September 1973
A publication of:
Philips-Electrologica B.V.
Main Marketing Group OEM
P.O. Box 245 Apeldoorn, Netherlands
Copyright © by Philips-Electrologica B.V.

Printed in The Netherlands

5122 991 16902

PHILIPS

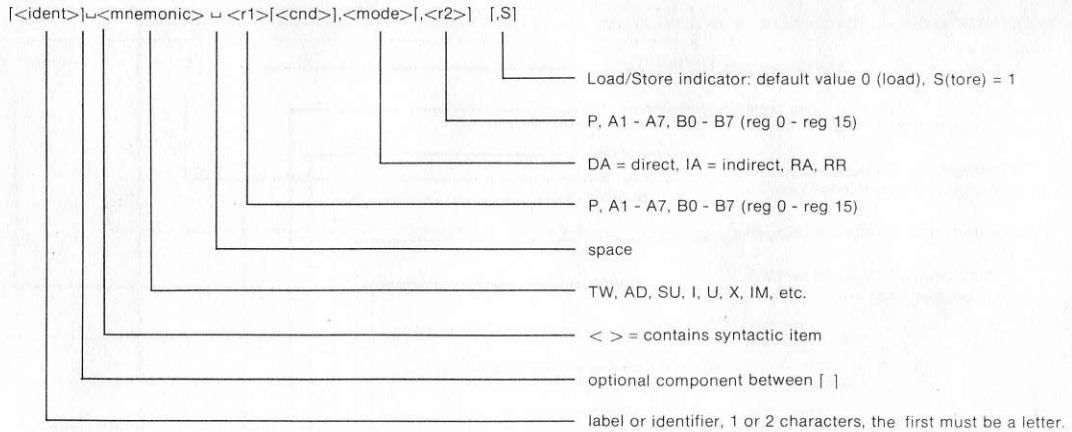
P850M/P855M

Hardware Reference Data

PHILIPS Data Systems

CONTENTS

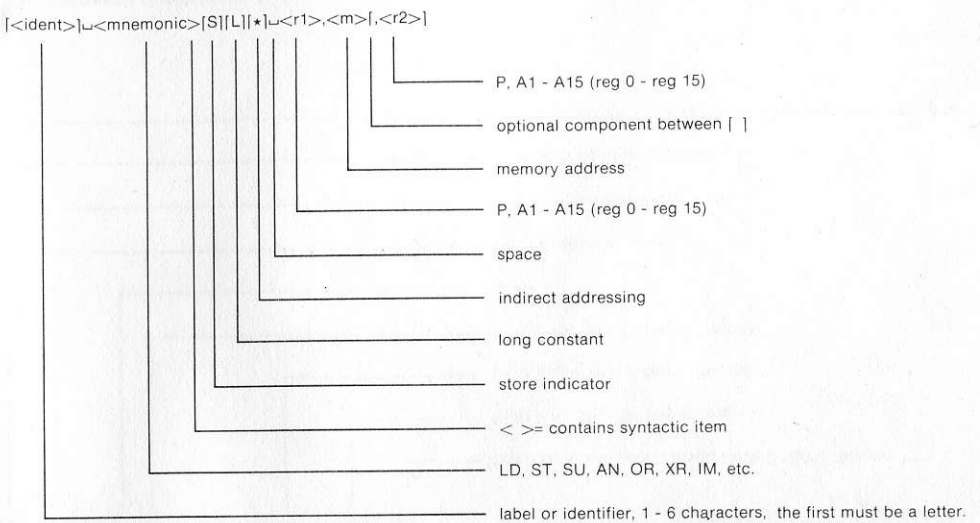
ASSEMBLY LANGUAGE P850M - GENERAL FORMAT - 1)



* in operand field represents current value of location counter

1) For details, see P850M User Manual (publication number 5122 991 1453X)

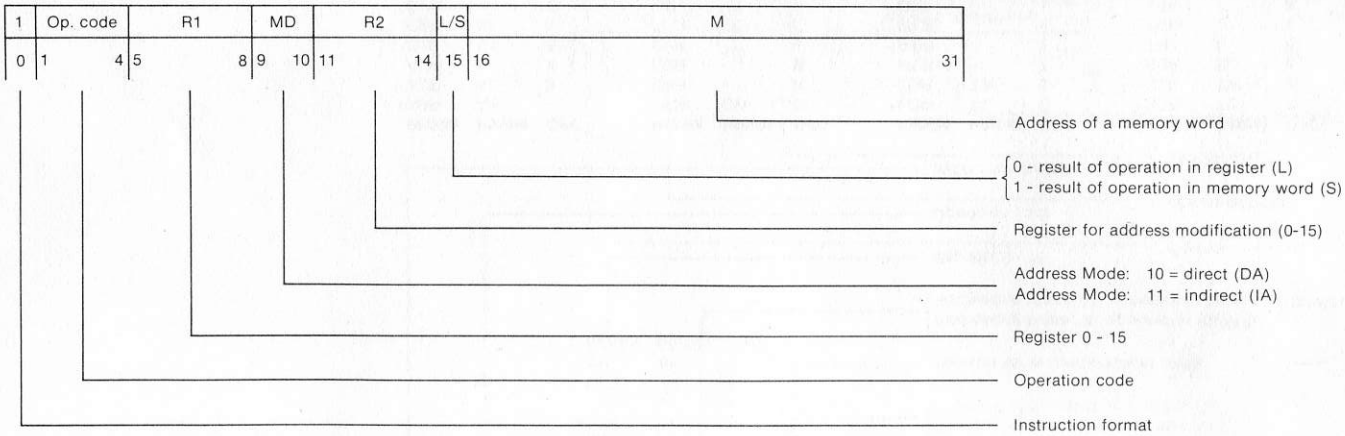
ASSEMBLY LANGUAGE P855M - general format - 2)



* in operand field represents current value of location counter (except when immediately after mnemonic)

2)For details, see P855M/P860M System Software Manual Paper Tape or Disc (publication number 5122 991 1155X or 5122 991 1162X)

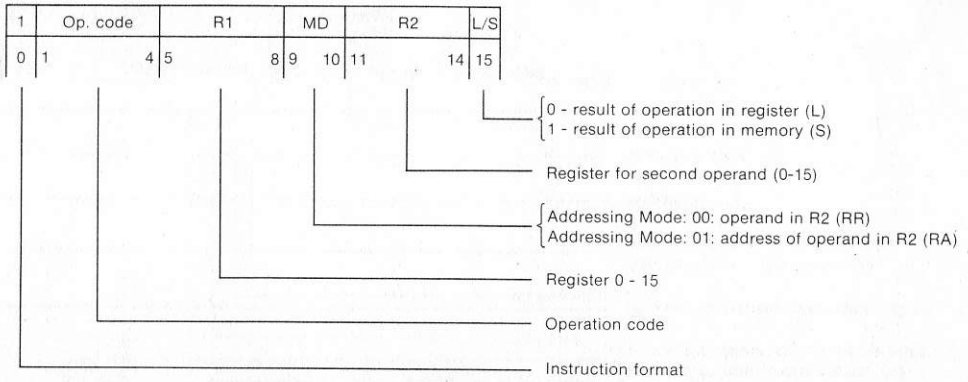
Memory Reference Instructions



P855M	P850M	OPC	P855M	P850M	OPC	P855M	P850M	OPC	P855M	P850M	OPC
*ABI	AB	1	*CFI	CF	E	*LC	TC	C	*RB	RB	B
AD	AD	2	CW	CW	D	LD	TW	0	*RF	RF	A
AN	I	4	DA	-	A	ML	-	7	*SC	TC	C
C1	C1	F	DS	-	B	*MS	-	7	*ST	TW	0
C2	C2	3	DV	-	9	MU	-	8	SU	SU	3
CC	-	D	IM	IM	2	OR	U	5	XR	X	6

*No CR setting

Register/Register Instructions



P855M	P850M	OPC
*ABR	AB	1
ADR	AD	2
ANR	I	4
C1R	C1	F
C2R	C2	3
CCR	–	D
*CFR	CF	E

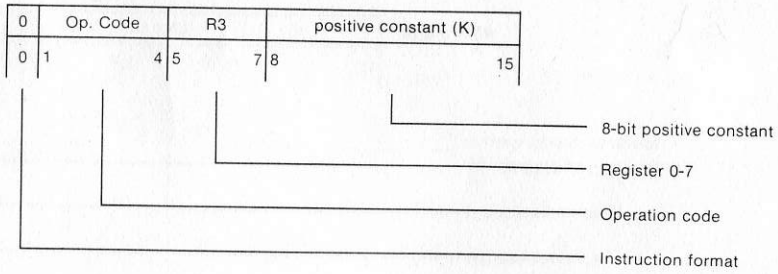
P855M	P850M	OPC
CWR	CW	D
DAR	–	A
DSR	–	B
DVR	–	9
ECR	EC	C
IMR	IM	2

P855M	P850M	OPC
*LCR	TC	C
LDR	TW	0
MLR	–	7
*MSR	–	7
MUR	–	8
ORR	U	5
*RTN	RT	E

P855M	P850M	OPC
*SCR	TC	C
*STR	TW	0
SUR	SU	3
TM	I	4
TNM	U	6
XRR	X	6

*No CR setting

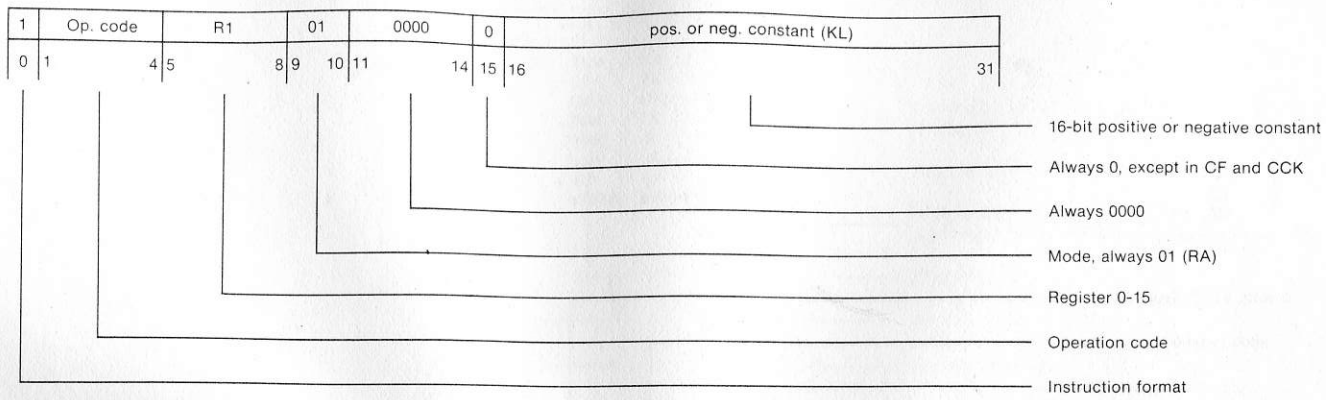
Constant instructions (short format)



P855M	P850M	OPC
*AB	BK	1
ADK	AK	2
ANK	IK	4
*LDK	LK	0
ORK	UK	5
SUK	SK	3
XRK	XK	6

*No CR setting

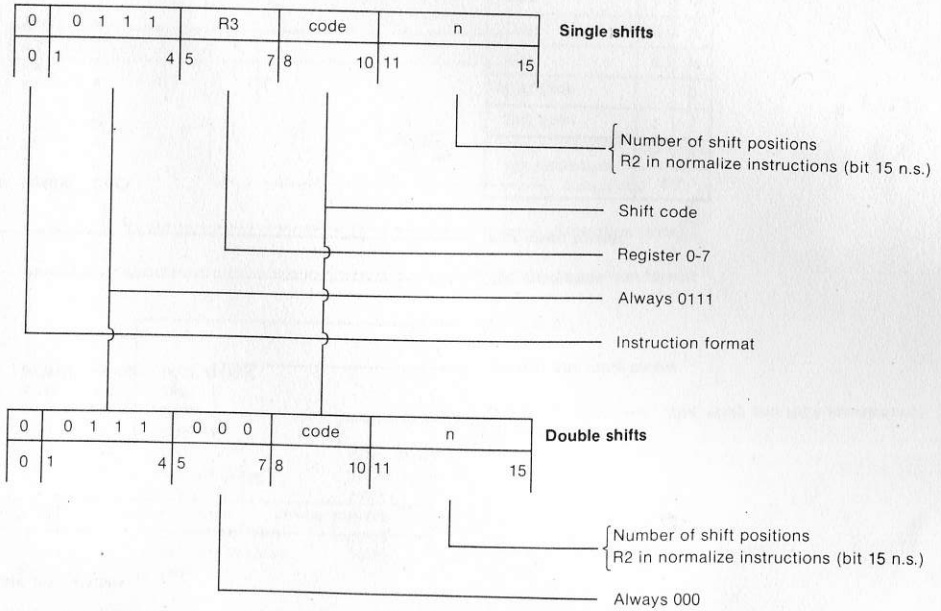
Constant Instructions (long format)



P855M	P850M	OPC	P855M	P850M	OPC
*ABL	AB	1	DVK	–	9
ADKL	AD	2	LDKL	TW	0
ANKL	I	4	*LCK	TC	C
CCK	–	D	MLK	–	7
*CF	CF	E	MUK	–	8
CWK	CW	D	ORKL	U	5
DAK	–	A	SUKL	SU	3
DSK	–	B	XRKL	X	6

*No CR setting

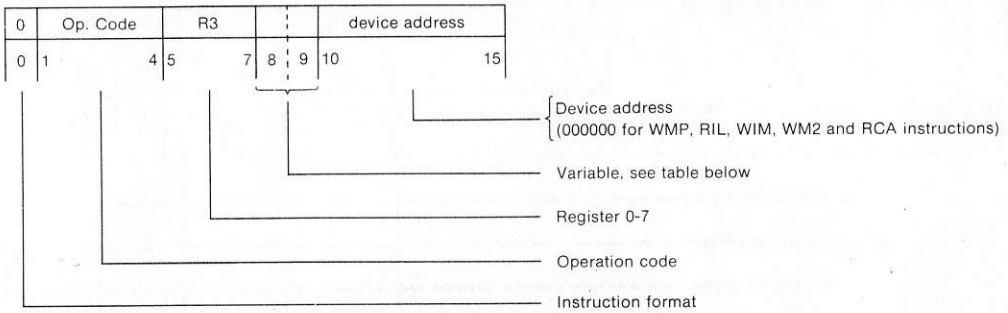
Shift Instructions



P855M	P850M	Shift Code
DLA	–	0
DLC	–	6
DLL	–	2
*DLN	–	4
DRA	–	1
DRC	–	7
DRL	–	3
*DRN	–	5
SLA	–	0
–	LA	0
SLC	–	6
SLL	–	2
–	LL	2
SLN	–	4
SRA	–	1
–	RA	1
SRC	–	7
–	RC	7
SRL	–	3
*SRN	–	5

*No CR setting

Input/Output Instructions

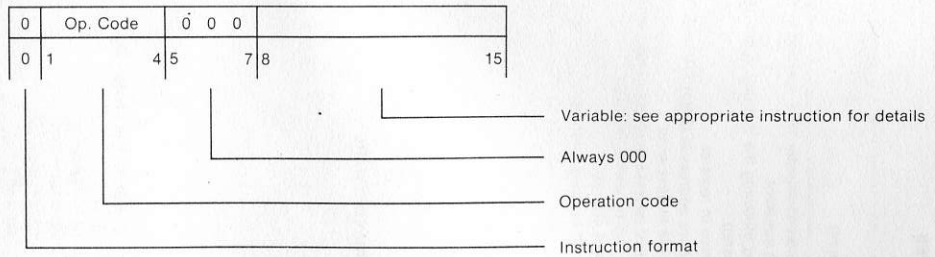


P855M	P850M [1]	OPC
CIO	CT	8
INR	IN	9
OTR	OT	8
RCA	-	9
RIL	RL	9
SST	SS	9
TST	TS	9
WIM	WM	8
WM2	-	8
WMP	-	8

1) Mnemonics are not recognized by the Assembler

I/O Instruction	bit	
	8	9
CIO Start	1	1
CIO Stop	1	0
INR	0	X
OTR	0	X
TST	1	0
SST	1	1

Miscellaneous instructions



P855M	P850M	HEX. CODE	
ENB	EN	2840	
HLT	HT	207F	
INH	IH	20BF	
LKM	LM	2804	
RIT	RI	control panel	20DF
		PFAURE	20EF
		RT clock	20FE
		prog. error/LKM	20FB
		memory protect	20FD
SMD	-	2801	

1) Mnemonics are not recognized by the Assembler

ADDRESSING MODES - REFERENCE TYPES

Mode	MD 9	MD 10	R2	Effective Addresses	Meaning
RR	0	0	xxxx	R2	Register-to-register (2nd operand in R2)
RA	0	1	0000	(P)	Long constant. (word following the instruction is a 16-bit operand).
RA	0	1	≠0	(R2)	Address in register (effective address is in R2)
DA	1	0	0000	(q)	Address in next word
DA	1	0	≠0	(q) + (R2)	Indexed address (address in next word is modified by contents of R2).
IA	1	1	0000	[(q)]	Indirect address
IA	1	1	≠0	[(q) + (R2)]	Indirect indexed address

P = instruction counter
q = contents of P (i.e. word after instruction)
R2 = bits 11 - 14 of instruction word
() = contents of
[ ] = indirect addressing.

FILE CODES TO BE USED WITH STANDARD SOFTWARE

a P850M File codes

01 = Source input file
02 = Listing output file
03 = Punch output file
04 = Object code input file
05 = Operator's typewriter

b P855M File codes - Basic

01 = Standard Source Input
02 = Standard Listing Output
03 = Standard Punch Output
04 = Standard Object Input
05 = Standard Operator Keyboard
06 = ASR Reader
07 = ASR Punch
08 = Punched Tape Reader
09 = Tape Punch
0A = Line Printer
0B = Card Reader
0C = Cassette Tape.

c P855M File codes - Disc

01 = Typewriter
02 = Print unit
03 = Punch unit
D4 = /S
D5 = /0 } for DOM
D6 = /L
D7 = System object file
D8 = User object file
E0 = Control command input
E1 = Source input
E2 = Object input
EF = Typewriter (display only)
F0 – FF = Disc units

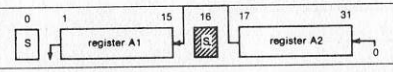
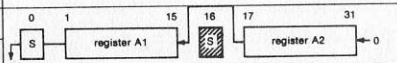
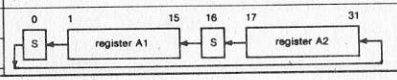
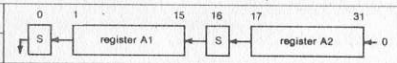
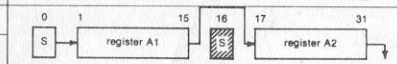
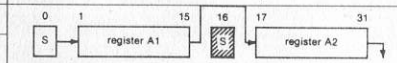
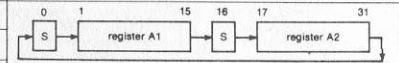
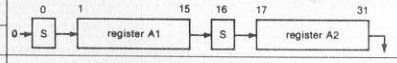
DIRECTIVES

```
[<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>, ....,
          <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 P850M	mnemonic P855M	format	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) P850M m.c.	P855M l.c.	P855M m.c.	remarks
Absolute branch	AB	ABI	1	0001	10	0	(M) → P		7		3	bits 5-7: condition
					10	0	(M + (R2)) → P		7		3	bit 8: n.s.
					11	0	((M)) → P		9		4	
					11	0	((M + (R2))) → P		9		4	
						0	no branch: (P) + 4 → P		3	1	1	
Absolute conditional branch (with constant)	BK	AB	0	0001	-	n.s.	K → P (q) → P	3)	4	1	1	short format
							no branch: (P) + 2 → P		3	1	1	
Absolute conditional branch (with constant)	AB	ABL	1	0001	01	0	KL → P		5	1	2	long format
							no branch: (P) + 2 → P		3	1	1	
Absolute conditional branch to register	AB	ABR	1	0001	00	n.s.	(R2) → P		4	1	1	bits 5-7: condition
					01	0	((R2)) → P		5		2	bit 8: n.s.
							no branch: (P) + 2 → P		3	1	1	
Add constant	AK	ADK	0	0010	-	-	(R3) + K → R3		4	1	1	short; R1 = 1111: system mode
	AD	ADKL	1	0010	01	0	(R1) + KL → R1		5		2	long
Addition	AD	AD	1	0010	10	0	(R1) + (M) → R1	2)	7		3	R1 = 1111: system mode
					10	1	(R1) + (M) → M	2)	8		4	when l/s bit = 1,
					10	0	(R1) + (M + (R2)) → R1	2)	7		3	R1 must be ≠ 0
					10	1	(R1) + (M + (R2)) → M	2)	8		4	
					11	0	(R1) + ((M)) → R1	2)	9		4	
					11	1	(R1) + ((M)) → M	2)	10		5	
					11	0	(R1) + ((M + (R2))) → R1	2)	9		4	
					11	1	(R1) + ((M + (R2))) → M	2)	10		5	
Addition/register	AD	ADR	1	0010-	00	n.s.	(R1) + (R2) → R1	2)	4	1	1	when l/s bit = 1,
					01	0	(R1) + ((R2)) → R1	2)	5		2	R1 must be ≠ 0;
					01	1	(R1) + ((R2)) → (R2)	2)	6		3	R1 = 1111: system mode
Call function	CF	CFI	1	1110			(P) → (R1), (R1) - 2 → R1 P855/P860					R1 = 1111: system mode
							(PSW) → (R1), (R1) - 2 → R1					
							(CR) → (R1), (R1) - 2 → R1 P850					
							(P) → (R1), (R1) - 2 → R1					
							then:					
					10	1	(M) → P		13	2	5	if reg. 15 = stack
					10	1	(M + (R2)) → P		13	2	5	pointer,
					11	1	((M)) → P		15	2	6	$>128_{10}$ → overflow
					11	1	((M + (R2))) → P		15	2	6	
Call function/constant	CF	CF	1	1110	01	1	(P) → (R1), (R1) - 2 → R1 P855/60	3)	11	2	4	if reg. 15 = stack
							(PSW) → R1, (R1) - 2 → R1,					pointer
							KL → P					$>128_{10}$ → overflow;
							(CR) → R1, (R1) - 2 → R1					
							(P) → R1, (R1) - 2 → R1 KL → P P850					R1 = 1111: system mode

name (in alphabetical order)	mnemonic P850M	mnemonic P855M	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) P850M m.c.	P855M l.c.	P855M m.c.	remarks
Call function/register	CF	CFR	1	1110			(P) → (R1), (R1) - 2 ⟶ R1 } P855/60					if reg. 15 = stack
							(PSW) → (R1), (R1) - 2 → R1 }					pointer,
							(CR) → (R1), (R1) - 2 ⟶ R1 } P850					$>128_{10}$: overflow
							(P) → (R1), (R1) - 2 ⟶ R1 }	3)				
							then:					R1 = 1111: system mode
					00	1	(R2) → P		10	3	3	
					01	1	((R2)) → P		11	2	4	
Compare characters	-	CC	1	1101	10	1	(R1)r ÷ (M) l/r ⟶ CR				4	R1 = 1111: system mode
					10	1	(R1)r ÷ (M + (R2)) l/r → CR				4	
					11	1	(R1)r ÷ ((M)) l/r ⟶ CR				5	
					11	1	(R1)r ÷ ((M + (R2))) l/r → CR				5	
Compare characters	-	CCR	1	1101	01	1	(R1)r ÷ ((R2)) l/r ⟶ CR				3	R1 = 1111: system mode
register/register								4)				
Compare character	-	CCK	1	1101	01	1	(R1)r ÷ KLI → CR		-		3	R1 = 1111: system mode
with constant												
Compare words	CW	CW	1	1101	10	0	(R1) ÷ (M) ⟶ CR		7		3	R1 = 1111: system mode
					10	0	(R1) ÷ (M + (R2)) → CR		7		3	
					11	0	(R1) ÷ ((M)) ⟶ CR		9		4	
					11	0	(R1) ÷ ((M + (R2))) → CR		9		4	
Compare words	CW	CWR	1	1101	00	n.s.	(R1) ÷ (R2) ⟶ CR		4	1	1	R1 = 1111: system mode
register/register					01	0	(R1) ÷ ((R2)) ⟶ CR		5		2	
Compare word	CW	CWK	1	1101	01	0	(R1) ÷ KL ⟶ CR		5		2	R1 = 1111: system mode
with constant												
Control Input/Output	CT	CIO	0	1000	-	-	Start (bit 9=1) or stop (bit 9=0)	5)	4		3	bit 8 = 1
							any I/O operation					system mode
							quotient remainder					
Divide	-	DV	1	1001	10	0	(A1, A2) / (M) ⟶ A2 A1		-	10.5	3	
					10	0	(A1, A2) / (M + (R2)) → A2 A1		-	10.5	3	
					11	0	(A1, A2) / ((M)) ⟶ A2 A1		-	10.5	4	
					11	0	(A1, A2) / ((M + (R2))) → A2 A1		-	10.5	4	
							quotient remainder					
Divide by constant	-	DVK	1	1001	01	0	(A1, A2) / KL ⟶ A2 A1		-	10.5	2	
							quotient remainder	2)				
Divide registers/registers	-	DVR	1	1001	00	0	(A1, A2) / (R2) ⟶ A2 A1		-	11.5	1	
					01	0	(A1, A2) / ((R2)) ⟶ A2 A1		-	10.5	2	
Double Add	-	DA	1	1010	10	0	(M, M + 1) + (A1, A2) → A1, A2		-	1	4	
					10	0	(M + (R2), M + (R2) + 1) + (A1, A2) → A1, A2		-	1	4	
					11	0	((M), (M) + 1) + (A1, A2) → A1, A2		-	1	5	
					11	0	((M + (R2)), ((M + (R2) + 1) + (A1, A2) → A1, A2		-	1	5	

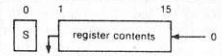
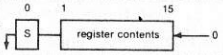
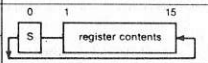
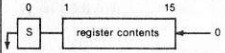
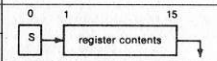
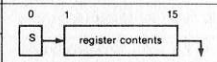
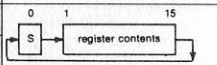
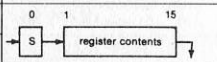
name (in alphabetical order)	mnemonic P850M	mnemonic P855M	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles[6] P850M m.c.	execution time in cycles[6] P855M l.c.	execution time in cycles[6] P855M m.c.	remarks
Double add registers/	-	DAR	1	1010	00	0	(R2, R2 + 1) + (A1, A2) ⟶ A1, A2		-	3	1	
registers					01	0	((R2), (R2 + 1)) + (A1, A2) → A1, A2		-	1	3	
Double add with constant	-	DAK	1	1010	01	0	KL + (A1, A2) → A1, A2		-	1	3	
Double subtract	-	DS	1	1011	10	0	(A1, A2) - (M, M + 1) ⟶ A1, A2		-	1	4	
					10	0	(A1, A2) - (M + (R2), M + (R2) + 1) → A1, A2		-	1	4	
					11	0	(A1, A2) - ((M), (M) + 1) ⟶ A1, A2	2)	-	1	5	
					11	0	(A1, A2) - ((M + (R2)), ((M + (R2)) + 1) → A1, A2		-	1	5	
Double subtract	-	DSR	1	1011	00	0	(A1, A2) - (R2, R2 + 1) ⟶ A1, A2		-	3	1	
registers/registers					01	0	(A1, A2) - ((R2), (R2 + 1)) → A1, A2		-	1	3	
Double subtract	-	DSK	1	1011	01	0	(A1, A2) - KL → A1, A2		-	1	3	
with constant												
Double left and	-	DLN	1	0111	-	n.s.		3)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 100
normalize shift												bits 11-14: R2;
												bit 15: n.s.
Double left arithmetic shift	-	DLA	0	0111	-	-		2)	-	$(1.5+\frac{n}{2})$		bits 8-10: 000
Double left circular shift	-	DLC	0	0111	-	-		1)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 110
Double left logical shift	-	DLL	0	0111	-	-			-	$(1.5+\frac{n}{2})$	1	bits 8-10: 010
Double right and	-	DRN	0	0111	-	-		3)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 101
normalize shift												bits 11-14: R2;
												bit 15: n.s.
Double right arithmetic	-	DRA	0	0111	-	-				$(1.5+\frac{n}{2})$	1	bits 8-10: 001
shift												
Double right circular shift	-	DRC	0	0111	-	-		1)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 111
Double right logical shift	-	DRL	0	0111	-	-			-	$(1.5+\frac{n}{2})$	1	bits 8-10: 011
Enable interrupt	EN[7]	ENB	0	0101	-	-	machine status = 'permit interrupt'		4	1	1	bits 8-15: 01000000
Exchange characters	EC	ECR	1	1100	00	n.s.	(R2) → (R1)r; (R2)r → (R1)	3)	5	1	1	R1 = 1111: system mode
register/register												
Exclusive OR	X	XR		0110	10	0	(R1) ⩛ (M) → R1		7		3	R1 = 1111: system mode
					10	1	(R1) ⩛ (M) → M		8		4	
					10	0	(R1) ⩛ (M + (R2)) → R1		7		3	
					10	1	(R1) ⩛ (M + (R2)) → M + (R2)	1)	8		4	
					11	0	(R1) ⩛ ((M))→R1		9		4	
					11	1	(R1) ⩛ ((M))→(M)		10		5	
					11	0	(R1) ⩛ ((M + (R2)))→R1		9		4	
					11	1	(R1) ⩛ ((M + (R2)))→(M + (R2))		10		5	

name (in alphabetical order)	mnemonic P850M	mnemonic P855M	format	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles[6] P850M m.c.	P855M l.c.	P855M m.c.	remarks
Exclusive OR	X	XRR	1	0110	00	0	(R1) ∀ (R2) → R1		4	1	2	R1 = 1111: system mode
register/register					01	0	(R1) ∀ ((R2)) → R1		5		2	
					01	1	(R1) ∀ ((R2)) → R2	1)	6		3	
Exclusive OR	XK	XRK	0	0110	-	-	$(R3)_{8-15}$ ∀ K → $R3_{8-15}$		4	1	1	short; R1 = 1111: system
with constant												mode
Exclusive OR	X	XRKL	1	0110	01	0	(R1) ∀ KL → R1		5		2	long; R1 = 1111: system
with constant												mode
Halt	HT[7]	HLT	0	0100	-	-	machine → 'halt' mode	3)	4	1	1	bits 8-15: 01111111;
												system mode
Increment Memory	IM	IM	1	0010	10	1	(M) + 1 → M		8		4	bits 5-8: 0000
					10	1	(M + (R2)) + 1 → M		8		4	
					11	1	((M +)) + 1 → M	2)	10		5	
					11	1	((M) + (R2))) + 1 → (M + (R2))		10		5	
Increment memory/	IM	IMR	1	0010	01	1	((R2)) + 1 → R2		6		3	bits 5-8: 0000
register												
Inhibit interrupt	IH[7]	INH	0	0100	-	-	machine status = 'prohibit all interrupts'	3)	4	1	1	bits 8-15: 10111111
												system mode
Input to register	IN[7]	INR	0	1001	-	-	word/character from device → R3	5)	4		3	bit 8 = 0; system mode
Link to monitor	LM[7]	LKM	0	0101	-	-	user mode → system mode		4	1	1	bits 8-15: 00000100
Load character	TC	LC	1	1100	10	0	(M) l/r → R1r		7		3	R1 = 1111: system mode
					10	0	(M + (R2)) l/r → R1r	3)	7		3	R1 must be =/0
					11	0	((M)) l/r → R1		9		4	
					11	0	((M + (R2))) l/r → R1r		9		4	
Load character/constant	TC	LCK	1	1100	01	0	KL l → R1r		5		2	R1 = 1111: system mode
Load character/register	TC	LCR	1	1100	01	0	((R2)) l/r → R1r		5		2	R1 = 1111: system mode
Load constant	LK	LDK	0	0000	-	-	K → $R3_{8-15}$, 0 → $R3_{0-7}$	3)	4	1	1	short; R1 = 1111: system
												mode
	TW	LDKL	1	0000	01	0	KL → R1		5		2	long; R1 = 1111: system
												mode
Load register	TW	LD	1	0000	10	0	(M) → R1		7		3	R1 = 111: system mode
					10	0	(M + (R2)) → R1		7		3	
					11	0	((M)) → R1		9		4	
					11	0	((M + (R2))) → R1		9		4	
Load register/register	TW	LDR	1	0000	00	n.s.	(R2) → R1	1)	4	1	1	R1 = 1111: system mode
					01	0	((R2)) → R1		5		2	
					01	0	(A15) + 2 → A15, ((A15)) → R1		6	1	2	system mode
Logical AND	I	AN	1	0100	10	0	(R1) ∧ (M) → R1		7		3	R1 = 1111: system mode
					10	1	(R1) ∧ (M) → M		8		4	
					10	0	(R1) ∧ (M + (R2)) → R1		7		3	
					10	1	(R1) ∧ (M + R2)) → M + (R2)		8		4	

name (in alphabetical order)	mnemonic P850M	mnemonic P855M	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) P850M m.c.	P855M l.c.	P855M m.c.	remarks
					11	0	(R1) ∧ ((M)) ⟶ R1		9		4	
					11	1	(R1) ∧ ((M)) ⟶ (M)		10		5	
					11	0	(R1) ∧ ((M + (R2))) → R1		9		4	
					11	1	(R1) ∧ ((M + (R2))) → (M + (R2))		10		5	
Logical AND	I	ANR	1	0100	00	0	(R1) ∧ (R2) ⟶ R1		4	1	1	R1 = 1111: system mode
register/register					01	0	(R1) ∧ ((R2)) → R1		5		2	
					01	1	(R1) ∧ ((R2)) → (R2)		6		3	
Logical AND	IK	ANK	0	0100	-	-	$(R3)_{8-15}$ ∧ K ⟶ $R3_{8-15}$		4	1	1	short; R1 = 1111: system
with constant	I	ANKL	1	0100	01	0	(R1) ∧ KL ⟶ R1	1)	5		2	long; mode
Logical OR	U	OR	1	0101	10	0	(R1) ∨ (M) ⟶ R1		7		3	R1 = 1111: system mode
					10	1	(R1) ∨ (M) ⟶ M		8		4	R1 = 1111: system mode
					10	0	(R1) ∨ (M + (R2)) ⟶ R1		7		3	
					10	1	(R1) ∨ (M + (R2)) ⟶ M + (R2)		8		4	
					11	0	(R1) ∨ ((M)) ⟶ R1		9		4	
					11	1	(R1) ∨ ((M)) ⟶ (M)		10		5	
					11	0	(R1) ∨ ((M + (R2))) → R1		9		4	
					11	1	(R1) ∨ ((M + (R2))) → (M + (R2))		10		5	

Logical OR	U	ORR	1	0101	00	0	(R1) ∨ (R2) ⟶ R1		4	1	1	R1 = 1111: system mode
register/register					01	0	(R1) ∨ ((R2)) → R1		5		2	
					01	1	(R1) ∨ ((R2)) → (R2)		6		3	
Logical OR	UK	ORK	0	0101	-	-	$(R3)_{8-15}$ ∨ K ⟶ $R3_{8-15}$		4	1	1	short; R1 = 1111: system
with constant												mode
Logical OR	U	ORKL	1	0101	01	0	(R1) ∨ KL ⟶ R1		5		2	long; R1 = 1111: system
with constant												mode
Multiple load	-	ML	1	0111	10	0	(M) ... (M + n) ⟶ A1 ... An		-		*	*3+(n-1)1 bits 5-8: n
					10	0	(M + (R2)) ... (M + (R2) + n) ⟶ A1 ... An		-		*	
					11	0	((M)) ... ((M) + n) → A1 ... An		-		**	**4+(n-1)1
					11	0	((M + (R2))) ... ((M + (R2)) + n) → A1 .. An		-		**	n = 1111: system mode
Multiple load/constant	-	MLK	1	0111	01	0	KL1, KL2, ... KLn → A1, A2, ... An		-		n+1	n = 1111: system mode
Multiple load/register	-	MLR	1	0111	01	0	((R2)) → A1; ((R2) + 2) → A2; ..... ;		-		n+1	n = 1111: system mode
							((R2) + 2n-2) ⟶ An	1)				bits 5-8; n.
						-	(A15) + 2n ⟶ A15; ((A15)) → A1;		-	1	n+1	system mode;
							((A15) - 2) ⟶ A2; .... ;					bits 5-8: n
							((A15) - 2n + 2) → An					
Multiple store	-	MS	1	0111	10	1	A1 ... An → M ... M + n		-	n	2+n	n = 1111: system mode;
					10	1	A1 ... An → M + (R2) ... M + (R2) + n	3)	-	n	2+n	bits 5-8: n
					11	1	A1 ... An → (M) ... (M) + n		-	n	2+n	
					11	1	A1 ... An → (M + (R2)) ... (M + (R2)) + n		-	n	2+n	

name (in alphabetical order)	mnemonic P850M	mnemonic P855M	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) P850M m.c.	P855M l.c.	P855M m.c.	remarks
Multiple store/register	-	MSR	1	0111	01	1	(A1) → (R2) ; (A2) → (R2) + 2 ;		-	n	n+1	n = 1111: system mode;
							....; (An) → (R2) + 2n - 2					bits 5-8: n
							(A1) → (A15) ; (A2) → (A15) - 2n + 2		-	n	n+1	pointer 128_{10}: stack overfl.
							.... ; (An, → (A15) - 2n + 2 ;					bits 5-8: n;
							(A15) - n → (A15)					system mode
Multiply	-	MU	1	1000	10	0	(A2) × (M) → A1, A2		-	8.5	3	
					10	0	(A2) × (M + (R2)) → A1, A2		-	8.5	3	
					11	0	(A2) × ((M)) → A1, A2		-	8.5	4	
					11	0	(A2) × ((M + (R2))) → A1, A2	2)	-	8.5	4	
Multiply registers/registers	-	MUR	1	1000	00	0	(A2) × (R2) → A1, A2		-	9.5	1	
					01	0	(A2) × ((R2)) → A1, A2		-	8.5	2	
Multiply with constant	-	MUK	1	1000	01	0	(A2) × KL → A1, A2		-	8.5	2	
One's complement	C1	C1	1	1111	10	0	$(\overline{M})$ → R1		7		3	R1 = 1111: system mode
					10	1	$(\overline{M})$ → M		8		4	
					10	0	$\overline{(M + (R2))}$ → R1	1)	7		3	
					10	1	$\overline{(M + (R2))}$ → M + (R2)		8		4	
					11	0	$\overline{((M))}$ → R1		9		4	
					11	1	$\overline{((M))}$ → (M)		10		5	
					11	0	$\overline{((M + (R2)))}$ → R1		9		4	
					11	1	$\overline{((M + (R2)))}$ → (M + (R2))		10		5	
One's Complement register/register	C1	C1R	1	1111	00	n.s.	$\overline{(R2)}$ → R1		4	1	1	when l/s bit = 0,
					01	0	$\overline{((R2))}$ → R1		5		2	R1 must be ≠ 0;
					01	1	$\overline{((R2))}$ → (R2)		6		3	R1 = 1111: system mode
Output from register	OT	OTR	0	1000	-	-	word/character from R3 → device	5)	4		3	bit 8 = 0; system mode
Read channel address	-	RCA	0	1001	-	-	channel number causing MIDB interrupt		-		3	bits 8-15: 01000000;
							request → R3					system mode
Read interrupt lines	RL	RIL	0	1001	-	-	State interrupt lines → R3		4	2	1	bits 8-15: 00000000;
												system mode
Relative backwards conditional branch	RB	RB	0	1011	-	0	(P) → displ. → P (branch effective)		4	1	1	bits 5-7: condition
							(P) → P (no branch)		3	1	1	bits 8-14: displ.;
								3)				bit 15: n.s.
Relative forward conditional branch	RF	RF	0	1010	-	0	(P) + displ. → P (branch effective)		4	1	1	bits 5-7: condition
						-	(P) → P (no branch)		3	1	1	bits 8-14: displ.;
												bit 15: n.s.
Reset internal interrupt	RI 7)	RIT	0	0100	-	1	to clear internal interrupt bits		4	1	1	bits 8, 9, 15: 1,
												bits 10-14: D.A.; system
												mode
Return from Function	RT	RTN	1	1110	01	0	(R2) + 4 → R2 ; ((R2)) → P ;	*	9	1	3	*reloaded from stack;
							((R2) -2) → PSW					R2 = 1111: system mode
Send status	SS 7)	SST	0	1001	-	-	status character/word from device → R3	5)	4		3	bits 8-9:11; system mode

name (in alphabetical order)	mnemonic P850M	mnemonic P855M	format	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) P850M m.c.	P855M l.c.	P855M m.c.	remarks
Set mode	-	SMD	0	0101	-	-	system mode → user mode	3)	-	1	1	bits 8-15: 00000001
Single left and normalize shift	-	SLN	0	0111	-	-			-	$(1.5+\frac{n}{2})$	1	bits 8-10: 100; bits 11-14: R2; bit 15: n.s.
Single left arithmetic shift	-	SLA	0	0111	-	-		2)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 000
Single left circular shift	-	SLC	0	0111	-	-			-		*	bits 8-10: 110 *$(2+\frac{1}{2}n+\frac{1}{2})$
Single left logical shift	-	SLL	0	0111	-	-		1)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 010
Single right and normalize shift	-	SRN	0	0111	-	-		3)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 101 bits 11-14: R2; bit 15: n.s.
Single right arithmetic shift	-	SRA	0	0111	-	-			-	$(1.5+\frac{n}{2})$	1	bits 8-10: 001
Single right circular shift	-	SRC	0	0111	-	-		1)	-	$(1.5+\frac{n}{2})$	1	bits 8-10: 111
Single right logical shift	-	SRL	0	0111	-	-			-	$(1.5+\frac{n}{2})$	1	bits 8-10: 011

Store character	TC	SC	1	1100	10	1	(R1)r → (M) r/l		8		4	R1 = 1111: system mode
					10	1	(R1)r → (M + (R2)) r/l		8		4	R1 must be ≠ 0
					11	1	(R1)r → ((M)) r/l		10		5	
					11	1	(R1)r → ((M + (R2))) r/l		10		5	
Store character/register	TC	SCR	1	1100	01	1	(R1)r → (R2) r/l		6		3	R1 = 1111: system mode
Store register	TW	ST	1	0000	10	1	(R1) → M	3)	8	1	3	R1 = 1111: system mode
					10	1	(R1) → M + (R2)		8	1	3	
					11	1	(R1) → ((M))		10	1	4	
					11	1	(R1) → ((M + (R2)))		10	1	4	
Store register/register	TW	STR	1	0000	01	1	(R1) → (R2)		6	1	2	R1 = 1111: system mode
					01	1	(R1) → (A15) ; (A15) - 2 → A15		6	1	2	if pointer 128_{10}: stack overflow
Subtract constant	SK	SUK	0	0011	-	-	(R3) - K → R3		4	1	1	short; R1 = 1111: system mode
Subtract constant	SU	SUKL	1	0011	01	0	(R1) - KL → R1		5		2	long; R1 = 1111: system mode
Subtract register/register	SU	SUR	1	0011	00	n.s.	(R1) - (R2) → R1	2)	4	1	1	when l/s bit = 1, R1 must
					01	0	(R1) - ((R2)) → R1		5		2	be ≠ 0; R1 = 1111:
					01	1	(R1) - ((R2)) → (R2)		6		3	system mode
Subtract word	SU	SU	1	0011	10	0	(R1) - (M) → R1		7		3	R1 = 1111: system mode
					10	1	(R1) - (M) → M		8		4	when l/s bit = 1,
					10	0	(R1) - (M + (R2)) → R1		7		3	R3 must be ≠ 0

name (in alphabetical order)	mnemonic P850M	mnemonic P855M	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) P850M m.c.	execution time in cycles 6) P855M l.c.	execution time in cycles 6) P855M m.c.	remarks
					10	1	(R1) - (M + (R2)) → M + (R2)		8		4	
					11	0	(R1)-((M)) → R1		9		4	
					11	1	(R1) - ((M)) → (M)		10		5	
					11	0	(R1) - ((M + (R2))) → R1		9		4	
					11	1	(R1 - ((M + (R2))) → (M + (R2))		10		5	
Test mask	I	TM	1	0100	00	1	[(R1) ∧ (R2)] ÷ 0 → CR	1)	4	1	1	R1 = 1111: system mode
Test not mask	U	TNM	1	0110	00	1	[(R1) ∀ (R2)] ÷ 0 → CR		4	1	1	R1 = 1111: system mode
Test status	TS 7)	TST	0	1001	-	-	test DCU 'ready state'	5)	4		3	bits 8-9: 10: system mode
Two's complement	C2	C2	1	0011	10	1	0 - (M) → M		8		4	bits 5-8: 0000
					10	1	0 - (M + (R2)) → M + (R2)	2)	8		4	
					11	1	0 - ((M)) → (M)		10		5	
					11	1	0 - ((M + (R2))) → (M + (R2))		10		5	
Two's complement/register	CR	C2R	1	0011	01	1	0 - ((R2)) → (R2)		6		3	bits 5-8: 0000
Write interrupt mask	W 7)	WIM	0	1000	-	-	(R3) → mask register		4	2	1	bits 8-15: 00000000
								3)				system mode
Write mask protection	-	WMP	0	1000	-	-	(R3) → M.P. key register		-	2	1	bits 8-15: 01000000;
												system mode
Write mask protection no. 2		WM2	0	1000	-	-	(R3) → M.P. key register		-	2	1	bits 8-15: 11000000; s.m.

NOTES FOR INSTRUCTION SET

Condition register

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 uncharged

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) Execution times in microseconds

	P850M	P855M
mc (memory cycle)	1.6	0.84
lc (logic cycle)	-	0.72

Mnemonics are not recognized by the Assembler.

P850M MEMORY MODULES

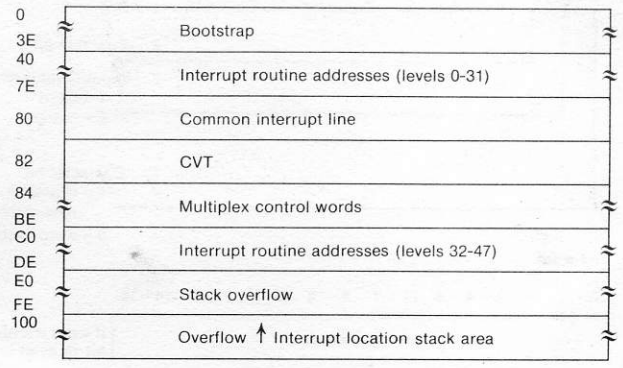
0000-03FE: MM0
0400-07FE: MM1
0800-0BFE: MM2
0C00-0FFE: MM3

P855M MEMORY PROTECTION

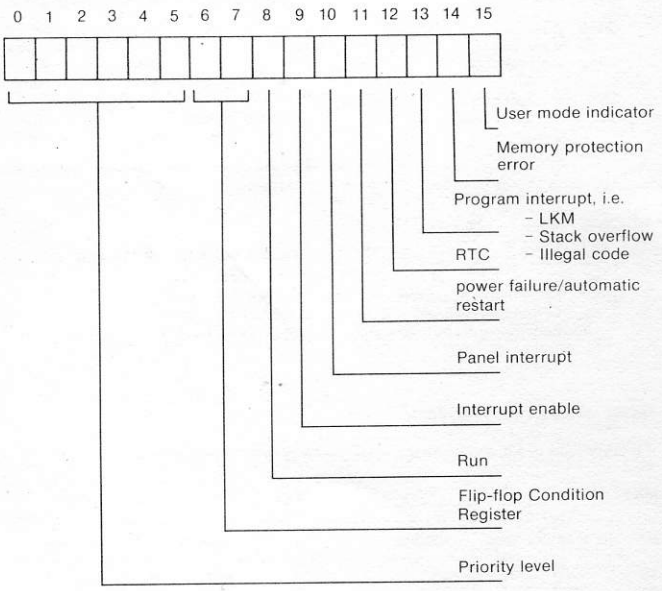
If a certain bit is set, the corresponding memory area is protected.

MK1 bit	memory area	module	MK2 bit	memory area	module
15	0000 - 07FE	MM0	15	8000 - 87FE	MM4
14	0800 - 0FFE		14	8800 - 8FFE	
13	1000 - 17FE		13	9000 - 97FE	
12	1800 - 1FFE		12	9800 - 9FFE	
11	2000 - 27FE	MM1	11	A000 - A7FE	MM5
10	2800 - 2FFE		10	A800 - AFFE	
9	3000 - 37FE		9	B000 - B7FE	
8	3800 - 3FFE		8	B800 - BFFE	
7	4000 - 47FE	MM2	7	C000 - C7FE	MM6
6	4800 - 4FFE		6	C800 - CFFE	
5	5000 - 57FE		5	D000 - D7FE	
4	5800 - 5FFE		4	D800 - DFFE	
3	6000 - 67FE	MM3	3	E000 - E7FE	MM7
2	6800 - 6FFE		2	E800 - EFFE	
1	7000 - 77FE		1	F000 - F7FE	
0	7800 - 7FFE		0	F800 - FFFE	

DEDICATED MEMORY LOCATIONS

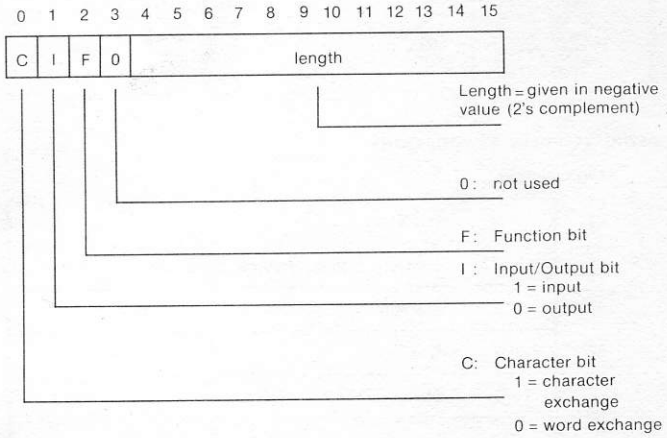


PROGRAM STATUS WORD P850M/P855M

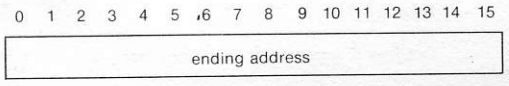


MULTIPLEX CONTROL WORDS

1st word



2nd word



MEMORY CLEAR PROGRAM

This program puts the contents of A2 in the memory location X'40' upto the address given in A3.

address	code	mnemonic P850M		mnemonic P855M	
0	822D	TW	A2,RA,A3,S	STR	A2, A3
2	1B02	SK	A3,2	SUK	A3,2
4	5C06	RB	4,6	RB(4)	⋆-4
6	207F	HT		HLT	

This program puts the contents of A1 in the memory locations /4C upto the address given in A3.

P850M

40	024C	LK	A2,/4C
42	8129	TW	A1,RA,A2,S
44	1202	AK	A2,2
46	EA0C	CW	A2,RR,A3
48	5A08	RB	2,8
4A	207F	DATA	/207F
4C			

P855M

	LDK	A2,/4C
LAB	STR	A1,A2
	ADK	A2,2
	CWR	A2,A3
	RB(2)	LAB
	HLT	

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

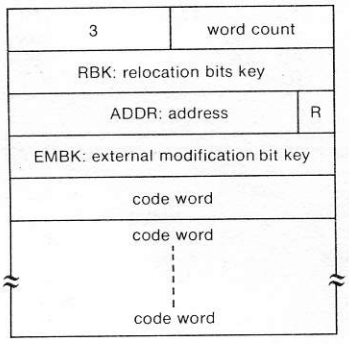
2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

P850M/P855M OBJECT CODE FORMAT

Program identification

IDENT – program name – line feed – X off – carriage return.

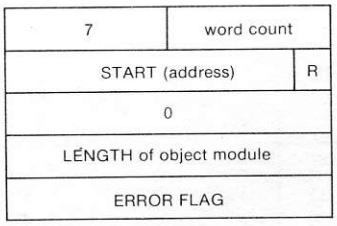
Code cluster



= 0: address is absolute
= 1: address is relative

P855M only

END/START cluster



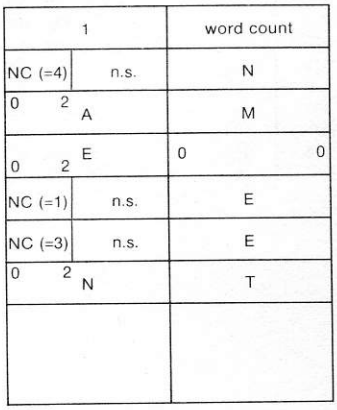
= 0: start address absolute
= 1: start address relative

even number of characters

indicates number of errors in binary

P855M OBJECT CODE FORMAT

Entry point names cluster

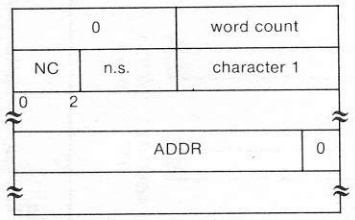


NC = number of characters of entry point name
n.s. = not significant

External reference names cluster

Same format as 'Entry point names cluster'.

BLOCK data cluster

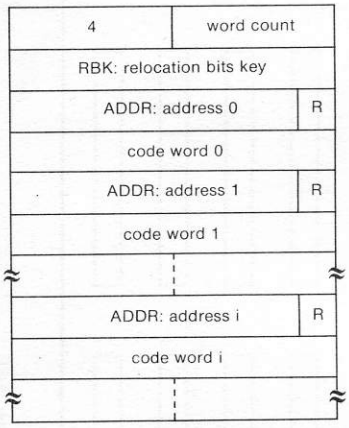


name of labelled COMMON block (up to 6 characters)

absolute data words to be loaded, starting at ADDR in named COMMON block

NC = number of characters of the symbol (3 bits)
n.s. = not significant.

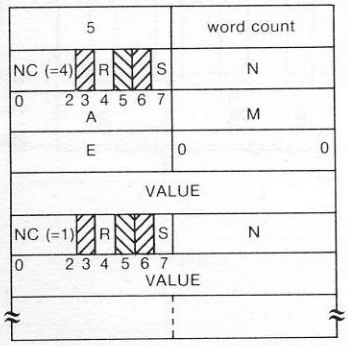
Internal modification cluster



= 0: address absolute
= 1: address relative

maximum number of code words: 16

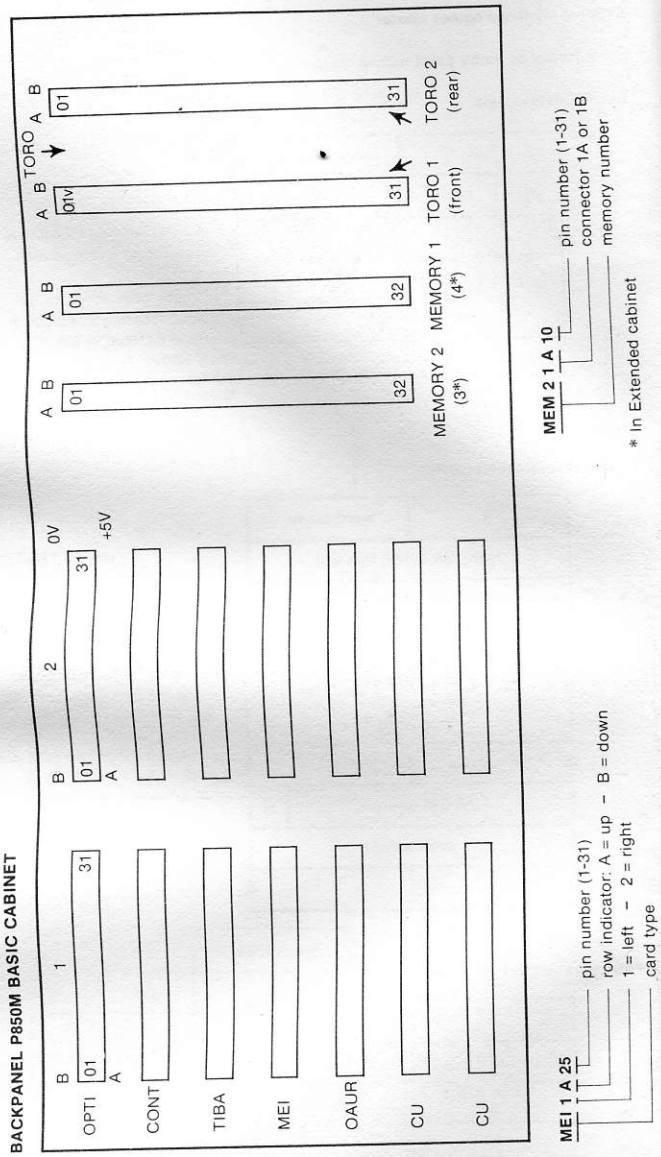
Entry point definition cluster



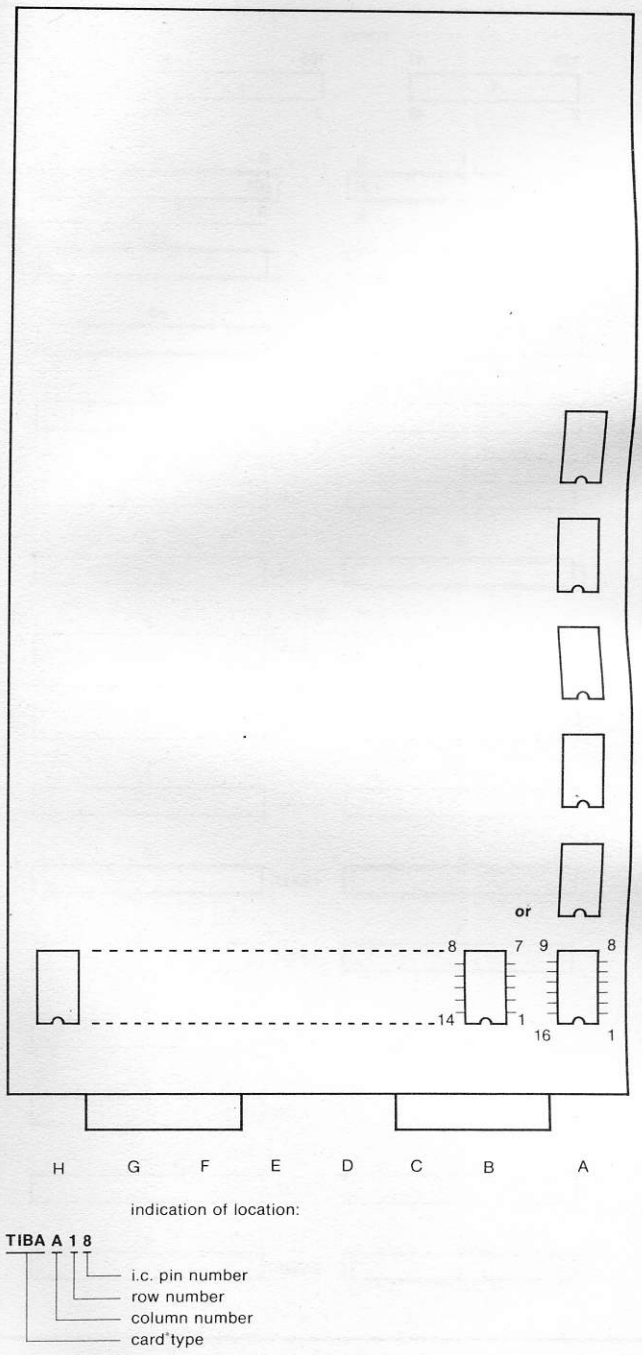
NC = number of characters of entry point name
R = 0: absolute entry point
= 1: relative entry point
S = 1 if entry point name = internal symbol table name.

COMMON length definition cluster

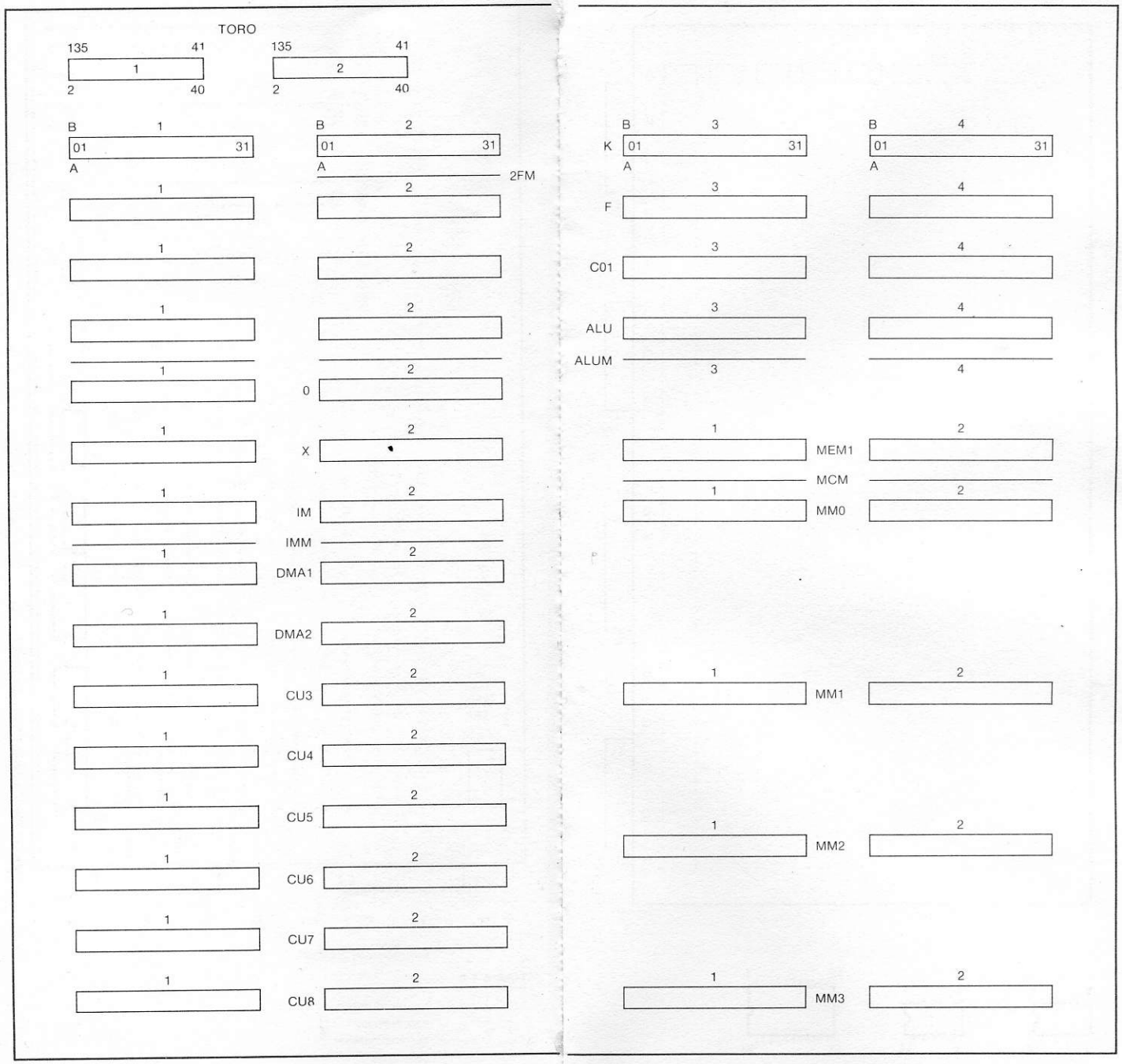
Same format as 'Entry point definition cluster'.



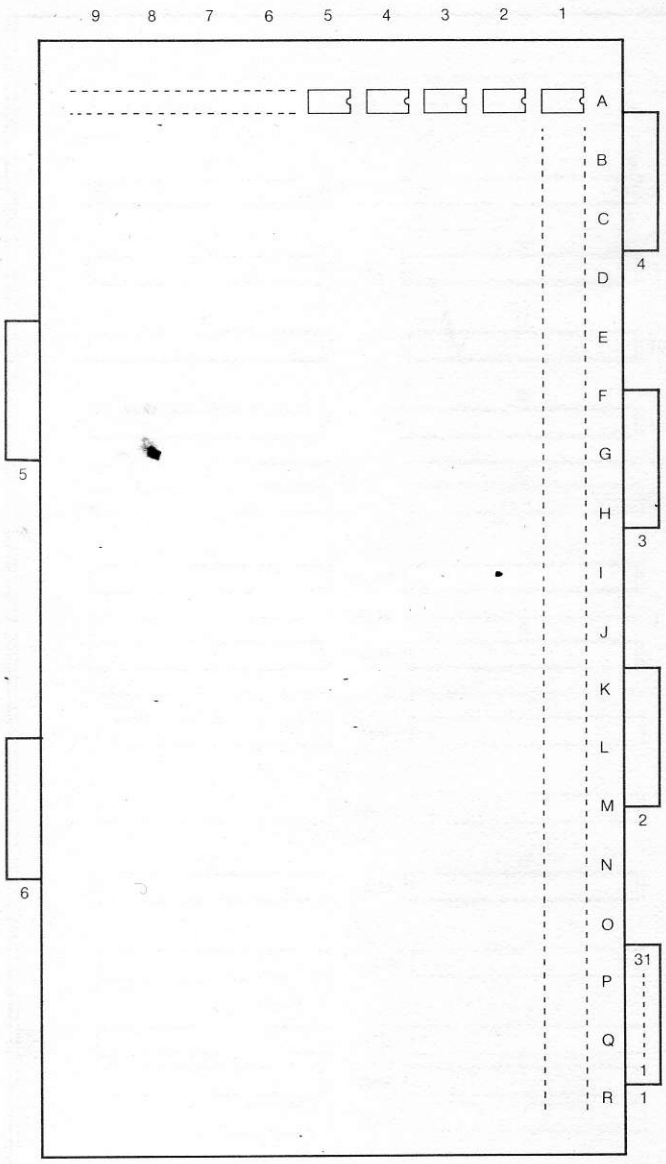
P850M CPU CARD AND P850M/P855M CU CARD



BACKPANEL P855M BASIC CABINET (8 UNITS)



DOUBLE CARDS (P855M CPU)



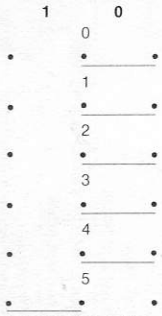
CONTROL UNITS

Status flip-flops

	F0	F1
INCT	0	0
EX	0	1
EXCH	1	1
WST	1	0

CHANGING THE DEVICE ADDRESS

The device address on a C.U. card may be altered by means of jumpers.



Example for device address 0001.

C.U. STATUS WORD CONFIGURATION

Bit	Description	CU									
		ASR	CR	MHD	LP	TP	PTR	CASS Tape	PLOT	FHD	MT
0	-										
1	ready			x				x			x
2	rewind										x
3	tape mark has been read										x
4	EOT							x			
5	on cylinder load point			x							x
6	seek error			x							
	write unable							x			x
7	A or B side							x			
8	Device Address							x		x	
9	Device Address			x				x		x	
10	EOT						x	x			
	tape low					x					
11	Program error			x				x	x	x	x
12	Incorrect length		x	x				x		x	x
	Y limit overpass								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	x

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

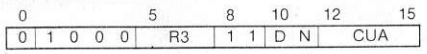
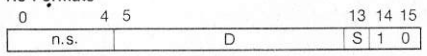
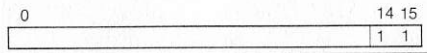
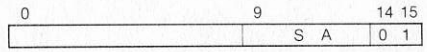
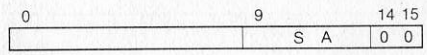
DEVICES AND RECOGNIZED COMMANDS

PTR	CR	PLOT PTP	ASR	LP	Recognised Command	Format	Meaning
x			x	x	TST	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 0 0 1 | R3 | 1 0 | D A	Test status DA = Device Address R3 = register into which status is loaded.
x	x	x	x	x	CIO start (read a card for CR)	0 1 0 0 0 | R3 | 1 1 | D A 15 Register indicated in R3	R3 only significant for ASR bit 15 = 1 start input bit 15 = 0 start output
x	x	x	x	x	CIO stop	0 1 0 0 0 | R3 | 1 0 | D A	Stop input R3 not significant
x	x		x		INR	0 1 0 0 1 | R3 | 0 X | D A	Input to register indicated in R3 field. X: not significant
x	x	x	x	x	SST	0 1 0 0 1 | R3 | 1 1 | D A	Send status The status word of the control unit is sent to the register indicated in R3 field.
		x	x	x	OTR	0 1 0 0 0 | R3 | 0 X | D A	Output from the register indicated in R3 field to CU
						R3 Formats 8 9 10 11 12 13 14 15	
				x		1 1 X 0 | Chan. Nr.	Advance till given channel
						8 9 10 11 12 13 14 15	
				x		1 1 X 1 | Nr. of lines	Skip the given number of lines
				x		0 0 0 0 1 0 1 0	Paper feed and carriage return
				x		0 0 0 0 1 1 0 0	Form feed and carriage return
				x		0 0 0 0 1 1 0 1	Carriage return

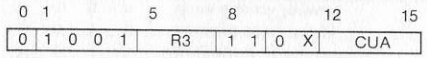
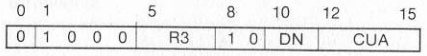
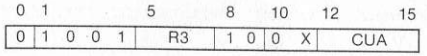
DIOS

Recognised Commands	Format	Meaning
OTR	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 5 8 10 11 12 15 0 1 0 0 0 | R3 | 0 0 | W A | CUA	Output data word R3 = register from which data is read CUA = CU address WA = Word address 00 = word A 01 = word B 10 = word C 11 = word D
INR	0 1 5 8 10 11 12 15 0 1 0 0 1 | R3 | 0 0 | W A | CUA	Input data word R3 = register into which data is loaded CUA = CU address WA = word address (see above)
INR	0 1 5 8 10 11 12 15 0 1 0 0 1 | R3 | 0 1 | W A | CUA	Input word address R3 = register into which the address is loaded CUA = CU address WA = not significant
	Format of R3 0 12 15 A B C D	A one-bit in field 12-15 indicates that the corresponding word is active.

MOVING HEAD DISC CU

Recognised Commands	Format 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	
CIO start general		Start an Operation DN = Device number 00 or 01 CUA = Control Unit address R3 = specifies the command
Seek command	**R3 Formats** 	move the head to another cylinder D = difference between cylinder number S = direction of motion 1 = forward 0 = reverse
Seek to zero command		move heads to cylinder 00
Write a sector		Start a write operation on the sector address on the BOU lines on the cylinder of the moment. SA = Sector Address 0-31
Read a sector		Start a read operation on the sector which is addressed on the BOU lines on the cylinder of the moment SA = Sector Address 0-31

MOVING HEAD DISC CU (CONTINUED)

Recognised Commands	Format 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Meaning
SST		Exchange status between CU and CPU CUA = CU address X = not significant R3 = specifies register into which status is loaded (see status word).
CIO Stop		Stop data transfer R3 = not significant DN = Device number CUA = CU address
TST		Test status of CU CUA = CU Address R3 = register into which status is loaded.

CASSETTE TAPE CU

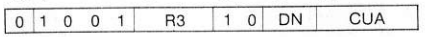
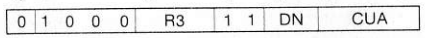
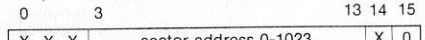
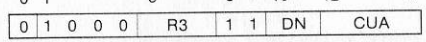
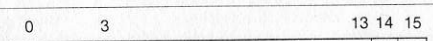
Recognised Commands	Format 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Meaning
TST	0 1 | 5 | 8 | 10 | 12 | 15 | | | R3 | 1 0 | D N | CUA	Test status CUA = CU address DN = Device number (00-11) R3 = register into which status is loaded
CIO Start	0 1 | 5 | 8 | 10 | 12 | 15 0 1 0 0 0 | R3 | 1 1 | D N | CUA R3 contents	Start command CUA = CU address DN = Device number R3 = register specifying the command.

bit number 12 13 14 15	Command
0 0 0 0	Lock/unlock
0 0 0 1	Erase forward
0 0 1 0	Backwards space block
0 0 1 1	Forward space block
0 1 0 1	Write a block forward
0 1 1 1	Read a block forward
1 0 0 0	Rewind at fast speed
1 0 1 0	Search tape mark backwards
1 0 1 1	Search tape mark forward

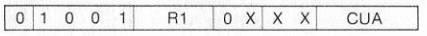
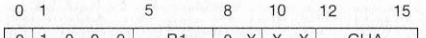
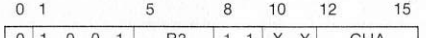
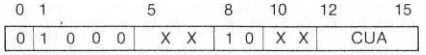
CASSETTE TAPE CU (CONTINUED)

Recognised Commands	Format 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Meaning
CIO Stop	0 1 | 5 | 8 | 10 | 12 | 15 0 1 0 0 0 | R3 | 1 0 | X X | CUA	Stop transfer CUA = CU address DN = not significant R3 = register specifying the command X = not significant
INR	0 1 | 5 | 8 | 10 | 12 | 15 0 1 0 0 1 | R3 | 0 X | 0 X | CUA	Input to register DN = not significant R3 = register into which data is loaded X = not significant
OTR	0 1 | 5 | 8 | 10 | 12 | 15 0 1 0 0 0 | R3 | 0 X | 0 X | CUA	Output from register DN = not significant X = not significant R3 = register from which the data is read.
SST	0 1 | 5 | 8 | 10 | 12 | 15 0 1 0 0 1 | R3 | 1 1 | 0 X | CUA	Send Status See status word configuration on page 37 For explanation of field, see above

FIXED HEAD DISC CU

Recognised Commands	Format 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Meaning
TST		Test status of CU DN = not significant CUA = CU address R3 = register into which status is loaded.
Write one sector		Start a write operation on the sector on the BOU-lines DN = Device number 00, 01, 10 or 11 CUA = CU address R3 = register specifying the command
		R3 format
Read one sector		Start a write operation on the sector on the BOU-lines. DN = Device number 00, 01, 10 or 11 CUA = CU address R3 = register specifying the command.
		R3 format

FIXED HEAD DISC CU (CONTINUED)

Recognised Commands	Format 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Meaning
INR		Input data word X = not significant CUA = CU address R1 = register into which the word is loaded.
OTR		Output data word X = not significant CUA =CU address R1 = register from which the word is read.
SST		Exchange status between CU and CPU X = not significant CUA = CU address R3 = register into which status word is loaded (see status word).
CIO Stop		Stop data transfer X =not significant CUA = CU address

MAGNETIC TAPE CU

TST

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	1	R3			1	0	DN		CUA			

Test status
CUA = CU address
DN = Device number
R3 = register into which status word is loaded

CIO Start

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	R3			1	1	DN		CUA			

R3 contents

bit number 10	11	12	13	14	15	Command
1	X	X	0	X	X	Off-line
1	X	X	1	X	X	Rewind
0	0	0	0	1	1	Write
0	0	1	0	1	1	Write-edit
0	1	0	0	0	1	Write file mark
0	1	1	0	0	1	Erase gap
0	0	0	0	1	0	Read
0	1	0	0	0	0	Search tape mark forward
0	1	0	1	0	0	Search tape mark backwards
0	0	0	0	0	0	Forward space block
0	0	0	1	0	0	Backwards space block
0	0	1	1	0	0	Backwards space block edit

Start command
CUA = CU address
DN = Device number 00, 01, 10 or 11
R3 = register specifying the command.

MAGNETIC TAPE CU (CONTINUED)

CIO stop

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	X	X	X	1	0	DN		CUA			

Stop transfer
CUA = CU address
DN = Device number 00, 01, 10 or 11
X = not significant

SST

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	1	R3			1	1	X	X	CUA			

Send status
CUA = CU address
R3 = register into which status word is loaded
X = not significant

DMA-channel

Recognised Commands	Format 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Meaning
CIO Start	0 1 5 8 10 15 0 1 0 0 0 X X 1 1 DA 1)	Initialize a DMAC exchange DA = DMAC address
OTR write length	0 1 5 8 10 15 0 1 0 0 0 R1 0 0 DA	Load DMAC length register DA = DMAC address R1 = register from which length and functions are read.
	0 1 2 3 length format 15 X I F 2's complement of length 1)	I = input bit F = function bit Length, expressed in characters, must be an even number.
OTR write address	0 1 5 8 10 15 0 1 0 0 0 R1 0 1 DA	DA = DMAC address R1 = register from which address is read
	0 14 15 starting address X1)	address format
CIO Stop	0 1 5 8 15 0 1 0 0 0 X X X 1 0 DA 1)	De-activate DMAC DA = DMAC address
INR read length	0 1 5 8 10 15 0 1 0 0 1 R1 0 X DA 1)	Transfer DMAC length into R1 DA = DMAC address R1 = register into which DMAC length is loaded

1) X = not significant

P850M OPTION CARD (OPTI) STANDARD INTERRUPT CABLING

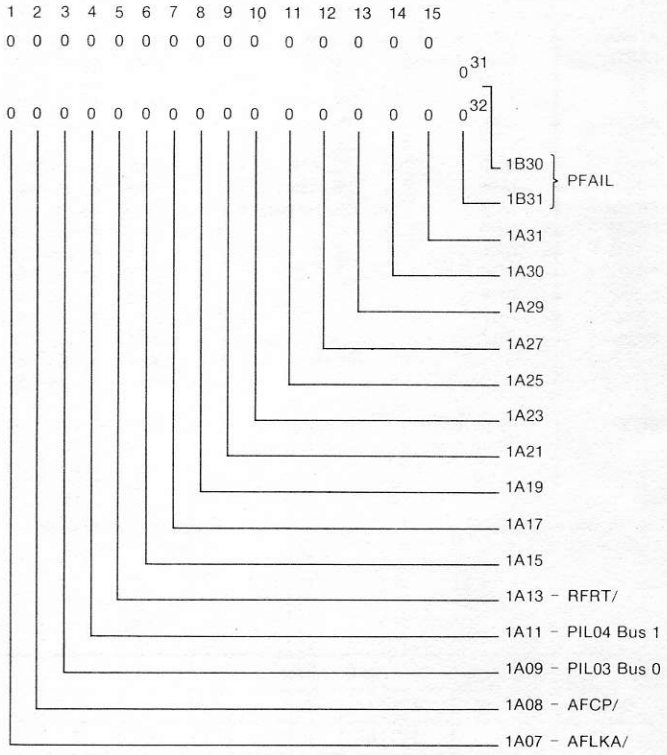
Option Card S 307A (5111 199 93920)

BAD - LINES

2	3	4	5	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
0	0	0	0	CPU	ASR1	PTR1	PTP1
0	0	0	1	FHD	FHD	FHD	FHD
0	0	1	0	MHD	MHD	PTR2	PTP2
0	0	1	1	MHD	MHD	MHD	MHD
0	1	0	0	MT	MT	MT	MT
0	1	0	1	CR	ASR2	*	*
0	1	1	0	CT	CT	CT	CT
0	1	1	1	*	*	*	*
1	0	0	0	*	*	*	*
1	0	0	1	*	*	*	*
1	0	1	0	DIOC	*	*	*
1	0	1	1	*	*	*	*
1	1	0	0	*	*	*	*
1	1	0	1	LP	*	*	*
1	1	1	0	PLT	*	*	*
1	1	1	1	ADAP	DMA	MIDB	RTC

* Not allocated

P850M OPTION CARD (OPTI) STANDARD INTERRUPT CABLING (up to P850M-26)



P855M PIN BOARD CABLING

Interrupt mask bits	Pin location	Dedicated memory address	Standard connection
BMS KIL 00/	F 1A11		reserved
BMS KIL 01/	F 1A14		plotter
BMS KIL 02/	F 1A16		line printer
BMS KIL 03/	F 1B16		MHD2
BMS KIL 04/	F 1B25		MHD1
BMS KIL 05/	F 1B17		FHD
BMS KIL 06/	F 1B18		PEC tape
BMS KIL 07/	F 1B22		ASR2
BMS KIL 08/	F 1A23		ASR1
BMS KIL 09/	F 1B28		PTP2
BMS KIL 10/	F 1A19		PTP1
BMS KIL 11/	F 1A20		Cass. tape
BMS KIL 12/	F 2B10		PTR2
BMS KIL 13/	F 1A28		PTR1
BMS KIL 14/	F 1A31		CR
BMS KIL 15/	F 2B09		reserved

P855M

Standard interrupt signals	Pin location	Dedicated memory address	Standard connection
CFPF/	IM 1A16		
CFEM/	IM 1A17		
CFRT/	IM 2A27		
CFPI/	F 3B31		
CFCP/	F 4B04		
CFMIDB/	X 1A06		
PIL	CU 1A15		
CISMSK/	F 2A02		

Standard interrupt levels	Pin location	Dedicated memory address	Standard connection
BIS 00/	F 2B12	/0040	PFAURE
BIS 01/	F 2B11	/0042	LKM
BIS 02/	F 2A12	/0044	RTC
BIS 03/	F 2A11	/0046	MIDB
BIS 04/	F 2A10	/0048	reserved
BIS 05/	F 2A09	/004A	common int. line
BIS 06/	F 2A08	/004C	memory protect
BIS 07/	F 2A07	/004E	CP

Optional interrupt levels	Pin location	Address on header connector	Dedicated memory address	Standard connection
JIS 08/	IM 1A02	6A07	/0050	ASR
JIS 09/	IM 1A03	6A08	/0052	disc
JIS 10/	IM 1A04	6A09	/0054	line printer
JIS 11/	IM 1A05	6A10	/0056	card reader
JIS 12/	IM 1A06	6A11	/0058	magtape
JIS 13/	IM 1A07	6A12	/005A	
JIS 14/	IM 1A08	6A13	/005C	
JIS 15/	IM 1A09	6A14	/005E	
JIS 16/	IM 1A10	6A15	/0060	
JIS 17/	IM 1A11	6A16	/0062	

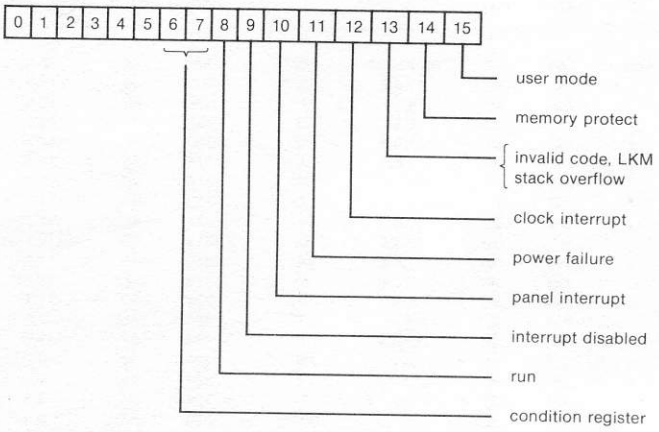
Optional interrupt levels	Pin location	Dedicated memory address	Connected to
KIS 18/	IM 2A05	/0064	
KIS 19/	IM 2A04	/0066	
KIS 20/	IM 2A03	/0068	
KIS 21/	IM 2A02	/006A	
KIS 22/	IM 2A01	/006C	
KIS 23/	IM 2B01	/006E	
KIS 24/	IM 2B02	/0070	
KIS 25/	IM 2B03	/0072	
KIS 26/	IM 2B04	/0074	
KIS 27/	IM 2B05	/0076	
KIS 28C/	IM 5A23	/0078	
KIS 29C/	IM 5A22	/007A	
KIS 30C/	IM 5A21	/007C	
KIS 31C/	IM 5A20	/007E	
LIS 32C/	IM 5A19	/00C0	
LIS 33C/	IM 5A18	/00C2	
LIS 34C/	IM 5A17	/00C4	
LIS 35C/	IM 5A16	/00C6	
LIS 36C/	IM 5A15	/00C8	
LIS 37C/	IM 5A14	/00CA	
LIS 38C/	IM 5A13	/00CC	
LIS 39C/	IM 5A12	/00CE	
LIS 40C/	IM 5A11	/00D0	
LIS 41C/	IM 5A10	/00D2	
LIS 42C/	IM 5A09	/00D4	
LIS 43C/	IM 5A08	/00D6	
LIS 44C/	IM 5A07	/00D8	
LIS 45C/	IM 5A06	/00DA	
LIS 46C/	IM 5A05	/00DC	
LIS 47C/	IM 5A04	/00DE	

P855M BREAK LEVELS

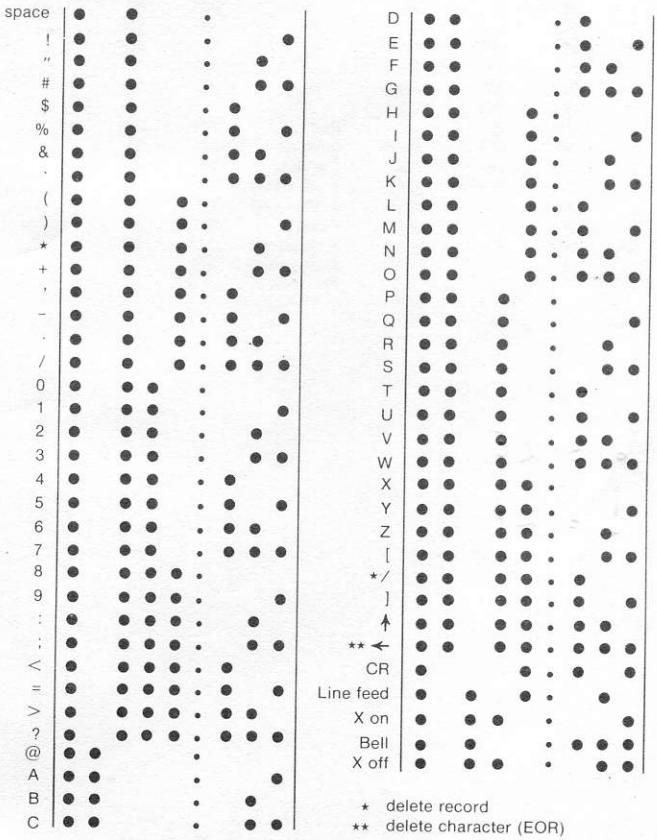
Level number	Pin location	Address on header connector E	Dedicated Multiplex address	Connected to
XBS 01	X 1A02	X 5A11	/0084 - /0086	
XBS 02	X 1B02	X 5A12	/0088 - /008A	
XBS 03	X 1B03	X 5A13	/008C - /008E	
XBS 04	X 1B04	X 5A15	/0090 - /0092	
WBS 05	X 1B05	X 5A17	/0094 - /0096	
WBS 06	X 1B06	X 5A14	/0098 - /009A	
WBS 07	X 1B07	X 5A16	/009C - /009E	
WBS 08	X 2B26	X 5A03	/00A0 - /00A2	
WBS 09	X 2B27	X 5A04	/00A4 - /00A6	
WBS 10	X 2B28	X 5A05	/00A8 - /00AA	
WBS 11	X 2B29	X 5A06	/00AC - /00AE	
WBS 12	X 2B30	X 5A07	/00B0 - /00B2	
WBS 13	X 1A05	X 5A08	/00B4 - /00B6	
WBS 14	X 1A04	X 5A09	/00B8 - /00BA	
WBS 15	X 1A03	X 5A10	/00BC - /00BE	

Note
Break level = device address.
Break is generated on c.u. 1A16.

FORMAT OF CPU STATUS WORD



ASCII CODE



* delete record
** delete character (EOR)

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	

• = 2E

! = 21

November 1973

A publication of:

Philips-Electrologica B.V.
Main Marketing Group OEM
P.O.BOX 245 Apeldoorn, The Netherlands

Copyright © by Philips-Electrologica B.V.

5122 991 12351

Printed in The Netherlands

PHILIPS

P855M

System Software Reference Data

Data Systems

CONTENTS

DISC BOOTSTRAP

0000	20BF	BBC0	0032	911C	921C	3F42	220F	1778
0010	BB3D	0503	8420	001A	8F1C	050C	8420	0022
0020	8F1C	8340	003C	9B20	0198	812C	5C2E	1304
0030	8F0C	45C0	4DC0	5C04	8F10	4E66	3E00	0002

P8 BOOTSTRAP FOR PTR

0000	20BF	43E0	4A20	5C04	227F	5808	0100	0402
0010	0602	E508	4A20	5C04	1E01	590A	9508	1C01
0020	5006	520C	8714	5F18	8314	1502	8094	5F20
0030	B114	1302	852D	1F01	5C2A	8F02	0000	0002

P4 BOOTSTRAP FOR ASR

0000	20BF	42D0	4A10	5C04	227F	5808	8320	FFFC
0010	0604	4A10	5C04	220F	3D41	3D41	3D41	3D41
0020	9508	1E01	5C14	1302	5006	5108	8717	5F20
0030	8314	5F24	B114	852D	1F01	5C2C	4590	207F

Standard file codes - basic

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

Standard file codes - disc

/01 = operator's typewriter
/02 = listing output
/03 = punch output
/D0 = catalogued procedure input
/D3 = reserved for system use
/D4 = /S, or library source file (Update output)
/D5 = /0 (ASM output, LKE input)
/D6 = /L (LKE output)
/D7 = system object file (library)
/D8 = user object file (library)
/D9 - DA = 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

Standard device addresses

ASR1 = 10
PTR1 = 20
PTP1 = 30
FHD1 = 01
FHD2 = 11
FHD3 = 21
FHD4 = 31
} one CU (MX chan or DMAC)

MHD1 = 02
MHD2 = 12
} one CU (MX chan or DMAC)
MHD3 = 03
MHD4 = 13
} one CU (MX chan only)
PTR2 = 22
PTP2 = 32
MT1 = 04
MT2 = 14
MT3 = 24
MT4 = 34
} one CU (MX chan only)
CR = 05
ASR2 = 15
TC1 = 06
TC2 = 16
TC3 = 26
TC4 = 36
} one CU (MX chan only)
LP = 0D
PL = 0E
Adapter= 0F
DMAC = 1F
MIDB = 2F
RTC = 3F

MONITOR OPERATOR MESSAGES

BOM - DOM

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

BRTM

LD	Load a program
ST	Start a program
WM␣<address>␣<value1>[␣<value2>..<value n>]	Write into memory

Message	Meaning
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

CCI COMMANDS

syntax: ASG␣<file code1>,[<file code2>|<device name>][,<name>] [,<userid>[,<file code3>]]
Assign a file code to a peripheral unit, a disc file or a temporary area on disc.

syntax: BYE[␣BYE]
This command terminates the session or changes the operational mode

syntax: DCU␣<userid>,<disc number>
Declare a new user (system session only)

syntax: DEL␣[<name>|/OB][,[/S|/O|/L]]
Delete a file or object module from a library

syntax: DLU␣<userid>,<disc number>
Delete a user (system session only)

syntax: DUF␣[<file code|/O|/L|<name>][,<sect nb1>[,<sect nb2>]]
Dump file

syntax: INC␣[/OBJCT|<name>][,<userid>[,<disc file code>]]
Include an object module

syntax: JOB␣[<userid>|/FX,<userid>]
Define a new session

syntax: KPF␣[/S|/O|/L|<file code>][,<name>]
Keep a file

syntax: LIC␣<disc number>
List catalogue on operator's typewriter (system session only)

syntax: LSD␣[/OB]
List directory on operator's typewriter

syntax: LST␣[<file code>|/S|/S,<name>|<name>][,<line nb1>[,<line nb2>]]
List file

syntax: MES␣<message to operator>
Command especially used in batch processing. It allows the specified message to be sent to the operator

syntax: MOV␣<name>,[/S|/L|/<file code>][,<userid>[,<disc file code>]]
Move a file

syntax: PCH␣[<file code>|/S|<name>|<name>,/S]
Punch a file

syntax: PLB␣<file code>
Used to list the label of a tape on the operator's typewriter

syntax: PLD␣<name>[,L]
Punch a load file

syntax: POB␣[<name>]
Punch an object module

syntax: PRC␣<disc number>
Print catalogue on print unit (system session only)

syntax: PRD␣[/OB]
Print directory on print unit

syntax: PRT␣[<file code>|/S|/S,<name>|<name>][,<line nb1>[,<line nb2>]]
Print file

syntax: PSE␣<message to operator>
Used to send a message to the operator and go in Pause state. The operator has to give an RS command to continue

syntax: RDA␣<disc file code>[,/<input file code>]
Read data

syntax: RDO␣[<file code>]
Read object

syntax: RDS␣[<file code>]
Read source

syntax: REF␣<file code>
This command positions the file to the first record

syntax: RSU␣<disc number>
Replace supervisor (system session only)

syntax: RUN␣[<name>|<name>,0|0]
Run a program

syntax: SCR␣[[/S]/O|/L[<file code>]]
Scratch

syntax: SDM␣<disc file code>,<mag tape file code>
Save user file on magnetic tape

syntax: SEG␣<name list>
Define segments

syntax: SVD␣<disc number 1>,<disc number 2>
Save disc onto another disc (system session only)

syntax: SVU␣<userid>,<disc number>
Save user files

PROCESSOR CALLS

syntax: ASM␣[/S|<name>][,NL]
Call the Assembler

syntax: LKE␣[N|S|U][,M][,[DE|DS]][,<address>][,<start address>]
Call the Linkage Editor

syntax: UPD␣<name>[,<file code1>|/S[,<file code2>]]
Call the Update Package

syntax: EDT␣<name>
Call the Text Editor

syntax: DEB␣[<name>]
Call the Debugging Package

syntax: FOR␣[/S|<name>][,NL]
Call the Full FORTRAN Compiler

syntax: FRT␣[/S|<name>][,NL]
Call the Basic FORTRAN Compiler

syntax: TCD
Call the Full FORTRAN Transcoder

MAGNETIC TAPE AND CASSETTE TAPE COMMANDS

syntax: REW␣<file code>
Rewind until load point

syntax: ULD␣<file code>
Unlock device

syntax: FFS␣<file code>[,<number>|ALL]
File forward space

syntax: FBS␣<file code>[,<number>]
File backward space

syntax: RFS␣<file code>[,<number>]
Record forward space

syntax: RBS␣<file code>[,<number>]
Record backward space

syntax: WEF␣<file code>[,<number>]
Write End-Of-File

syntax: WES␣<file code>[,<number>]
Write End-Of-Segment

syntax: REF␣<file code>
This command positions the file to the first record

syntax: WLB␣<file code>,<serial number>,<security code>,<owner>
This command is used to write the VOL 1 label on magnetic tape or cassette tape

syntax: WEV␣<file code>
This command writes the End of Volume mark on the magnetic tape or cassette tape

SCL COMMANDS

LD␣<name>,<disc number>[,<level>|/CM,<level>[,SL,<number>]]
Load a memory-resident program

RO␣<name>,<disc number>[,SL,<number>]
Declare a Read-Only program

CN␣<name>,<level>
Connect a program to a software level

CT␣<name>,<NTIM>,<PR>,[<NC>|<HH>,<MM>,<SS>]
Connect a program to the clock or to a timer

DN␣<name>
Disconnect a program from a level

DT␣<name>
Disconnect a program from a clock or timer

SD␣<DD>,<MM>,<YY>
Set date

SC␣[<HH>[,<MM>[,<SS>]]]
Set clock

ST␣<name>[,<disc number>]
Start a program

KF␣<file code>,<file name>
Keep file

AS␣<file code1>,[<file code2>|DN[DA]|DKFX[,<file name>],<nr of granules>]]
Assign a file code

DL␣<file code>
Delete a file code

DF␣<file name>,<disc number>
Delete a file

HC Halt clock

EN End of SCL commands

DISC PROCESSOR MESSAGES

Update

!!DL␣<line number 1>[,<line number 2>]
Delete the line, or lines, specified

!!IL␣[<line number>]
Insert line(s) after the specified line number or after this statement if the parameter is not specified

!!EN End of Update

!!AB Abort the update and delete /S file

TERMINATE THE UPDATING WITH A KPF COMMAND

Debugging

AT␣<address>	Define a breakpoint
IF␣[<register>|<M␣<address>][<|=|>][/<value>|<register>|<M␣address>]	Conditional execution of breakpoint
DM<address1>[-<address2>]	Dump memory
DR␣[<number>][-<number>]	Dump register
WM␣<address>=<value1>[-<value2>-...<value n>]	Write memory
WR␣<number>=<value1>[-<value2>-...<value n>]	Write register
TR␣<2 characters>	Trace
GO␣<address>,	Restart program
CI	Change input
RE␣<file code>-<number>-<address>	Read from external device
DB␣<address>	Delete a breakpoint
RT	Return

BASIC PROCESSOR MESSAGES

Assembler

Option message:

[x x x x] [N] [R] [Q]

- object output must be in 4×4 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
- abs. start address of blank common
- program must be punched in 8+8
- program must be punched in 4×4
- name for load module
- object module 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 points (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 4×4
- 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 the 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 updating until next :EOF 2. After U: terminate processing.
LF CR	Concludes typed-in statement or message

Debugging Messages

AT␣<address>,	Define a breakpoint as specified
CI,	Read further control messages
DB␣<address>,	Delete breakpoint as specified
DM␣<address1>[-<address2>],	Dump memory as specified
DR␣[<number>][-<number>],	Dump registers as specified
GO␣<address>,	Start program at specified address
IF␣[<register>|M␣<address>][<|=|>][/<value>|<-register>|M␣<address>],	Conditional execution of breakpoint
RE␣<file code>-<number>-<address>,	Read from external device
RT,	Start execution of user program
TR␣<2 characters>,	Trace the execution of next breakpoint specified
WM␣<address>=<value1>[-<value2>-<value3>...<value n>],	Write into memory
WR␣<number>=<value1>[-<value2>-<value3>...<value n>]	Write into register

INSTRUCTIONS

In the instructions described below, each mnemonic (OPC) is followed by a letter (from a to g). This letter refers to the syntax description immediately following the instruction group heading (MEMORY REFERENCE INSTRUCTIONS etc.). Each instruction is followed by a number (from 1 to 15) referring to a condition register contents described hereafter.

1) CR = 0 if (<r1>) = 0
1 if (<r1>) > 0
2 if (<r1>) < 0

2) CR = unchanged

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

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

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

6) CR = 0 if (<r1>) = (2nd operand)
1 if (<r1>) > (2nd operand)
2 if (<r1>) < (2nd operand)

7) CR = 0 if (<r1>) = (2nd operand)l/r
1 if (<r1>) > (2nd operand)l/r
2 if (<r1>) < (2nd operand)l/r

8) CR = 0 if (A2) = 0
1 if (A2) > 0
2 if (A2) < 0
3 in case of overflow

9) CR = 0 if (<r1>)r = ((<r2>) l/r
1 if (<r1>)r > ((<r2>) l/r
2 if (<r1>)r < ((<r2>))l/r

10) Bits 0-5 of PSW loaded in PLR if <r2> = A15.
When reloaded from stack bits 6 and 7 of PSW are loaded in CR

11) CR = 0 if <lk> = 0
1 if <lk> > 0
2 if <lk> < 0

12) CR = 0 if (<r1>) = <lk>
1 if (<r1>) > <lk>
2 if (<r1>) < <lk>

13) CR = 0 if (<r1>)r = <lk>l
1 if (<r1>)r > <lk>l
2 if (<r1>)r < <lk>l

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

15) CR = 0 command accepted
3 device address unknown

USE OF P855M INSTRUCTION SET (I/O EXCLUDED)

MEMORY REFERENCE				CONSTANT		REGISTER TO REGISTER		SHIFT	MISCELLA-NEOUS
direct	indexed	indirect	indexed indirect	short constant	long constant	register register	register addressing		
T4 MD = 10 R2 = 0	T5 MD = 10 R2 ≠ 0	T6 MD = 11 R2 = 0	T7 MD = 11 R2 ≠ 0	T8	T2 MD = 01 R2 = 0	T1 MD = 00	T3 MD = 01 R2 ≠ 0	T8	T8
LD				LDK	LKDL	LDR	LDR*	SLA	
ST				-	-	-	STR	SRA	
AD				ADK	ADKL	ADR	ADR*	SLL	
SU				SUK	SUKL	SUR	SUR*	SRL	
AN				ANK	ANKL	ANR	ANR*	SLC	
-				-	-	TM	-	SRC	
OR				ORK	ORKL	ORR	ORR*	SRN	
XR				XRK	XRKL	XRR	XRR*	DLA	
-				-	-	TNM	-	SLN	
IM				-	-	-	IMR	DRA	
C2				-	-	-	C2R	DLL	
ML				-	MLK	-	MLR	DRL	
MS				-	-	-	MSR	DLC	
ABI				AB	ABL	ABR	ABR*	DRC	
CFI				-	CF	CFR	CFR*	DLN	
-				-	-	-	RTN	DRN	
C1				-	-	C1R	C1R*		LKM
CW				-	CWK	CWR	CWR*		
LC				-	LCK	-	LCR		
SC				-	-	-	SCR		
CC				-	CCK	-	CCR		
-	-	-	-	RF	-	ECR	-		
-	-	-	-	RB	-	-	-		
MU				-	MUK	MUR	MUR*		
DV				-	DVK	DVR	DVR*		
DA				-	DAK	DAR	DAR*		
DS				-	DSK	DSR	DSR*		

Memory Reference Instructions

Syntax:

a) [<ident>]␣<OPC>[*]␣<r1>,<m>[,<r2>]

b) [<ident>]␣<OPC>[*]␣<m>[,<r2>]

c) [<ident>]␣<OPC>[*]␣<n>,<m>[,<r2>]

d) [<ident>]␣<OPC>[(<cnd>)][*]␣<m>[,<r2>]

e) [<ident>]␣<OPC>[(<cnd>]␣<D>

f) [<ident>]␣<OPC>[*]␣[<r1>],<m>[,<r2>]

Mnemonic Function (without indirection and indexing)

LD	(a)	(<m>) → <r1>	(1)
ST	(a)	(<r1>) → <m>	(2)
AD[S]	(a)	(<r1>)+(<m>) → <r1>	(3)
SU[S]	(a)	(<r1>)−(<m>) → <r1>	(3)
AN[S]	(a)	(<r1>) ∧ (<m>) → <r1>	(4)
OR[S]	(a)	(<r1>) ∨ (<m>) → <r1>	(4)
XR[S]	(a)	(<r1>) ∀ (<m>) → <r1>	(4)
IM	(b)	(<m>) + 1 → <m>	(3)
C2	(b)	0 − (<m>) → <m>	(3)
ML	(c)	(<m>)...(<m>+<n>) → A1...An	(5)
MS	(c)	A1...An → <m>...<m>+<n>	(2)
ABI	(d)	(<m>) → P (branch) (P) + 4 → P (no branch)	(2)
CFI	(a)	(P) → (<r1>)(<r1>) − 2 → <r1> (<m>) → P	(2)
C1	(f)	$\overline{(<m>)}$ → <r1>	(4)
C1S	(b)	$\overline{(<m>)}$ → <m>	(4)
CW	(a)	(<r1>) ÷ (<m>) → CR	(6)
CC	(a)	$(<r1>)_r \div (<m>)_{l/r}$ → CR	(7)
LC	(a)	$(<m>)_{l/r}$ → <r1>	(2)
SC	(a)	$(<r1>)_r \rightarrow (<m>)_{r/l}$	(2)
DA	(b)	(A1,A2) + (<m>,<m> + 2) → A1,A2	(3)
DS	(b)	(A1,A2) − (<m>,<m> + 2) → A1,A2	(3)
MU	(b)	(A2) × (<m>) → A1,A2	(3)
DV	(b)	(A1,A2)/(<m>) → A2(quotient) A1(remainder)	(8)

RF (e) (P) + <D> → P (branch)
(P) + 2 → P (no branch) (2)

RB (e) (P) + <D> → P (branch)
(P) + 2 → P (no branch) (2)

Register-To-Register Instructions

Syntax: (a) [<ident>]␣<OPC>[*]␣<r1>,<r2>

(b) [<ident>]␣<OPC>␣<r1>,<r2>

(c) [<ident>]␣<OPC>␣<r2>

(d) [<ident>]␣<OPC>␣<n>,<r2>

(e) [<ident>]␣<OPC>[(<cnd>)][*]␣<r2>

(f) [<ident>]␣<OPC>[*]␣[<r1>],<r2>

(g) [<ident>]␣<OPC>[*]␣<r2>

LDR (a) (<r2>) → <r1> (1)

STR (b) (<r1>) → (<r2>) (2)

ADR[S] (a) (<r1>) + (<r2>) → <r1> (3)

SUR[S] (a) (<r1>) − (<r2>) → <r1> (3)

ANR[S] (a) (<r1>) ∧ (<r2>) → <r1> (4)

ORR[S] (a) (<r1>) ∨ (<r2>) → <r1> (4)

XRR[S] (a) (<r1>) ⊻ (<r2>) → <r1> (4)

IMR (c) ((<r2>)) + 1 → (<r2>) (3)

C2R (c) 0 − ((<r2>)) → (<r2>) (3)

MLR (d) ((<r2>)) → A1
((<r2>) +2) → A2
.
.
((<r2>) + 2n − 2) → An (5)

MSR (d) (A1) → (<r2>)
(A2) → (<r2>) + 2
(An) → (<r2>) + 2n − 2 (2)

ABR (e) (<r2>) → P (branch)
(P) + 2 → P (no branch) (2)

CFR (a) (P) → (<r1>),(<r1>) − 2 → <r1>
(PSW) → (<r1>),(<r1>) − 2 → <r1>
(<r2>) → P (2)

C1R (f) $\overline{(<r2>)}$ → <r1> (3)

C1RS (c) $\overline{((<r2>))}$ → (<r2>) (3)

CWR (a) (<r1>) ÷ (<r2>) → CR (6)

LCR (b) $((<r2>))_{l/r}$ → $<r1>_{r}$ (2)

CCR (b) $(<r1>)_{r}$ ÷ $((<r2>))_{l/r}$ → CR (9)

SCR (b) $(<r1>)_{r}$ → $(<r2>)_{r/l}$ (2)

RTN (c) (<r2>) + 4 → <r2>
((<r2>)) → P
((<r2>)) − 2 → PSW (in system mode only) (10)

ECR (b) (<r2>) → <r1> and $(<r2>)_{r}$ → $(<r1>)_{l}$ (2)

MUR (g) (A2) × (<r2>) → A1,A2 (3)

DVR (g) (A1,A2) / (<r2>) → A2(quotient) A1(remainder) (5)

DAR (g) (<r2>,<r2> + 1) + (A1,A2) → A1,A2 (3)

DSR (g) (A1,A2) − (<r2>,<r2> + 1) → A1,A2 (3)

TM (b) [(<r1>) ∧ (<r2>)] ÷ 0 → CR (4)

TNM (b) [(<r1>) ⊻ (<r2>)] ÷ 0 → CR (4)

Constant Instructions

Syntax. (a) [<ident>]␣<OPC>␣<r3>,<k>

(b) [<ident>]␣<OPC>␣<r1>,<lk>

(c) [<ident>]␣<OPC>␣<n>

(d) [<ident>]␣<OPC>␣<lk>

(e) [<ident>]␣<OPC>(<cnd>)␣[<k>|<lk>]

(f) [<ident>]␣<OPC>␣$<lk_1>$,$<lk_2>$

LDK (a) <k> → $<r3>_{8\text{-}15}$ 0 → $<r3>_{0\text{-}7}$ (11)

LDKL (b) <lk> → <r1> (11)

ADK (a) (<r3>) + <k> → <r3> (3)

ADKL (b) (<r1>) + <lk> → <r1> (3)

SUK (a) (<r3>) − <k> → <r3> (3)

SUKL (b) (<r1>) − <lk> → <r1> (3)

ANK (a) $(<r3>)_{8\text{-}15}$ ∧ <k> → $<r3>_{8\text{-}15}$ (4)

ANKL (b) (<r1>) ∧ <lk> → <r1> (4)

ORK (a) $(<r3>)_{8\text{-}15}$ ∨ <k> → $<r3>_{8\text{-}15}$ (4)

ORKL (b) (<r1>) ∨ <lk> → <r1> (4)

XRK (a) $(<r3>)_{8\text{-}15}$ ⊻ <k> → $<r3>_{8\text{-}15}$ (4)

XRKL (b) (<r1>) ⊻ <lk> → <r1> (4)

MLK (c) $<lk_1>$,$<lk_2>$,.....$<lk_n>$ → A1,A2...An (5)

AB (e) <k> → P (branch)
(P) + 2 → P (no branch) (2)

ABL (e) <lk> → P (branch)

(P) + 2 → P (no branch) (2)

CF (b) (P) → <r1> (<r1>) - 2 → r1
(PSW) → <r1> (<r1>) - 2 → <r1> (2)

CWK (b) (<r1>) ÷ <lk> → CR (12)

CCK (b) $(<r1>)_r \div <lk>_l$ → CR (13)

LCK (b) $<lk>_l$ → $<r1>_r$ (2)

MUK (d) (A2) × <lk> → A1,A2 (3)

DVK (d) (A1,A2) / <lk> → A1(quotient) A2(remainder) (8)

DAK (f) $<lk_1>,<lk_2>$ + (A1,A2) → A1,A2 (3)

DSK (f) (A1,A2) - $<lk_1>,<lk_2>$ → A1,A2 (3)

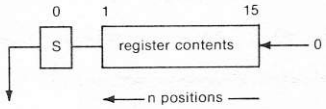
Shift Instructions

Syntax: (a) [<ident>]␣<OPC>␣<r3>,<n>

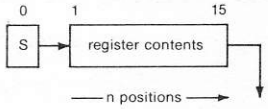
(b) [<ident>]␣<OPC>␣<r3>,<r2>

(c) [<ident>]␣<OPC>␣<n>

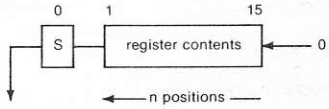
SLA (a) (3)



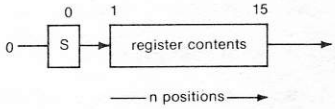
SRA (a) (4)



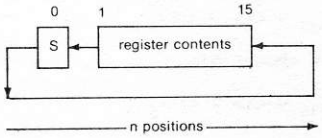
SLL (a) (4)



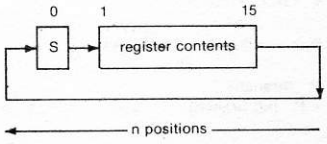
SRL (a) (4)



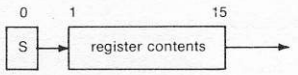
SLC (a) (4)



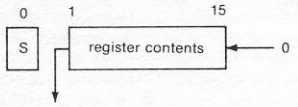
SRC (a) (4)



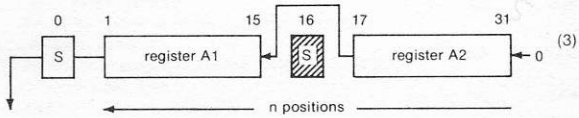
SRN (b) (2)



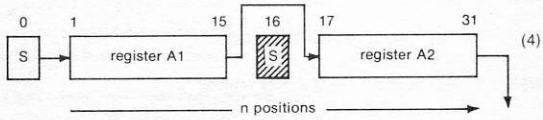
SLN (b) (2)



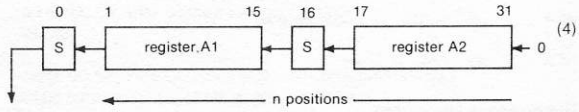
DLA (c) (3)



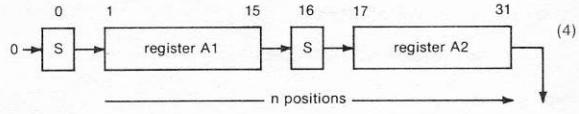
DRA (c) (4)



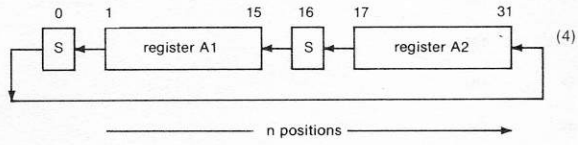
DLL (c) (4)



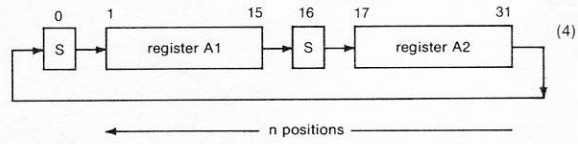
DRL (c) (4)



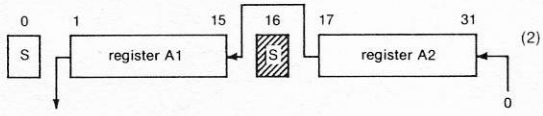
DLC (c) (4)



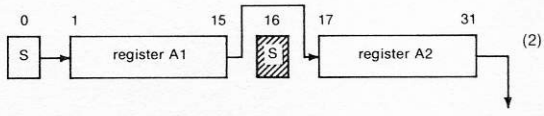
DRC (c) (4)



DLN (c) (2)



DRN (c) (2)



I/O Instructions

syntax:	(a)	[<ident>]␣<OPC>␣<r3>,[0|1],<dev>	
	(b)	[<ident>]␣<OPC>␣<r3>,[0|1],<dev>	
	(c)	[<ident>]␣<OPC>␣<r3>,<dev>	
	(d)	[<ident>]␣<OPC>␣<r3>	
CIO	(a)		(14)
INR	(b)		(14)
OTR	(b)		(14)
SST	(c)		(14)
TST	(c)		(15)
RIL	(d)	Interrupt lines → <r3>	(2)
WMP	(d)	(<r3>) → M.P. Key register	(2)
WM2	(d)	(<r3>) → M.P. Key register	(2)
WIM	(d)	(<r3>) → mask register	(2)
RCA	(d)	CNL → <r3>	(2)

Miscellaneous Instructions

Syntax:	(a)	[<ident>]␣<OPC>	
	(b)	[<ident>]␣<OPC>␣<k>	
ENB	(a)		(2)
HLT	(a)		(2)
SMD	(a)		(2)
RIT	(b)		(2)
INH	(a)		(2)
LKM	(a)		(2)

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>, ...,
<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>]

LKM	Monitor requests	BOM	DOM	SRTM	BRTM	DRTM
1	I/O	X	X	X	X	X
2	Wait for an event	X	X	X	X	X
3	Exit	X	X	X	X	X
4	Get Buffer	X	X		X	X
5	Release Buffer	X	X		X	X
6	Pause	X	X			
7	Keep control on abort condition	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

I/O FUNCTIONS

/01 Basic Read
/02 Standard Read
/05 Basic Write
/06 Standard Write
/07 Object Write 4 × 4
/08 Object Write 8+8
/0A Random Read (only for disc)
/0B Random Write (only for disc)
/30 Get Information about a file code

TAPE HANDLING ORDERS

/14 Skip Forward to next EOS mark
/16 Skip Forward to next EOF mark
/22 Write EOS mark
/24 Write EOV (End of Volume) Mark
/26 Write EOF mark
/31 Rewind to Load Point
/33 Backspace one block
/34 Skip backwards to EOF mark
/37 Lock
/38 Unlock

SOFTWARE STATUS WORD

Request accepted

For Basic orders: status word gives the hardware status

For Standard orders: Status 8000:			no operation (after RD message)
Status 8XXX:		bit 8	End Of Volume mark
		10	beginning of tape
		11	end of disc/tape
		12	incorrect length requested
		13	illegal character code or checksum error (object)
		14	EOS
		15	EOF
Status 0000:		normal	I/O completion

Request rejected

For Basic and Standard orders:

Status CXXX:		bit 5	disc overflow (no more granules available)
		6	no disc buffer available (Dynamic allocation area overflow)
		7	disc queue overflow
DRTM		11	function inknown or not compatible with device
	BOM	12	illegal buffer size
	DOM	13	illegal buffer address
	BRTM	14	device attached to another program
		15	illegal file code or non-existing

C.U. STATUS WORD CONFIGURATION

Bit	Description	CU									
		ASR	CR	MHD	LP	TP	PTR	CASS Tape	PLOT	FHD	MT
0	-										
1	ready			x				x			x
2	rewind										x
3	tape mark has been read							x			x
4	-										
5	on cylinder load point			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	x
12	Incorrect length		x	x				x		x	x
	Y limit overpass								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	x

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

DATA COMMUNICATION

LKM ± 8 (- 8, <sched lab>)

I/O requests DATEM

/01 Read with Echo
/02 Read without Echo
/03 Read with Echo and without Time-out
/04 Read with Echo and with Time-out
/06 Write a Transmission Block
/0F Get Line Type
/10 Stop the Exchange
/11 Disconnect the Line
/12 Search Line Address
/13 Accept Automatic Call
/14 Accept Data
/15 Set Time-out

I/O requests BSC

01 Read Initial
02 Write Initial
} Point-to-point lines
03 Read Initial
04 Write Initial
} Multipoint lines
05 Search
06 Read Continue
07 Read Repeat
08 Write Continue
09 Write EOT
10 Write Forward Abort
11 Request for Line Disconnection

Line Control Table

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
WORD 0								▒					▒	▒	▒	▒
WORD 1											▒	▒				
WORD 2	TIME OUT VALUE															
WORD 3	ECB ADDRESS															
WORD 4	CURRENT CHARACTER ADDRESS															
WORD 5	NUMBER OF CHARACTERS REQUESTED															
WORD 6	NUMBER OF RECEIVED / SENT CHARACTERS															
WORD 7	WAIT FOR DATA BUFFER OR SYN PATTERN (IN AND OUT) (MELCU) (SYLCU)															
WORD 8	SCHEDULED LABEL ADDRESS															

word 0 bit 0 = 1 busy
1 = 1 write
2 = 1 read
3 = 1 echo
4 = 1 wait for data/ENQ
5 = 1 wait for call
6 = 1 time-out
8 = 1 multiplex channel
0 programmed channel
9 = 1 SYLCU
0 ASYLCU
10 = 1 MELCU
11 = 1 full duplex
0 half duplex

word 1
- bit 0 = 1 connected
- 1 = 1 start bit received
- 2 = 1 echo refused
- 3 = 1 character received
- 4 = 1 character sent
- 5 = 1 escape received
- 6 = 1 check character
- 7 = 1 wait for data character received
- 8 = 1 refuse this character
- 9 = 1 two more characters requested
- 12-15 line number (0 to 7 or /F)

DATEM

0F GET LINE TYPE

- bit 15, 14: 00 = ASYLCU; 01 = MELCU; 10 = SYLCU
- 13 = 1 DERCO (SYLCU) or MIU (MELCU)
- 12 = 1 connected
- 11 not used (zero)
- 10 = 1 Multiplex Channel
- 9 = 1 full-duplex
- 8 = 1 CRC computer (DERCO)
- 7 = 1 parity computed
- 6 = 1 odd parity; 0 = even parity
- 5-1 not used (zeroes)
- 0 line busy

DATEM ECB

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

WORD 0	EVENT CHARACTER	LOGICAL LINE NO
WORD 1	TRANSMISSION BLOCK ADDRESS	
WORD 2	REQUESTED LENGTH	
WORD 3	TRANSMITTED LENGTH	
WORD 4	STATUS WORD	
WORD 5	SPECIAL CHARACTER TABLE ADDRESS	
WORD 6	TIME-OUT VALUE	

The status word has the following format:

- bit 15 = 1 modem not operable
- 14 = 1 throughput error
- 13 = 1 parity error
- 12 = 1 I/O error
- 11 = 1 Escape received (interrupt in transmission block)
- 10 = 1 time over
- 9 = 1 not used
- 8 = 1 transmission stopped
- 7 = 1 buffer overflow
- 6 = 1 time-out request may not be served
- 5 = 1 non-positive message length
- 4 = 1 characters lost (wait for data)
- 3 = 1 non-compatible request, e.g. Wait for Data for SYLCU or illegal request
- 2 = 1 illegal file code
- 1 = 1 non-connected line
- 0 = 1 line is busy

BSC ECB

	0	8	9	11	12	13	14	15
WORD 0	+	*	+	*	*	+/*	+/*	+/*
WORD 1	RESERVED	LINE CODE						
WORD 2	BUFFER AREA ADDRESS							
WORD 3	BUFFER LENGTH							
WORD 4	RESERVED							
WORD 5	STATUS WORD							
WORD 6	RESERVED							
WORD 7	POLLING/SELECTING TABLE ADDRESS OR TIME-OUT VALUE							
WORD 8	LINE AREA ADDRESS							

* BIT SET/RESET BY PROGRAM + BIT SET/RESET BY USER

Where:

Word 0
- bit 0 = 1 the operation is completed
- 8 = 0 slave/tributary station
- 1 master/control station
- 9 = 1 ITB control character received
- 11 = 0 ASCII-code
- 1 EBCDIC-code
- 12 = 0 LRC
- 1 CRC
- 13 = 0/1 user is sending a header (reset by BSC)
  user has received a header (to be reset by user)

Word 5 **Status word**

The status word is divided as follows:
- bits 0 - 3 Only one of four bits is set to give the kind of status:
  - bit 0 Acknowledgement status
  - bit 1 Software error status
  - bit 2 Transmission error status
  - bit 3 Procedure error status
- bits 4 - 15 give, per bit, a description of the status.

Acknowledgement status (bit 0)
- bit 15 Conversational mode received
- 14 ENQ received (contention: local station is slave)
- 13 RVI received
- 12 EOT received (normal condition)
- 11 Forward abort, STXENQ or EOT received
- 10 ENQ received or write EOT

Software error status (bit 1)
- bit 15 Write conversational refused (NAK pending)
- 14 Non-positive text length (Write request)
- 13 Pending BSC request on this line
- 12 Header transmission refused
- 11 Illegal request
- 10 Illegal file code
- 9 Non-connected line
- 8 Line busy

(10, 9, 8: detected by DATEM)
- 7 System error (stack error)
- 6 System error (time-out not serviced)

Transmission error status (bit 2)
- bit 15 Modem not operable
- 14 Throughput error
- 13 Parity error
- 12 BCC error (software computation)

SYSTEM MESSAGES

Message	Program	Meaning
A:	ASM	Request for option/control message. See page 8
A:	IPLRT	Request for activate message. Answer with A or LF CR
ABORT <code> <address>	BOM	Program aborted
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
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
BOOT CK ER	IPL	Checksum error encountered
C:	UPD	Request for control message. See page 9
C?	UPD	Parameter error. Request for correct control message
*C	ASM	Illegal constant
CATALOG OVERFLOW	CCI	Too many userids catalogued
COMMAND NOT ALLOWED	CCI	This is not a system userid
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. See page 10
DATE:	DOM/ DRTM	Disk system asking for date. DD MM YY or YY MM DD
D.D.	LKE	Double definition error

Message	Program	Meaning
DBL.DEF <name>	LKE	Name is defined more than once as an entry point or in the name of a common block
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	CCI	Directory overflow
DIRECTORY OVERFLOW ON XXXXXFT	CCI	Directory overflow. File FT is catalogued
DISK ADDRESS MISSING	CCI	Disk address not specified
DISK <address> UNKNOWN	DRTM	Disk unknown by CPU
DISK ASSIGN ERROR	CCI	System cannot assign a temporary work file
DISK FILE CODE ABSENT	CCI	Disk file code not specified in command
DISK FILE CODE ERROR	CCI	2nd parameter not numeric
DISK FILE CODE UNKNOWN	CCI	File code not declared at sysgen
DISK FILE CODE MISSING	CCI	No file code specified
DISK NOT OPERATIONAL	CCI	Disk unit not ready
DISK I/O ERROR	CCI	I/O error on disk
DISK OVERFLOW	CCI	No free granule available to allocate to temporary disk file
DISK UNKNOWN	CCI/DRTM	Disk not specified at sysgen
DKER␣<address><sect number> <status>	DOM/ DTRM	Disk not ready to be used
DSK INPUT ERR, UPD ABORTED	CCI	Erroneous input for Disk Update
DSK INIT ERR	DOM/ DRTM	Disk not ready to be used
DSK OUTPUT ERR, UPD ABORTED	CCI	Erroneous output from Disk Update
DYN AREA LENGTH:	DRTM	Length of dynamic area requested (4 hexa char.)
** DRTM ** yy **	DRTM	yy = release number
* E	ASM	Address is not even

Message	Program	Meaning
E = <absolute address>	LKE	Address is the highest absolute address in the generated module
****** E	ASM	END directive missing
EC	BOM/ BRTM IPLRT	Erroneous cluster or input error encountered during loading
EC TYPE	LKE	Erroneous cluster encountered
END MIS	LKE	END cluster missing
:EOF	BOM/ASM	End-of-file mark encountered
EOF, UPD TERMINATED	CCI	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 00	SCL	Command unknown
01	SCL	System 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
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

Message	Program	Meaning
15	SCL	Parameter error
16	SCL	The specified timer has not been assigned
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
51	SCL	I/O error on disk
52	SCL	No spare entry available in FCT
53	SCL	No disk 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.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
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

Message	Program	Meaning
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	CCI	CCI dit 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 disk to be copied not specified in command
FIRST FILE CODE UNKNOWN	CCI	File code not known by system
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	CII	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 assigned. The message is followed by the reason.
INVALID DISK FILE CODE	CCI	Second parameter not in range /F0 to /FF.

Message	Program	Meaning
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
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 9
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
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

Message	Program	Meaning
MODULE UNKNOWN	CCI	Object module unknown
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
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 disk
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)

Message	Program	Meaning
PARAM MISSING	CCI	Request for program control table size (type in 4 digit hexa number)
PRG.OVL	CCI	Generated load module exceeds 32k
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
PROGRAM NAME ERROR	CCI	Erroneous name specified
PROGRAM NOT CATALOGUED	CCI	Program not catalogued
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
R	DBG	Request input from external device
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
*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

Message	Program	Meaning
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
SEQUENCE ERR, [TRY AGAIN]	CCI	Update the lines in ascending order
START ADDR. REDUNDANT	CCI	Start address is redundant
SYNTAX ERR, [TRY AGAIN]	CCI	Error in Update command
SYSTEM DISK I/O ERROR	CCI	I/O error
SYSTEM LIB ASSIGN ERROR	CCI	Assign error in system library
SYSTEM TAPE ON READER, PLEASE	IPL	Load request for system program
SYSTEM SESSION COMMAND	CCI	LIC (List Catalogue) may only be used in system session
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 10
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

Message	Program	Meaning
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
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

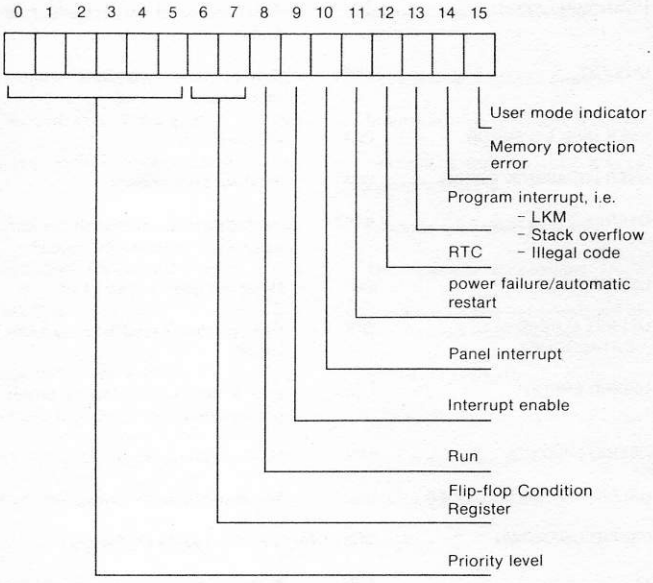
STANDARD PRIORITY INTERRUPT LEVELS

0	Power failure	interrupt location /40
1	LKM - stack overflow - illegal code	interrupt location /42
2	Real time clock	interrupt location /44
3	Memory Increment Data Break	interrupt location /46
4	Reserved	interrupt location /48
5	Common interrupt line	interrupt location /4A
6	Memory protect	interrupt location /4C
7	Control panel	interrupt location /4E

STANDARD CONNECTIONS COMMON INTERRUPT LINE

0	Reserved	*(lowest priority)*
1	Plotter	
2	Line printer	
3	Moving head disc 2 (not serviced by BOM)	
4	Moving head disc 1 (not serviced by BOM)	
5	Fixed head disc (not serviced by BOM)	
6	Magnetic tape	
7	(Operator's typewriter 2)	
8	Operator's typewriter 1	
9	(Tape punch 2)	
10	Tape punch 1	
11	Cassette tape	
12	(Punched tape reader 2)	
13	Punched tape reader 1	
14	Card reader	
15	Reserved	*(highest priority)*

PROGRAM STATUS WORD



bit 8, 9 and 13 are not copied into the stack but they are displayed on the control panel

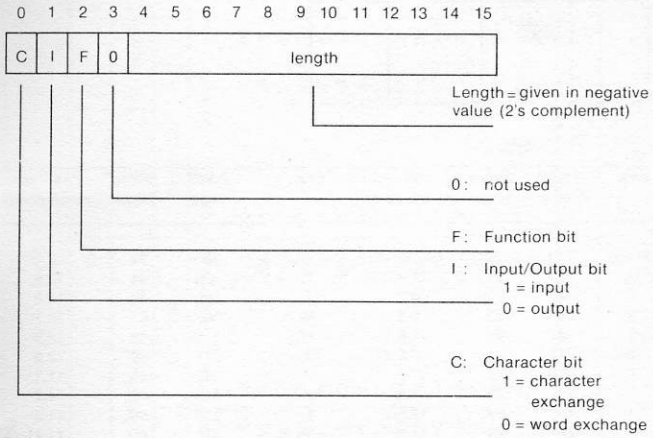
MEMORY PROTECTION

If a certain bit is set, the corresponding memory area is protected.

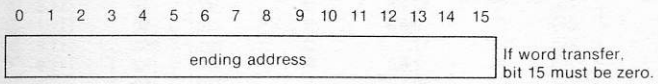
MK1			MK2		
bit	memory area	module	bit	memory area	module
15	0000 - 07FE	MM0	15	8000 - 87FE	MM4
14	0800 - 0FFE		14	8800 - 8FFE	
13	1000 - 17FE		13	9000 - 97FE	
12	1800 - 1FFE		12	9800 - 9FFE	
11	2000 - 27FE	MM1	11	A000 - A7FE	MM5
10	2800 - 2FFE		10	A800 - AFFE	
9	3000 - 37FE		9	B000 - B7FE	
8	3800 - 3FFE		8	B800 - BFFE	
7	4000 - 47FE	MM2	7	C000 - C7FE	MM6
6	4800 - 4FFE		6	C800 - CFFE	
5	5000 - 57FE		5	D000 - D7FE	
4	5800 - 5FFE		4	D800 - DFFE	
3	6000 - 67FE	MM3	3	E000 - E7FE	MM7
2	6800 - 6FFE		2	E800 - EFFE	
1	7000 - 77FE		1	F000 - F7FE	
0	7800 - 7FFE		0	F800 - FFFE	

MULTIPLEX CONTROL WORDS

1st word



2nd word



The control unit number = multiplex control word couple number

FIXED MEMORY LOCATIONS

Bootstrap	:	/0 thru /3E
Hardware interrupt locations	:	/40 thru /7E
Interrupt Table address	:	/80
Communication Vector Table address	:	/82
Multiplex control words	:	/84 thru /BE
Hardware interrupt locations	:	/C0 thru /DE
Stack	:	/100

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

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

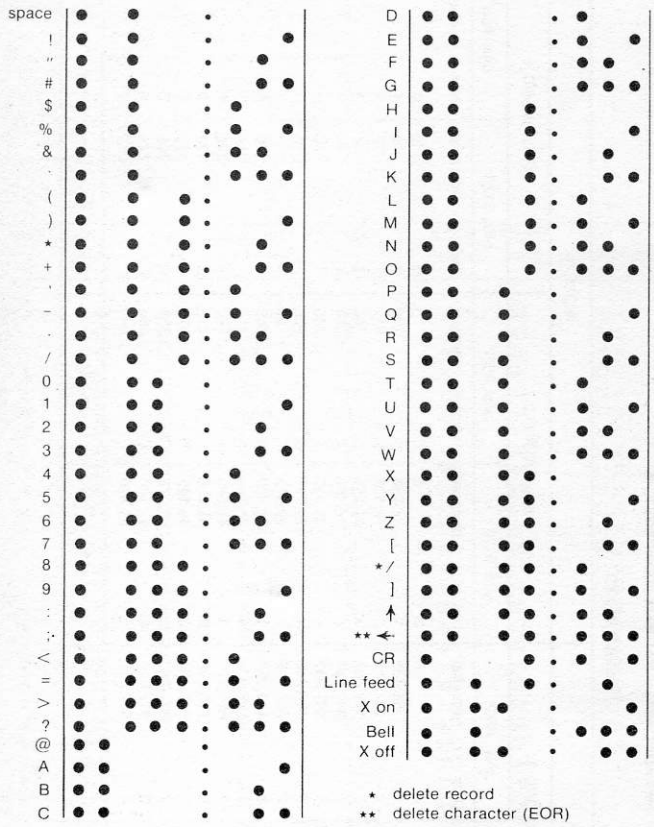
char.	*ASCII octal*	*Intern Hexa*	*char. set punch comb.*	*char.*	*ASCII octal*	*Intern Hexa*	*char. set punch comb.*
space	240	20	on punch	D	304	44	12,4
!	241	21	11,8,2	E	305	45	12,5
"	242	22	8,7	F	306	46	12,6
#	243	23	8,3	G	307	47	12,7
$	244	24	11,8,3	H	310	48	12,8
%	245	25	0,8,4	I	311	49	12,9
&	246	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



* delete record
** delete character (EOR)