 1 right hand char. addressed
0 left hand char. addressed

(Only if bit 0 in 1st word = 0)

Powers of 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15
18 446 744 073 709 551 616	16

Powers of 2

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

		AND	OR	XOR
p	q	p ∧ q	p ∨ q	p ∀ q
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

char.	ASCII octal	Intern Hexa	char. set punch comb.	char.	ASCII octal	Intern Hexa	char. set punch comb.
space	240	20	on punch	D	304	44	12,4
!	241	21	11,8,2	E	305	45	12,5
"	242	22	8,7	F	306	46	12,6
#	243	23	8,3	G	307	47	12,7
$	244	24	11,8,3	H	310	48	12,8
%	245	25	0,8,4	I	311	49	12,9
&	246	26	12	J	312	4A	11,1
'	247	27	8,5	K	313	4B	11,2
(	250	28	12,8,5	L	314	4C	11,3
)	251	29	11,8,5	M	315	4D	11,4
*	252	2A	11,8,4	N	316	4E	11,5
+	253	2B	12,8,6	O	317	4F	11,6
,	254	2C	0,8,3	P	320	50	11,7
–	255	2D	11	Q	321	51	11,8
.	256	2E	12,8,3	R	322	52	11,9
/	257	2F	0,1	S	323	53	0,2
0	260	30	0	T	324	54	0,3
1	261	31	1	U	325	55	0,4
2	262	32	2	V	326	56	0,5
3	263	33	3	W	327	57	0,6
4	264	34	4	X	330	58	0,7
5	265	35	5	Y	331	59	0,8
6	266	36	6	Z	332	5A	0,9
7	267	37	7	[	333	5B	
8	270	38	8	\	334	5C	
9	271	39	9	]	335	5D	
:	272	3A	8,2	↑	336	5E	
;	273	3B	11,8,6	←	337	5F	
<	274	3C	12,8,4				
=	275	3D	8,6	Bell	207	07	
>	276	3E	0,8,6	Linefeed	212	0A	
?	277	3F	0,8,7	Car.Ret.	215	0D	
@	300	40	8,4	X on reader	221	11	
A	301	41	12,1	X off reader	223	13	
B	302	42	12,2	Rubout	377	7F	
C	303	43	12,3	X on punch	222	12	
				X off punch	224	14	
				FF		0C	

HEXADECIMAL - DECIMAL CONVERSION

DOUBLE WORD

WORD 2								WORD 1							
HALFWORD 4				HALFWORD 3				HALFWORD 2				HALFWORD 1			
bits: 0123		bits: 4567		bits: 0123		bits: 4567		bits: 0123		bits: 4567		bits: 0123		bits: 4567	
hex	*decimal*	*hex*	*decimal*	*hex*	*decimal*	*hex*	*decimal*	*hex*	*decimal*	*hex*	*decimal*	*hex*	*decimal*	*hex*	*decimal*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	268,435,456	1	16,777,216	1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	536,870,912	2	33,554,432	2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	805,306,368	3	50,331,648	3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	1,073,741,824	4	67,108,864	4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	1,342,177,280	5	83,886,080	5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	1,610,612,736	6	100,663,296	6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	1,879,048,192	7	117,440,512	7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	2,147,483,648	8	134,217,728	8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	2,415,919,104	9	150,994,944	9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	2,864,354,560	A	167,772,160	A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	2,952,690,016	B	184,549,376	B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	3,221,225,472	C	201,326,592	C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	3,489,660,928	D	218,103,808	D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	3,758,096,384	E	234,881,024	E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	4,026,531,840	F	251,658,240	F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
8		7		6		5		4		3		2		1	

ASCII CODE

space	●		●			·			
!	●		●			·			●
"	●		●			·		●	
#	●		●			·		●	●
$	●		●			·	●		
%	●		●			·	●		●
&	●		●			·	●	●	
`	●		●			·	●	●	●
(	●		●		●	·			
)	●		●		●	·			●
*	●		●		●	·		●	
+	●		●		●	·		●	●
'	●		●		●	·	●		
–	●		●		●	·	●		●
.	●		●		●	·	●	●	
/	●		●		●	·	●	●	●
0	●		●	●		·			
1	●		●	●		·			●
2	●		●	●		·		●	
3	●		●	●		·		●	●
4	●		●	●		·	●		
5	●		●	●		·	●		●
6	●		●	●		·	●	●	
7	●		●	●		·	●	●	●
8	●		●	●	●	·			
9	●		●	●	●	·			●
:	●		●	●	●	·		●	
;	●		●	●	●	·		●	●
<	●		●	●	●	·	●		
=	●		●	●	●	·	●		●
>	●		●	●	●	·	●	●	
?	●		●	●	●	·	●	●	●
@	●	●				·			
A	●	●				·			●
B	●	●				·		●	
C	●	●				·		●	●
D	●	●				·	●		
E	●	●				·	●		●
F	●	●				·	●	●	
G	●	●				·	●	●	●
H	●	●			●	·			
I	●	●			●	·			●
J	●	●			●	·		●	
K	●	●			●	·		●	●
L	●	●			●	·	●		
M	●	●			●	·	●		●
N	●	●			●	·	●	●	
O	●	●			●	·	●	●	●
P	●	●		●		·			
Q	●	●		●		·			●
R	●	●		●		·		●	
S	●	●		●		·		●	●
T	●	●		●		·	●		
U	●	●		●		·	●		●
V	●	●		●		·	●	●	
W	●	●		●		·	●	●	●
X	●	●		●	●	·			
Y	●	●		●	●	·			●
Z	●	●		●	●	·		●	
[	●	●		●	●	·		●	●
*/	●	●		●	●	·	●		
]	●	●		●	●	·	●		●
↑	●	●		●	●	·	●	●	
** ←	●	●		●	●	·	●	●	●
CR	●				●	·	●		●
Line feed	●		●		●	·		●	
X on	●		●	●		·			●
Bell	●		●			·	●	●	●
X off	●		●	●		·		●	●

* delete record

** delete character (EOR)