SPERRY UNIVAC
90/30 Data Processing System

Hardware and Software Summary

SPERRY UNIVAC

UP-8203 Rev. 1

H

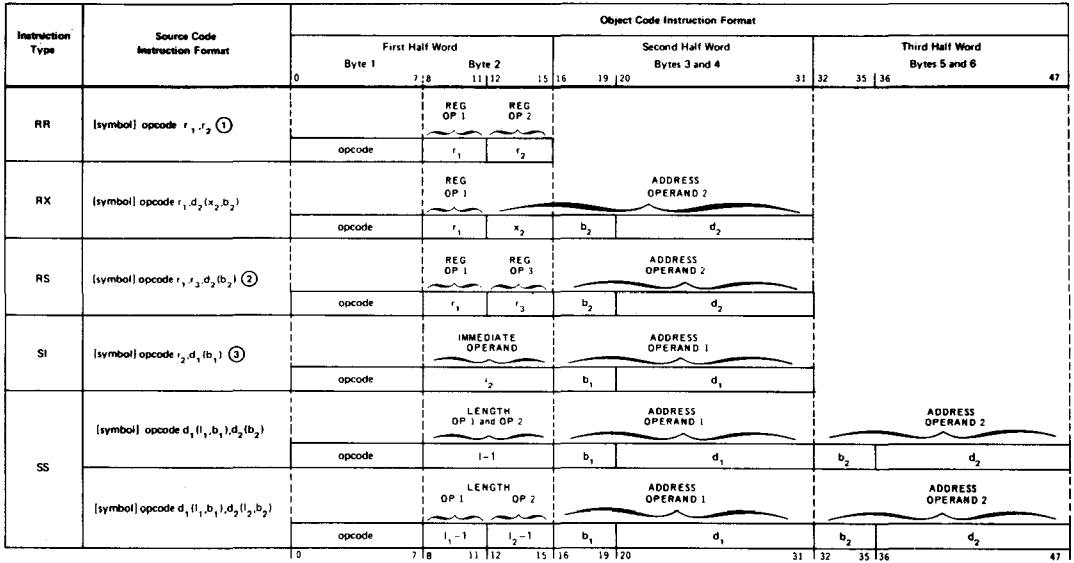
This document contains the latest information available at the time of publication. However, Sperry Univac reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Sperry Univac representative.

Sperry Univac is a division of Sperry Rand Corporation.

AccuScan, FASTRAND, PAGEWRITER, SPERRY UNIVAC, UNISCOPE, UNISERVO, and UNIVAC are trademarks of the Sperry Rand Corporation.

 PRINTED IN U.S.A.

CONTENTS



NOTES:

(1) The RR instruction has two other forms:

[symbol] opcode i_1 for the SVC and LBR instructions, and
[symbol] opcode r_1 for the SPM instruction.

(2) The RS shift instructions are written without use of the r_3 operand, in the form:

[symbol] opcode $r_1,d_2(b_2)$

(3) Some SI instructions, such as HIO and TIO, do not use an i_2 field. They are written in the form:

[symbol] opcode $d_1(b_1)$

Legend for Instruction Formats

Symbol	Meaning
opcode	Instruction operation code
r_1	Number of the register containing operand 1 or a register which is the first register of a multiregister group
r_2	Number of the register containing operand 2
r_3	An expression representing the last register in a multiregister group, an increment, an operand address, or a control storage address
x_2	Number of the register to be used as an index for operand 2 of an RX instruction
i_1	Immediate data used as operand 1 of the SVC instruction
i_2	Immediate data used as operand 2 of an SI instruction
l	Length of operands 1 and 2 as stated in source code*
l_1	Length of operand 1 as stated in source code*

l_2	Length of operand 2 as stated in source code*
b_1	Base register for operand 1
b_2	Base register for operand 2
d_1	Displacement for operand 1
d_2	Displacement for operand 2
m_1	Operand 1 mask
op_1	Operand 1
op_2	Operand 2
op_3	Operand 3
S_1	Symbol used to identify operand 1 in the implicit format
S_2	Symbol used to identify operand 2 in the implicit format

*This is coded as the true source code length of the operand, not the length less 1, as assembled in the object code. The assembler reduces the length by 1 when converting source code to object code.

Instruction	Mnemonic Code	Opcode	Use	Type	Instruction Source Formats		Execution Time in Microseconds
					Explicit	Implicit	
Fixed-Point Instructions							
Add (cc)	A	5A	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Add half word (cc)	AH	4A	N,C2	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Add half word (cc)	AH	AA	C1	RX	$r_1,d_2(,b_2)$	r_1,s_2	5.4
Add immediate (cc)	AI	9A	C2*	SI	$d_1(b_1),i_2$	s_1,i_2	6.0
Add immediate (cc)	AI	A6	C1	SI	$d_1(b_1),i_2$	s_1,i_2	6.0
Add (cc)	AR	1A	N,C2	RR	r_1,r_2	r_1,r_2	Native=3.0 (360/20=3.6)
Compare (cc)	C	59	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Compare half word (cc)	CH	49	N,C3	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Compare (cc)	CR	19	N	RR	r_1,r_2	r_1,r_2	3.0
Convert to binary	CVB	4F	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	36.0
Convert to decimal	CVD	4E	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	66.0+6.0s4
Divide	D	5D	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	65.4+1.2s1+0.6rn
Divide	DR	1D	N,F	RR	r_1,r_2	r_1,r_2	64.8+1.2s1
Load	L	58	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	4.8

Load complement (cc)	LCR	13	N,F	RR	r_1,r_2	r_1,r_2	3.0
Load half word	LH	48	N,C3	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Load multiple	LM	98	N	RS	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2	3.0+1.8gr
Load negative (cc)	LNR	11	N,F	RR	r_1,r_2	r_1,r_2	4.2
Load positive (cc)	LPR	10	N,F	RR	r_1,r_2	r_1,r_2	4.2
Load	LR	18	N	RR	r_1,r_2	r_1,r_2	3.0
Load and test (cc)	LTR	12	N	RR	r_1,r_2	r_1,r_2	3.0
Multiply	M	5C	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	39.6+0.6s1+0.6s2+0.6rn
Multiply half word	MH	4C	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	24.0+0.6s1+1.8s2+0.6rn
Multiply	MR	1C	N,F	RR	r_1,r_2	r_1,r_2	39.0+0.6s1+0.6s2+0.6rn
Subtract (cc)	S	5B	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Subtract half word (cc)	SH	4B	N,C2	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Subtract half word (cc)	SH	AB	C1	RX	$r_1,d_2(,b_2)$	r_1,s_2	5.4
Shift left single (cc)	SLA	8B	N,F	RS	$r_1,d_2(b_2)$	r_1,s_2	7.2+0.6p+0.6q
Shift left double (cc)	SLDA	8F	N,F	RS	$r_1,d_2(b_2)$	r_1,s_2	7.8+1.2p+1.2q

Instructions by Application (cont)

Instruction	Mnemonic Code	Opcode	Use	Type	Instruction Source Formats		Execution Time in Microseconds
					Explicit	Implicit	
Fixed-Point Instructions (cont)							
Supervisor load multiple (pi)	SLM	B8	N	RS	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2	4.2+1.8gr
Subtract (cc)	SR	1B	N,C2	RR	r_1,r_2	r_1,r_2	Native=3.0 (360/20=3.6)
Shift right single (cc)	SRA	8A	N,F	RS	$r_1,d_2(b_2)$	r_1,s_2	5.4+0.6p+0.6q
Shift right double (cc)	SRDA	8E	N,F	RS	$r_1,d_2(b_2)$	r_1,s_2	6.0+1.2p+1.2q
Supervisor store multiple (pi)	SSTM	B0	N	RS	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2	4.2+1.2gr
Store	ST	50	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Store half word	STH	40	N,C3	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	4.8
Store multiple	STM	90	N	RS	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2	4.2+1.2gr
Decimal Instructions							
Add decimal (cc)	AP	FA	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	36.6+0.75n1+0.375n2+6.0t1+3.0s5
Compare decimal (cc)	CP	F9	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	31.8+0.375n1+0.375n2+2.4s
Divide decimal (cc)	DP	FD	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	37.8+0.75n1+6.375n2+24.6(n1−n2)
Multiply decimal (cc)	MP	FC	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	36.4+0.75n1+14.4(n1−n2)+0.375n2
Move with offset	MVO	F1	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	10.2+1.2n1+1.2n2

Pack	PACK	F2	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	12.0+1.2(n1−1)+1.2(n2−1)
Subtract decimal (cc)	SP	FB	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	36.6+0.75n1+0.375n2+6t1+3.0s6
Unpack	UNPK	F3	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	12.0+1.2(n1−1)+1.2(n2−1)
Zero and add (cc)	ZAP	F8	N,C3	SS	$d_1(l_1,b_1)d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$	16.2+1.8n7+1.2n8+1.8t2(n2−n1)
Floating-Point Instructions							
Add normalized, long format (cc)	AD	6A	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	19.2+1.2ce+1.2pr+1.2t1+1.2rp
Add normalized, long format (cc)	ADR	2A	N,F	RR	r_1,r_2	r_1,r_2	16.2+1.2ce+1.2pr+1.2t1+1.2rp
Add normalized, short format (cc)	AE	7A	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	16.8+1.2ce+1.2pr+1.2t1+1.2rp
Add normalized, short format (cc)	AER	3A	N,F	RR	r_1,r_2	r_1,r_2	14.4+1.2ce+1.2pr+1.2t1+1.2rp
Add unnormalized, short format (cc)	AU	7E	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	16.8+1.2ce+0.6rp
Add unnormalized, short format (cc)	AUR	3E	N,F	RR	r_1,r_2	r_1,r_2	14.4+1.2ce+0.6rp
Add unnormalized, long format (cc)	AW	6E	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	19.2+1.2ce+0.6rp
Add unnormalized, long format (cc)	AWR	2E	N,F	RR	r_1,r_2	r_1,r_2	16.2+1.2ce+0.6rp
Compare, long format (cc)	CD	69	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	21.6+1.2ce
Compare, long format (cc)	CDR	29	N,F	RR	r_1,r_2	r_1,r_2	18.0+1.2ce
Compare, short format (cc)	CE	79	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	18.0+1.2ce
Compare, short format (cc)	CER	39	N,F	RR	r_1,r_2	r_1,r_2	15.6+1.2ce

Instructions by Application (cont)

Instruction	Mnemonic Code	Opcode	Use	Type	Instruction Source Formats		Execution Time in Microseconds
					Explicit	Implicit	
Floating-Point Instructions (cont)							
Divide, long format	DD	6D	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	208.2+0.6p1+0.6p2+15.0pn+0.6rn
Divide, long format	DDR	2D	N,F	RR	r_1,r_2	r_1,r_2	205.2+0.6p1+0.6p2+15.0pn+0.6rn
Divide, short format	DE	7D	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	47.4+0.6p1+0.6p2+15.0pn+0.6rn
Divide, short format	DER	3D	N,F	RR	r_1,r_2	r_1,r_2	45.0+0.6p1+0.6p2+6.6pn+0.6rn
Halve, long format	HDR	24	N,F	RR	r_1,r_2	r_1,r_2	7.8+1.2pr+.6pn+0.6(s2)
Halve, short format	HER	34	N,F	RR	r_1,r_2	r_1,r_2	7.2+1.2pr+0.6pn
Load complement, long format (cc)	LCDR	23	N,F	RR	r_1,r_2	r_1,r_2	4.8
Load complement, short format (cc)	LCER	33	N,F	RR	r_1,r_2	r_1,r_2	4.2
Load, long format	LD	68	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	6.6
Load, long format	LDR	28	N,F	RR	r_1,r_2	r_1,r_2	4.2
Load, short format	LE	78	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Load, short format	LER	38	N,F	RR	r_1,r_2	r_1,r_2	3.6
Load negative, long format (cc)	LNDR	21	N,F	RR	r_1,r_2	r_1,r_2	4.2
Load negative, short format (cc)	LNER	31	N,F	RR	r_1,r_2	r_1,r_2	3.6

Load positive, long format (cc)	LPDR	20	N,F	RR	r_1,r_2	r_1,r_2	4.2
Load positive, short format (cc)	LPER	30	N,F	RR	r_1,r_2	r_1,r_2	3.6
Load and test, long format (cc)	LTDR	22	N,F	RR	r_1,r_2	r_1,r_2	4.8
Load and test, short format (cc)	LTER	32	N,F	RR	r_1,r_2	r_1,r_2	4.2
Multiply, long format	MD	6C	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	118.2+0.6p1+0.6p2+1.2pn+0.6rn
Multiply, long format	MDR	2C	N,F	RR	r_1,r_2	r_1,r_2	115.2+0.6p1+0.6p2+1.2pn+0.6rn
Multiply, short format	ME	7C	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	41.4+0.6p1+0.6p2+0.6pn+0.6rn
Multiply, short format	MER	3C	N,F	RR	r_1,r_2	r_1,r_2	39.0+0.6p1+0.6p2+0.6pn+0.6rn
Subtract normalized, long format (cc)	SD	6B	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	19.2+1.2ce+1.2pr+1.2t1+1.2rp
Subtract normalized, long format (cc)	SDR	2B	N,F	RR	r_1,r_2	r_1,r_2	16.2+1.2ce+1.2pr+1.2t1+1.2rp
Subtract normalized, short format (cc)	SE	7B	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	16.8+1.2ce+1.2pr+1.2t1+1.2rp
Subtract normalized, short format (cc)	SER	3B	N,F	RR	r_1,r_2	r_1,r_2	14.4+1.2ce+1.2pr+1.2t1+1.2rp
Store, long format	STD	60	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	7.2
Store, short format	STE	70	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	6.0

Instructions by Application (cont)

Instruction	Mnemonic Code	Opcode	Use	Type	Instruction Source Formats		Execution Time in Microseconds
					Explicit	Implicit	
Floating-Point Instructions (cont)							
Subtract unnormalized, short format (cc)	SU	7F	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	16.8+1.2ce−0.6a
Subtract unnormalized, short format (cc)	SUR	3F	N,F	RR	r_1,r_2	r_1,r_2	14.4+1.2ce−0.6a
Subtract unnormalized, long format (cc)	SW	6F	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	19.2+1.2ce−0.6a
Subtract unnormalized, long format (cc)	SWR	2F	N,F	RR	r_1r_2	r_1,r_2	16.2+1.2ce+0.6rp
Logical Instructions							
Add logical (cc)	AL	5E	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Add logical (cc)	ALR	1E	C1,F	RR	r_1,r_2	r_1,r_2	3.0
Compare logical (cc)	CL	55	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Compare logical (cc)	CLC	D5	N,C3	SS	$d_1(l,b_1),d_2(b_2)$**	$s_1(l),s_2$**	9.6+1.2b
Compare logical (cc)	CLI	95	N,C3	SI	$d_1(b_1),i_2$	s_1,i_2	4.8

Compare logical (cc)	CLR	15	N	RR	r_1,r_2	r_1,r_2	3.0
Edit (cc)	ED	DE	N,C3	SS	$d_1(l_1,b_1),d_2(b_2)$	$s_1(l_1),s_2$	9.0+3.0n+0.6n3+3.0n4+0.6n6
Edit and mark (cc)	EDMK	DF	N,F	SS	$d_1(l_1,b_1),d_2(b_2)$	$s_1(l_1),s_2$	9.0+3.0n+1.2n3+3.0n4+1.2n6
Insert character	IC	43	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	4.2
Load address	LA	41	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	4.2
Move	MVC	D2	N,C3	SS	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$	7.6+0.6n+0.6t4(n−1)
Move	MVI	92	N,C3	SI	$d_1(b_1),i_2$	s_1,i_2	4.8
Move numerics	MVN	D1	N,C3	SS	$d_1(l_1,b_1),d_2(b_2)$	$s_1,(l),s_2$	10.2+2.1n
Move zones	MVZ	D3	N,C2	SS	$d_1(l,b_1),d_2(b_2)$	$s_1,(l),s_2$	10.2+2.1n
AND (cc)	N	54	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
AND (cc)	NC	D4	N,C1	SS	$d_1(l,b_1),d_2(b_2)$	$s_1,(l),s_2$	10.2+1.5n***
AND (cc)	NI	94	N,C3	SI	$d_1(b_1),i_2$	s_1,i_2	6.0
AND (cc)	NR	14	N	RR	r_1,r_2	r_1,r_2	3.0
OR (cc)	O	56	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
OR (cc)	OC	D6	N,C1	SS	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$	10.2+1.5n
OR (cc)	OI	96	N,C3	SI	$d_1(b_1),i_2$	s_1,i_2	6.0
OR (cc)	OR	16	N	RR	r_1,r_2	r_1,r_2	3.0

Instructions by Application (cont)

Instruction	Mnemonic Code	Op-code	Use	Type	Instruction Source Formats		Execution Time in Microseconds
					Explicit	Implicit	
Logical Instructions (cont)							
Subtract logical (cc)	SL	5F	N,F	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Shift left double logical	SLDL	8D	N,F	RS	$r_1,d_2(b_2)$	r_1,s_2	4.8+1.2p+1.2q
Shift left single logical	SLL	89	N	RS	$r_1,d_2(b_2)$	r_1,s_2	5.4+0.6p+0.6q
Subtract logical (cc)	SLR	1F	N,F	RR	r_1,r_2	r_1,r_2	3.0
Shift right double logical (cc)	SRDL	8C	N,F	RS	$r_1,d_2(b_2)$	r_1,s_2	4.8+1.2p+1.2q
Shift right single logical (cc)	SRL	88	N	RS	$r_1,d_2(b_2)$	r_1,s_2	5.4+0.6p+0.6q
Store character	STC	42	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	4.8
Test under mask (cc)	TM	91	N,C3	SI	$d_1(b_1),i_2$	s_1,i_2	6.0
Translate	TR	DC	N,C3	SS	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$	7.2+2.4n
Translate and test (cc)	TRT	DD	N	SS	$d_1(l,b_1),d_2(l_2,b_2)$	$s_1(l),s_2$	8.4+1.8b
Exclusive OR (cc)	X	57	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Exclusive OR (cc)	XC	D7	N	SS	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$	10.2+1.5n
Exclusive OR (cc)	XI	97	N	SI	$d_1(b_1),i_2$	s_1,i_2	6.0
Exclusive OR (cc)	XR	17	N	RR	r_1,r_2	r_1,r_2	3.0

Branching Instructions							
Branch and link	BAL	45	N,C1	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	6.0
Branch and link	BALR	05	N	RR	r_1,r_2	r_1,r_2	3.6+0.6s
Branch and store	BAS	4D	C2	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	5.4
Branch and store	BASR	0D	C2	RR	r_2,r_2	r_1,r_2	3.0+0.6s
Branch on condition (em)	BC	47	N,C3	RX	$m_1,d_2(x_2,b_2)$	$m_1,s_2(x_2)$	3.6
Branch on condition (em)	BCR	07	N,C2	RR	m_1,r_2	m_1,r_2	3.0
Branch on count	BCT	46	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	4.2
Branch on count	BCTR	06	N	RR	r_1,r_2	r_1,r_2	3.6
Branch on index high	BXH	86	N,F	RS	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2	7.2-1.2s3
Branch on index low or equal	BXLE	87	N,F	RS	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2	7.2-1.2s3
Execute	EX	44	N	RX	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$	3.6+0.6r+0.6nrr+e
Status Switching Instructions							
Halt and proceed (pi)	HPR	99	N,C2*	SI	$d_1(b_1),i_2$	s_1,i_2	3.6
Insert storage key (pi)	ISK	09	N,F	RR	r_1,r_2	r_1,r_2	4.2
Load control storage (pi)	LCS	B1	N	RS	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2	5.4+24.0w+4.8s8
Load program status word (pi,cc)	LPSW	82	N	SI	$d_1(b_1),i_2$	s_1,i_2	11.4

Instructions by Application (cont)

Instruction	Mnemonic Code	Op-code	Use	Type	Instruction Source Formats		Execution Time in Microseconds
					Explicit	Implicit	
Status Switching Instructions (cont)							
Set program mask (cc)	SPM	04	N	RR	r_1	r_1	3.0
Set storage key (pi)	SSK	08	N,F	RR	r_1,r_2	r_1,r_2	4.2
Set system mask (pi)	SSM	80	N	SI	$d_1(b_1)$	s_1	4.8
Supervisor call	SVC	0A	N	RR	i_1	i_1	15.0
Test and set (cc)	TS	93	N,F	SI	$d_1(b_1)$	s_1	6.0
Input/Output Instructions							
Start I/O (pi,cc)	SIO	9C	N	SI	$d_1(b_1)$	s_1	IPC paper peripheral Min Max 15 59 IPC communication 15.6 IDA Min Max 10.2 12.6

							Selector Min Max 25.2 26.4 Multiplexer Min Max 16.8 18.6	
Diagnostic Instructions								
Diagnose (pi)†	DIAG	83	N,C2*	SI	$d_1(b_1),i_2$	s_1,i_2	i_2=00 i_2=01 i_2=02	22.8 42.0 48.0
Soft-scope forward scan (pi)†	SSFS	A2	N	RS	(bit pattern)	(bit pattern)	7.2††	
Soft-scope reverse scan (pi)†	SSRS	A3	N	RS	(bit pattern)	(bit pattern)	7.2††	
Interval Timer Instruction								
Service timer register (pi)	STR	03	N	RR	r_1,r_2	r_1,r_2	6.0+0.6t3	

Instructions by Application (cont)

LEGEND:

cc	=	condition code
pi	=	privileged instruction
em	=	extended mnemonics
C1	=	instructions used only in 9200/9300 compatibility mode
C2	=	instructions used only in 360/20 compatibility mode
C3	=	instructions used in 9200/9300 or 360/20 compatibility modes
F	=	instructions added as features
N	=	instructions used in 90/30 native mode

NOTES:

*Operation exception.

**1 specification in operand 1 specifies length of both operands.

***Five cycles per half word: 3.0 microseconds per half word.

†These instructions are not resident at all times.

††This execution time is variable.

Legend for Instruction Execution Time

Symbol	Description
a	1 if overflow adjustment is necessary; otherwise 0
b	Number of first operand bytes processed
ce	Number of digit shifts required to equalize the characteristics
d1	Number of zero addresses in switch list
d2	1 if initial r odd general register has nonzero value; otherwise 0
d3	1 if sentinel found; otherwise 0
d4	Number of task control blocks scrutinized
d5	Number of linked task control blocks scrutinized
d6	1 when exclusive search is specified; otherwise 0
d7	1 when match is found; otherwise, 0

Legend for Instruction Execution Time (cont)

Symbol	Description
d8	Number of control blocks with absolute wait bits set
d9	Number of control blocks with wait bits set and ICOR bit clear
d10	1 if ICOR = 1; otherwise 0
d11	1 if ICOR = 0 and no wait bits set; otherwise 0
e	Execution time of subject instruction
gr	Number of general registers loaded or stored
n	Number of bytes in first operand (for instructions with a single field length)
n1	Number of operand 1 bytes
n2	Number of operand 2 bytes
n3	Number of field separator characters in pattern

n4	Number of digit select or significance starter characters in pattern
n6	Number of significant digits detected when significance indicator is not set before digit is examined
n7	Lowest number of bytes specified by L1 or L2
n8	0 if L1 $\leqslant$ L2 (number of bytes in L1 exceeds L2)
nrr	1 if subject instruction of execute instruction is not RR type; otherwise 0
p	Number of 4-place shifts
p1	Number of digit shifts required to prenormalize operand 1
p2	Number of digit shifts required to prenormalize operand 2
pn	1 if the result requires post-normalization; otherwise 0
pr	Number of digit shifts required for post-normalized result
q	Number of 1-place shifts

Legend for Instruction Execution Time (cont)

Symbol	Description
r	1 if r1 ≠ 0; otherwise 0
rn	1 if result (product or quotient) is negative; otherwise 0
rp	1 if recomplementing without post-normalization is required; otherwise 0
s	1 if branch is successful; otherwise 0
s1	1 if sign of op1 is negative; otherwise 0
s2	1 if sign of op2 is negative; otherwise 0
s3	1 if sum of first and third operand equal to comparand; otherwise 0
s4	1 if result is greater than one word (8 decimal digits); otherwise 0
s5	1 if signs of op1 and op2 are the same; otherwise 0
s6	1 if signs of op1 and op2 are different; otherwise 0

s8	1 if sentinel detected; otherwise 0
t1	1 if result is recomplemented; otherwise 0
t2	1 if n2 > n1; otherwise 0
t3	1 if timer stored; otherwise 0
t4	1 if one operand address is even and other is odd; otherwise 0
w	Number of control storage words loaded
w1	Number of channel status words
y	0 for byte count = 0
y1	1 for byte count ≠ 0
z	Number of half words in sum

Mnemonic	Instruction Name	Machine Code	Byte Length	Source Code Format	
				Explicit	Implicit
A	Add	5A	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
AD	Add Normalized, Long	6A	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
ADR	Add Normalized, Long	2A	2	r_1,r_2	r_1,r_2
AE	Add Normalized, Short	7A	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
AER	Add Normalized, Short	3A	2	r_1,r_2	r_1,r_2
AH	Add Half Word	4A	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
AI	Add Immediate	9A	4	$d_1(b_1),i_2$	s_1,i_2
AL	Add Logical	5E	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$
ALR	Add Logical	1E	2	r_1,r	r_1,r_2
AP	Add Decimal	FA	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$
AR	Add	1A	2	r_1,r_2	r_1,r_2
AU	Add Unnormalized, Short	7E	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *

AUR	Add Unnormalized, Short	3E	2	r_1,r_2	r_1,r_2
AW	Add Unnormalized, Long	6E	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
AWR	Add Unnormalized, Long	2E	2	r_1,r_2	r_1,r_2
BAL	Branch and Link	45	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
BALR	Branch and Link	05	2	r_1,r_2	r_1,r_2
BAS	Branch and Store	4D	4	{ compatibility	
BASR	Branch and Store	0D	2	{ mode only	
BC	Branch on Condition	47	4	$i,d_2(x_2,b_2)$	$i,s_2(x_2)$
BCR	Branch on Condition	07	2	i,r_2	i,r_2
BCT	Branch on Count	46	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
BCTR	Branch on Count	06	2	r_1,r_2	r_1,r_2
BXH	Branch on Index High	86	4	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2
BXLE	Branch on Index Low or Equal	87	4	$r_1,r_3,d_2(b_2)$	r_1,r_3,s_2
C	Compare Algebraic	59	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$
CD	Compare, Long	69	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
CDR	Compare, Long	29	2	r_1,r_2	r_1,r_2
CE	Compare, Short	79	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *

Mnemonic	Instruction Name	Machine Code	Byte Length	Source Code Format	
				Explicit	Implicit
CER	Compare, Short	39	2	r_1,r_2	r_1,r_2
CH	Compare Half Word	49	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
CL	Compare Logical	55	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
CLC	Compare Logical	D5	6	$d_1,(l,b_1),d_2(b_2)$	$s_1(l),s_2$
CLI	Compare Logical Immediate	95	4	$d_1(b_1),i_2$	s_1,i_2
CLR	Compare Logical	15	2	r_1,r_2	r_1,r_2
CP	Compare Decimal	F9	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$
CR	Compare Algebraic	19	2	r_1,r_2	r_1,r_2
CVB	Convert to Binary	4F	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
CVD	Convert to Decimal	4E	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
D	Divide	5D	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
DD	Divide, Long	6D	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
DDR	Divide, Long	2D	2	r_1,r_2	r_1,r_2

DE	Divide, Short	7D	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
DER	Divide, Short	3D	2	r_1,r_2	r_1,r_2
DP	Divide Decimal	FD	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),,s_2(l_2)$
DR	Divide	1D	2	r_1,r_2	r_1,r_2
ED	Edit	DE	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
EDMK	Edit and Mark	DF	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
EX	Execute	44	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
HDR	Halve, Long	24	2	r_1,r_2	r_1,r_2
HER	Halve, Short	34	2	r_1,r_2	r_1,r_2
HPR	Halt and Proceed	99	4	(Privileged)	(Privileged)
IC	Insert Character	43	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
ISK	Insert Storage Key	09	2	(Privileged)	(Privileged)
L	Load	58	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
LA	Load Address	41	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
LCDR	Load Complement, Long	23	2	r_1,r_2	r_1,r_2
LCER	Load Complement, Short	33	2	r_1,r_2	r_1,r_2
LCR	Load Complement	13	2	r_1,r_2	r_1,r_2

Mnemonic	Instruction Name	Machine Code	Byte Length	Source Code Format	
				Explicit	Implicit
LCS	Load Control Storage	B1	4	(Privileged)	(Privileged)
LD	Load, Long	68	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
LDR	Load, Long	28	2	r_1,r_2	r_1,r_2
LE	Load, Short	78	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
LER	Load, Short	38	2	r_1,r_2	r_1,r_2
LH	Load Half Word	48	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
LM	Load Multiple	98	4	$r_1,r_3,d_2(b_2)$ *	r_1,r_3,s_2 *
LNDR	Load Negative, Long	21	2	r_1,r_2	r_1,r_2
LNER	Load Negative, Short	31	2	r_1,r_2	r_1,r_2
LNR	Load Negative	11	2	r_1,r_2	r_1,r_2
LPDR	Load Positive, Long	20	2	r_1,r_2	r_1,r_2
LPER	Load Positive, Short	30	2	r_1,r_2	r_1,r_2
LPR	Load Positive	10	2	r_1,r_2	r_1,r_2

LPSW	Load Program Status Word	82	4	(Privileged)	(Privileged)
LR	Load	18	2	r_1,r_2	r_1,r_2
LTDR	Load and Test, Long	22	2	r_1,r_2	r_1,r_2
LTER	Load and Test, Short	32	2	r_1,r_2	r_1,r_2
LTR	Load and Test	12	2	r_1,r_2	r_1,r_2
M	Multiply	5C	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
MD	Multiply, Long	6C	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
MDR	Multiply, Long	2C	2	r_1,r_2	r_1,r_2
ME	Multiply, Short	7C	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
MER	Multiply, Short	3C	2	r_1,r_2	r_1,r_2
MH	Multiply Half Word	4C	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
MP	Multiple Decimal	FC	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$
MR	Multiply	1C	2	r_1,r_2	r_1,r_2
MVC	Move Characters	D2	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l), s_2$
MVI	Move Immediate	92	4	$d_1(b_1),i_2$	s_1,i_2
MVN	Move Numerics	D1	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
MVO	Move With Offset	F1	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$

Instructions by Mnemonic Code (cont)

Mnemonic	Instruction Name	Machine Code	Byte Length	Source Code Format	
				Explicit	Implicit
MVZ	Move Zones	D3	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
N	AND Logical	54	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
NC	AND Logical	D4	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
NI	AND Logical Immediate	94	4	$d_1(b_1),i_2$	s_1,i_2
NR	AND Logical	14	2	r_1,r_2	r_1,r_2
O	OR Logical	56	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
OC	OR Logical	D6	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
OI	OR Logical Immediate	96	4	$d_1(b_1),i_2$	s_1,i_2
OR	OR Logical	16	2	r_1,r_2	r_1,r_2
PACK	Pack	F2	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$
S	Subtract	5B	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
SD	Subtract Normalized, Long	6B	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
SDR	Subtract Normalized, Long	2B	2	r_1,r_2	r_1,r_2

SE	Subtract Normalized, Short	7B	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
SER	Subtract Normalized, Short	3B	2	r_1,r_2	r_1,r_2
SH	Subtract Half Word	4B	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
SIO	Start I/O	9C	4	(Privileged)	(Privileged)
SL	Subtract Logical	5F	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
SLA	Shift Left Single Algebraic	8B	4	$r_1,d_2(b_2)$	r_1,s_2
SLDA	Shift Left Double Algebraic	8F	4	$r_1,d_2(b_2)$	r_1,s_2
SLDL	Shift Left Double Logical	8D	4	$r_1,d_2(b_2)$	r_1,s_2
SLL	Shift Left Single Logical	89	4	$r_1,d_2(b_2)$	r_1,s_2
SLM	Supervisor Load Multiple	B8	4	(Privileged)	(Privileged)
SLR	Subtract Logical	1F	2	r_1,r_2	r_1,r_2
SP	Subtract Decimal	FB	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$
SPM	Set Program Mask	04	2	r_1	r_1
SR	Subtract	1B	2	r_1,r_2	r_1,r_2
SRA	Shift Right Single Algebraic	8A	4	$r_1,d_2(b_2)$	r_1,s_2
SRDA	Shift Right Double Algebraic	8E	4	$r_1,d_2(b_2)$	r_1,s_2
SRDL	Shift Right Double Logical	8C	4	$r_1,d_2(b_2)$	r_1,s_2

Mnemonic	Instruction Name	Machine Code	Byte Length	Source Code Format	
				Explicit	Implicit
SRL	Shift Right Single Logical	88	4	$r_1,d_2(b_2)$	r_1,s_2
SSK	Set System Key	08	2	(Privileged)	(Privileged)
SSM	Set System Mask	80	4	(Privileged)	(Privileged)
SSTM	Supervisor Store Multiple	B0	4	(Privileged)	(Privileged)
ST	Store	50	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
STC	Store Character	42	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
STD	Store Long	60	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
STE	Store Short	70	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
STH	Store Half Word	40	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
STM	Store Multiple	90	4	$r_1,r_3,d_2(b_2)$ *	r_1,r_3,s_2 *
STR	Service Timer Register	03	2	(Privileged)	(Privileged)
SU	Subtract Unnormalized, Short	7F	4	$r_1,d_2(x_2,b_2)$ *	$r_1,s_2(x_2)$ *
SUR	Subtract Unnormalized, Short	3F	2	r_1,r_2	r_1,r_2

SVC	Supervisor Call	0A	2	i	i
SW	Subtract Unnormalized, Long	6F	4	$r_1,d_2(x_2,b_2)$ **	$r_1,s_2(x_2)$ **
SWR	Subtract Unnormalized, Long	2F	2	r_1,r_2	r_1,r_2
TM	Test Under Mask	91	4	$d_1(b_1),i_2$	s_1,i_2
TR	Translate	DC	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
TRT	Translate and Test	DD	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
TS	Test and Set	93	4	$d_1(b_1)$	s_1
UNPK	Unpack	F3	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$
X	Exclusive OR	57	4	$r_1,d_2(x_2,b_2)$	$r_1,s_2(x_2)$
XC	Exclusive OR	D7	6	$d_1(l,b_1),d_2(b_2)$	$s_1(l),s_2$
XI	Exclusive OR, Immediate	97	4	$d_1(b_1),i_2$	s_1,i_2
XR	Exclusive OR	17	2	r_1,r_2	r_1,r_2
ZAP	Zero and Add Decimal	F8	6	$d_1(l_1,b_1),d_2(l_2,b_2)$	$s_1(l_1),s_2(l_2)$

*Operand 2 must be aligned on a full-word boundary.

**Operand 2 must be aligned on a double-word boundary.

Instruction Name		Machine Code	Mnemonic
Add	(Native and 360/20 modes)	1A	(C)AR
Add		5A	A
Add Decimal		FA	(C)AP
Add Half Word	(Native and 360/20 modes)	4A	(C)AH
Add Half Word	(9200/9300 mode only)	(AA)	(C)AH
Add Immediate		9A	AI
Add Immediate	(9200/9300 mode only)	(A6)	(C)AI
Add Logical	(9200/9300 mode only)	1E	(F)ALR

Add Logical	5E	(F)AL
Add Normalized, Long	2A	(F)ADR
Add Normalized, Long	6A	(F)AD
Add Normalized, Short	3A	(F)AER
Add Normalized, Short	7A	(F)AE
Add Unnormalized, Long	2E	(F)AWR
Add Unnormalized, Long	6E	(F)AW
Add Unnormalized, Short	3E	(F)AUR
Add Unnormalized, Short	7E	(F)AU
AND	14	NR
AND	54	N

Instruction Name		Machine Code	Mnemonic
AND		94	(C)NI
AND	(Native and 9200/9300 modes)	D4	(C)NC
Branch and Link		05	BALR
Branch and Link	(Native and 9200/9300 modes)	45	(C)BAL
Branch and Store	(360/20 mode only)	4D	(C)BAS
Branch and Store	(360/20 mode only)	0D	(C)BASR
Branch on Condition	(Native and 360/20 modes)	07	(C)BCR)

Branch on Condition	47	(C)BC
Branch on Count	06	BCTR
Branch on Count	46	BCT
Branch on Index High	86	(F)BXH
Branch on Index Low or Equal	87	(F)BXLE
Compare	19	CR
Compare	59	C
Compare Decimal	F9	(C)CP
Compare Half Word	49	(C)CH
Compare Logical	15	CLR
Compare Logical	55	CL
Compare Logical	95	(C)CLI
Compare Logical	D5	(C)CLC

Instruction Name	Machine Code	Mnemonic
Compare, Long	29	(F)CDR
Compare, Long	69	(F)CD
Compare, Short	39	(F)CER
Compare, Short	79	(F)CE
Convert to Binary	4F	CVB
Convert to Decimal	4E	CVD
Diagnose — Privileged	83	DIAG

Divide	1D	(F)DR
Divide	5D	D
Divide Decimal	FD	(C)DP
Divide, Long	2D	(F)DDR
Divide, Long	6D	(F)DD
Divide, Short	3D	(F)DER
Divide, Short	7D	(F)DE
Edit	DE	(C)ED
Edit and Mark	DF	(F)EDMK
Exclusive OR	17	XR
Exclusive OR	57	X
Exclusive OR	97	XI

Instructions by Instruction Name (cont)

Instruction Name	Machine Code	Mnemonic
Exclusive OR	D7	XC
Execute	44	EX
Halt and Proceed — Privileged	99	HPR
Halve, Long	24	(F)HDR
Halve, Short	34	(F)HER
Insert Character	43	IC
Insert Storage Key — Privileged	09	(F)ISK
Load	18	LR
Load	58	L

Load Address	41	LA
Load and Test	12	LTR
Load and Test, Long	22	(F)LTDR
Load and Test, Short	32	(F)LTER
Load Complement	13	(F)LCR
Load Complement, Long	23	(F)LCDR
Load Complement, Short	33	(F)LCER
Load Control Storage — Privileged	B1	LCS
Load Half Word	48	(C)LH
Load, Long	28	(F)LDR
Load, Long	68	(F)LD

Instruction Name	Machine Code	Mnemonic
Load Multiple	98	LM
Load Negative	11	(F)LNR
Load Negative, Long	21	(F)LNDR
Load Negative, Short	31	(F)LNER
Load Positive	10	(F)LPR
Load Positive, Long	20	(F)LPDR
Load Positive, Short	30	(F)LPER
Load PSW — Privileged	82	LPSW

Load, Short		38	(F)LER
Load, Short		78	(F)LE
Move		92	(C)MVI
Move		D2	(C)MVC
Move Numerics		D1	(C)MVN
Move With Offset		F1	(C)MVO
Move Zones	(Native and 360/20 modes)	D3	(C)MVZ
Multiply		1C	(F)MR
Multiply		5C	M
Multiply Decimal		FC	(C)MP
Multiply Half Word		4C	(F)MH

Instructions by Mnemonic Code (cont)

Instruction Name	Machine Code	Mnemonic
Multiply, Long	2C	(F)MDR
Multiply, Long	6C	(F)MD
Multiply, Short	3C	(F)MER
Multiply, Short	7C	(F)ME
OR	16	OR
OR	56	O
OR	96	(C)OI
OR (Native and 9200/9300 modes)	D6	(C)OC

Pack	F2	(C)PACK
Service Timer Register — Privileged	03	STR
Set Program Mask	04	SPM
Set Storage Key — Privileged	08	(F)SSK
Set System Mask — Privileged	80	SSM
Shift Left Double	8F	(F)SLDA
Shift Left Double Logical	8D	(F)SLDL
Shift Left Single	8B	(F)SLA
Shift Left Single Logical	89	SLL
Shift Right Double	8E	(F)SRDA
Shift Right Double Logical	8C	(F)SRDL

Instructions by Instruction Name (cont)

Instruction Name	Machine Code	Mnemonic
Shift Right Single	8A	(F)SRA
Shift Right Single Logical	88	SRL
Softscope Forward Scan — Privileged	A2	SSFS
Softscope Reverse Scan — Privileged	A3	SSRS
Start I/O — Privileged	9C	SIO
Store	50	ST
Store Character	42	STC
Store Half Word	40	(C)STH
Store, Long	60	(F)STD

Store Multiple		90	STM
Store, Short		70	(F)STE
Subtract	(Native and 360/20 modes)	1B	(C)SR
Subtract		5B	S
Subtract Decimal		FB	(C)SP
Subtract Half Word	(Native and 360/20 modes)	4B	(C)SH
Subtract Half Word	(9200/9300 mode only)	(AB)	(C)SH
Subtract Logical		1F	(F)SLR
Subtract Logical		5F	(F)SL
Subtract Normalized, Long		2B	(F)SDR
Subtract Normalized, Long		6B	(F)SD

Instructions by Instruction Name (cont)

Instruction Name	Machine Code	Mnemonic
Subtract Normalized, Short	3B	(F)SER
Subtract Normalized, Short	7B	(F)SE
Subtract Unnormalized, Long	2F	(F)SWR
Subtract Unnormalized, Long	6F	(F)SW
Subtract Unnormalized, Short	3F	(F)SUR
Subtract Unnormalized, Short	7F	(F)SU
Supervisor Call	0A	SVC
Supervisor Load Multiple — Privileged	B8	SLM

Supervisor Store Multiple — Privileged	B0	SSTM
Test and Set	93	(F)TS
Test Under Mask	91	(C)TM
Translate	DC	(C)TR
Translate and Test	DD	TRT
Unpack	F3	(C)UNPK
Zero and Add	F8	(C)ZAP

NOTES:

1. Tag symbol F before mnemonic indicates instructions that are added as features.
2. Tag symbol C before mnemonic indicates instruction available in native mode and in 9200/9300 and 360/20 compatibility modes, unless indicated otherwise by notes. The absence of C indicates instruction available in native mode only. Machine codes in parenthesis execute in 9200/9300 compatibility mode only.

Machine Code	Mnemonic	Instruction Name
03	STR	Service Timer Register — Privileged
04	SPM	Set Program Mask
05	BALR	Branch and Link
06	BCTR	Branch on Count
07	(C)BCR	Branch on Condition (Native and 360/20 modes)
08	(F)SSK	Set Storage Key — Privileged
09	(F)ISK	Insert Storage Key — Privileged
0A	SVC	Supervisor Call

0D	(C)BASR	Branch and Store (360/20 mode only)
10	(F)LPR	Load Positive
11	(F)LNR	Load Negative
12	LTR	Load and Test
13	(F)LCR	Load Complement
14	NR	AND
15	CLR	Compare Logical
16	OR	OR
17	XR	Exclusive OR
18	LR	Load
19	CR	Compare

Machine Code	Mnemonic	Instruction Name
1A	(C)AR	Add (Native and 360/20 modes)
1B	(C)SR	Subtract (Native and 360/20 modes)
1C	(F)MR	Multiply
1D	(F)DR	Divide
1E	(F)ALR	Add Logical
1F	(F)SLR	Subtract Logical
20	(F)LPDR	Load Positive, Long
21	(F)LNDR	Load Negative, Long

22	(F)LTDR	Load And Test, Long
23	(F)LCDR	Load Complement, Long
24	(F)HDR	Halve, Long
28	(F)LDR	Load, Long
29	(F)CDR	Compare, Long
2A	(F)ADR	Add Normalized, Long
2B	(F)SDR	Subtract Normalized, Long
2C	(F)MDR	Multiply, Long
2D	(F)DDR	Divide, Long
2E	(F)AWR	Add Unnormalized, Long

Machine Code	Mnemonic	Instruction Name
2F	(F)SWR	Subtract Unnormalized, Long
30	(F)LPER	Load Positive, Short
31	(F)LNER	Load Negative, Short
32	(F)LTER	Load And Test, Short
33	(F)LCER	Load Complement, Short
34	(F)HER	Halve, Short
38	(F)LER	Load, Short
39	(F)CER	Compare, Short

3A	(F)AER	Add Normalized, Short
3B	(F)SER	Subtract Normalized, Short
3C	(F)MER	Multiply, Short
3D	(F)DER	Divide, Short
3E	(F)AUR	Add Unnormalized, Short
3F	(F)SUR	Subtract Unnormalized, Short
40	(C)STH	Store Half Word
41	LA	Load Address
42	STC	Store Character

Machine Code	Mnemonic	Instruction Name
43	IC	Insert Character
44	EX	Execute
45	(C)BAL	Branch and Link (Native and 9200/9300 modes)
46	BCT	Branch on Count
47	(C)BC	Branch on Condition
48	(C)LH	Load Half Word
49	(C)CH	Compare Half Word
4A	(C)AH	Add Half Word (Native and 360/20 modes)

4B	(C)SH	Subtract Half Word (Native and 360/20 modes)
4C	(F)MH	Multiply Half Word
4D	(C)BAS	Branch and Store (360/20 mode only)
4E	CVD	Convert to Decimal
4F	CVB	Convert to Binary
50	ST	Store
54	N	AND
55	CL	Compare Logical
56	O	OR

Instructions by Machine Code (cont)

Machine Code	Mnemonic	Instruction Name
57	X	Exclusive OR
58	L	Load
59	C	Compare
5A	A	Add
5B	S	Subtract
5C	M	Multiply
5D	D	Divide
5E	(F)AL	Add Logical

5F	(F)SL	Subtract Logical
60	(F)STD	Store, Long
68	(F)LD	Load, Long
69	(F)CD	Compare, Long
6A	(F)AD	Add Normalized, Long
6B	(F)SD	Subtract Normalized, Long
6C	(F)MD	Multiply, Long
6D	(F)DD	Divide, Long
6E	(F)AW	Add Unnormalized, Long

Instructions by Machine Code (cont)

Machine Code	Mnemonic	Instruction Name
6F	(F)SW	Subtract Unnormalized, Long
70	(F)STE	Store, Short
78	(F)LE	Load, Short
79	(F)CE	Compare, Short
7A	(F)AE	Add Normalized, Short
7B	(F)SE	Subtract Normalized, Short
7C	(F)ME	Multiply, Short
7D	(F)DE	Divide, Short

7E	(F)AU	Add Unnormalized, Short
7F	(F)SU	Subtract Unnormalized, Short
80	SSM	Set System Mask — Privileged
82	LPSW	Load PSW — Privileged
83	DIAG	Diagnose — Privileged
86	(F)BXH	Branch on Index High
87	(F)BXLE	Branch on Index Low or Equal
88	SRL	Shift Right Single Logical
89	SLL	Shift Left Single Logical
8A	(F)SRA	Shift Right Single
8B	(F)SLA	Shift Left Single

Instructions by Machine Code (cont)

Machine Code	Mnemonic	Instruction Name
8C	(F)SRDL	Shift Right Double Logical
8D	(F)SLDL	Shift Left Double Logical
8E	(F)SRDA	Shift Right Double
8F	(F)SLDA	Shift Left Double
90	STM	Store Multiple
91	(C)TM	Test Under Mask
92	(C)MVI	Move Immediate
93	(F)TS	Test and Set

94	(C)NI	AND
95	(C)CLI	Compare Logical
96	(C)OI	OR
97	XI	Exclusive OR
98	LM	Load Multiple
99	HPR	Halt and Proceed — Privileged
9A	AI	Add Immediate
9C	SIO	Start I/O — Privileged
A2	SSFS	Softscope Forward Scan — Privileged
A3	SSRS	Softscope Reverse Scan — Privileged
(A6)	(C)AI	Add Immediate (9200/9300 mode only)

Instructions by Machine Code (cont)

Machine Code	Mnemonic	Instruction Name
(AA)	(C)AH	Add Half Word (9200/9300 mode only)
(AB)	(C)SH	Subtract Half Word (9200/9300 mode only)
B0	SSTM	Supervisor Store Multiple — Privileged
B1	LCS	Load Control Storage — Privileged
B8	SLM	Supervisor Load Multiple — Privileged
D1	(C)MVN	Move Numerics
D2	(C)MVC	Move
D3	(C)MVZ	Move Zones (Native and 360/20 modes)
D4	(C)NC	AND (Native and 9200/9300 modes)

D5	(C)CLC	Compare Logical
D6	(C)OC	OR (Native and 9200/9300 modes)
D7	XC	Exclusive OR
DC	(C)TR	Translate
DD	TRT	Translate and Test
DE	(C)ED	Edit
DF	(F)EDMK	Edit and Mark
F1	(C)MVO	Move With Offset
F2	(C)PACK	Pack
F3	(C)UNPK	Unpack
F8	(C)ZAP	Zero and Add

Machine Code	Mnemonic	Instruction Name
F9	(C)CP	Compare Decimal
FA	(C)AP	Add Decimal
FB	(C)SP	Subtract Decimal
FC	(C)MP	Multiply Decimal
FD	(C)DP	Divide Decimal

NOTES:

1. Tag symbol F before mnemonic indicates instructions that are added as features.
2. Tag symbol C before mnemonic indicates instruction available in native mode and in 9200/9300 and 360/20 compatibility modes, unless indicated otherwise by notes. The absence of C indicates instruction available in native mode only.
3. Machine codes in parentheses execute in 9200/9300 compatibility mode only.

RR-Type Instructions		BC Equivalent		RX-Type Instructions		BC Equivalent		Function
Mnemonic Code	Hexadecimal Operation Code m_1	Mnemonic Code	Explicit Form	Mnemonic Code	Hexadecimal Operation Code m_1	Mnemonic Code	Explicit Form	
BR	07 F	BCR	$15,r_2$	–	–	–	–	Branch unconditionally
NOPR	07 0	BCR	$0,r_2$	–	–	–	–	No operation
–	–	–	–	B	47 F	BC	$15,d_2(x_2,b_2)$	Branch unconditionally
–	–	–	–	NOP	47 0	BC	$0,d_2(x_2,b_2)$	No operation
Used After Comparison Instructions								
BHR	07 2	BC	$2,r_2$	BH	47 2	BC	$2,d_2(x_2,b_2)$	Branch if high
BLR	07 4	BC	$4,r_2$	BL	47 4	BC	$4,d_2(x_2,b_2)$	Branch if low
BER	07 8	BC	$8,r_2$	BE	47 8	BC	$8,d_2(x_2,b_2)$	Branch if equal
BNHR	07 D	BC	$13,r_2$	BNH	47 D	BC	$13,d_2(x_2,b_2)$	Branch if not high
BNLR	07 B	BC	$11,r_2$	BNL	47 B	BC	$11,d_2(x_2,b_2)$	Branch if not low
BNER	07 7	BC	$7,r_2$	BNE	47 7	BC	$7,d_2(x_2,b_2)$	Branch if not equal

Used After Test-Under-Mask Instructions								
BOR	07 1	BC	1,r_2	BO	47 1	BC	1,$d_2(x_2,b_2)$	Branch if ones
BZR	07 8	BC	8,r_2	BZ	47 8	BC	8,$d_2(x_2,b_2)$	Branch if zeros
BMR	07 4	BC	4,r_2	BM	47 4	BC	4,$d_2(x_2,b_2)$	Branch if mixed
BNOR	07 E	BC	14,r_2	BNO	47 E	BC	14,$d_2(x_2,b_2)$	Branch if not ones
BNZR	07 7	BC	7,r_2	BNZ	47 7	BC	7,$d_2(x_2,b_2)$	Branch if not zeros
BNMR	07 B	BC	11,r_2	BNH	47 B	BC	11,$d_2(x_2,b_2)$	Branch if not mixed
Used After Arithmetic Instructions								
BOR	07 1	BC	1,r_2	BO	47 1	BC	1,$d_2(x_2,b_2)$	Branch if overflow
BZR	07 8	BC	8,r_2	BZ	47 8	BC	8,$d_2(x_2,b_2)$	Branch if zero
BMR	07 4	BC	4,r_2	BM	47 4	BC	4,$d_2(x_2,b_2)$	Branch if minus
BPR	07 2	BC	2,r_2	BP	47 2	BC	2,$d_2(x_2,b_2)$	Branch if positive
BNOR	07 E	BC	14,r_2	BNO	47 E	BC	14,$d_2(x_2,b_2)$	Branch if not overflow
BNZR	07 7	BC	7,r_2	BNZ	47 7	BC	7,$d_2(x_2,b_2)$	Branch if not zero
BNMR	07 B	BC	11,r_2	BNM	47 B	BC	11,$d_2(x_2,b_2)$	Branch if not minus
BNPR	07 D	BC	13,r_2	BNP	47 D	BC	13,$d_2(x_2,b_2)$	Branch if not positive

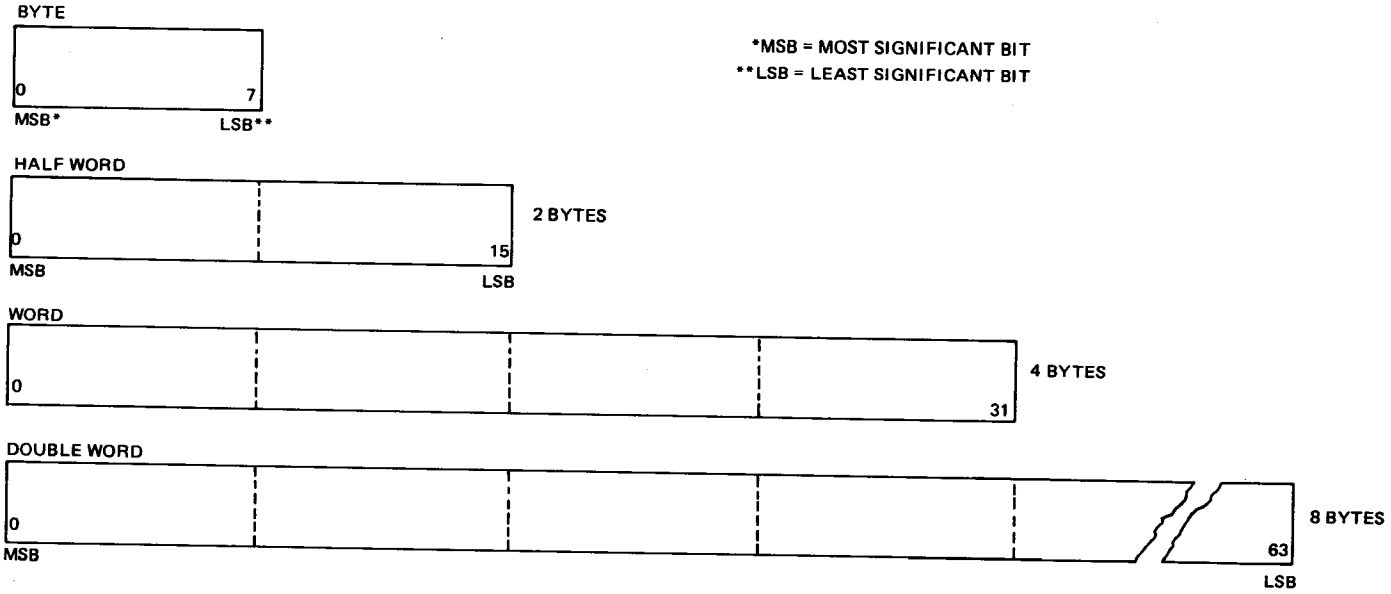
Mask in BC instruction	8	4	2	1
CC in PSW	0	1	2	3

Mask (Operand 1) Character	EBCDIC/ASCII	S Switch Status	Data (Operand 2) Character	Resulting (Operand 1) Character	Resulting S Switch Status
Fill character	Any	Off	Not examined	Remains same	Off
Digit select byte	20/80	On	Nonzero	Digit	On*
		On	Zero	Digit	On*
		Off	Nonzero	Digit	On*
		Off	Zero	Fill character	Off
Significance start byte	21/81	On	Nonzero	Digit	On*
		On	Zero	Digit	On*
		Off	Nonzero	Digit	On*
		Off	Zero	Fill character	On*

Mark (Operand 1) Character	EBCDIC/ASCII	S Switch Status	Data (Operand 2) Character	Resulting (Operand 1) Character	Resulting S Switch Status
Message character	Any except 20/80, 21/81, 22/82	On	Not examined	Message character	On*
		Off	Not examined	Fill character	Off*
Field separator byte	22/82	On	Not examined	Fill character	Off
		Off	Not examined	Fill character	Off

*Sign detection (examined simultaneously with operand 2 digit) affects the S switch as follows:

A plus or minus sign detected as most significant digit causes data exception.
A plus sign detected as a least significant digit causes S switch to be turned off.
A minus sign has no effect on the S switch.


BYTE
0
7
MSB*
LSB**
*MSB = MOST SIGNIFICANT BIT
**LSB = LEAST SIGNIFICANT BIT
HALF WORD
0
15
MSB
LSB
2 BYTES
WORD
0
31
4 BYTES
DOUBLE WORD
0
63
MSB
LSB
8 BYTES

To align boundaries on double-word, full-word, and half-word main storage boundaries, use the following source code statement formats:

LABEL	△OPERATION△	OPERAND	△	COMMENTS
1	10	16		
	DS	0D		DOUBLE WORD
	DS	0F		FULL WORD
	DS	0H		HALF WORD

Fixed-Point Numbers

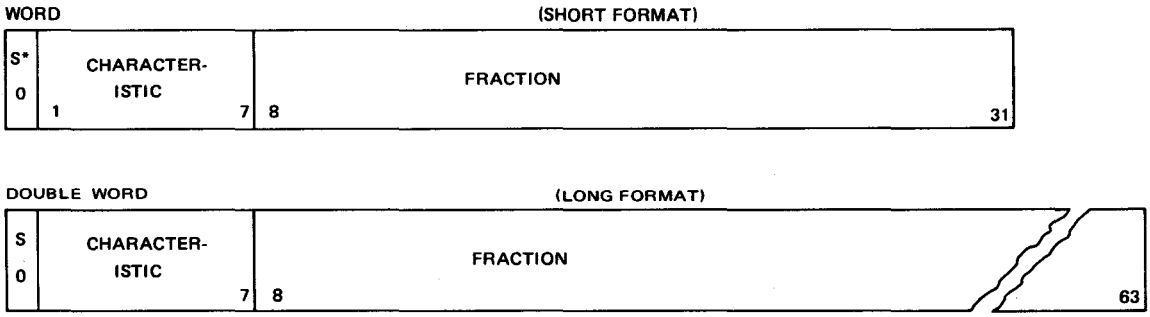
HALF WORD

*S = SIGN BIT

WORD

DOUBLE WORD

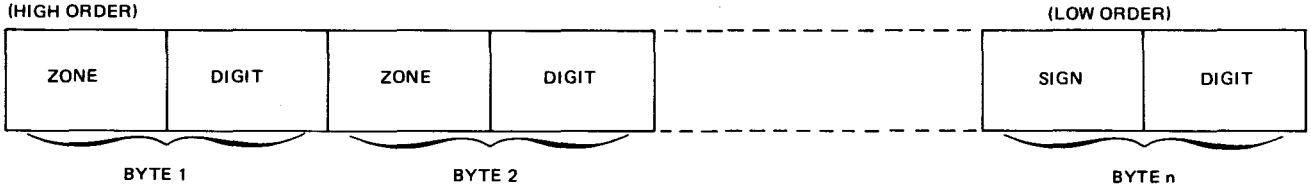
Floating-Point Numbers



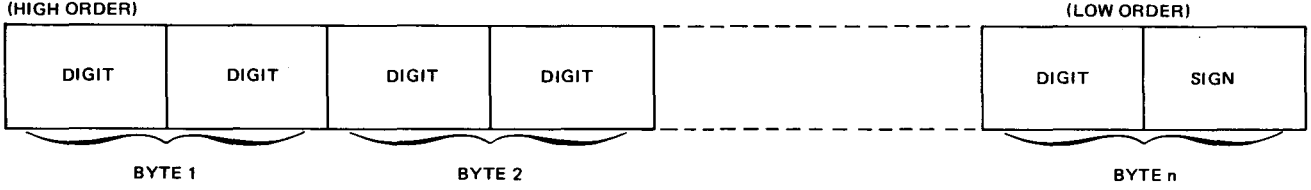
*S = SIGN BIT

Decimal Numbers

UNPACKED NUMBERS

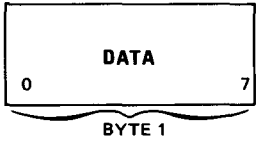


PACKED NUMBERS

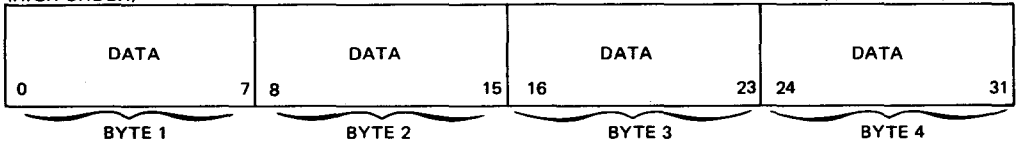


Logical Information

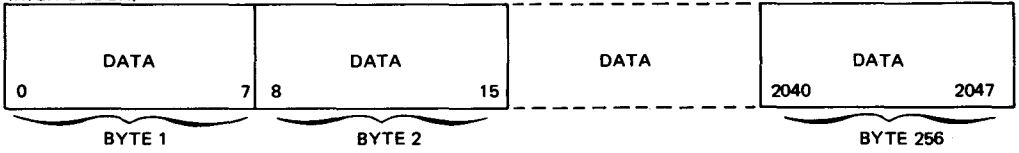
FIXED-LENGTH DATA

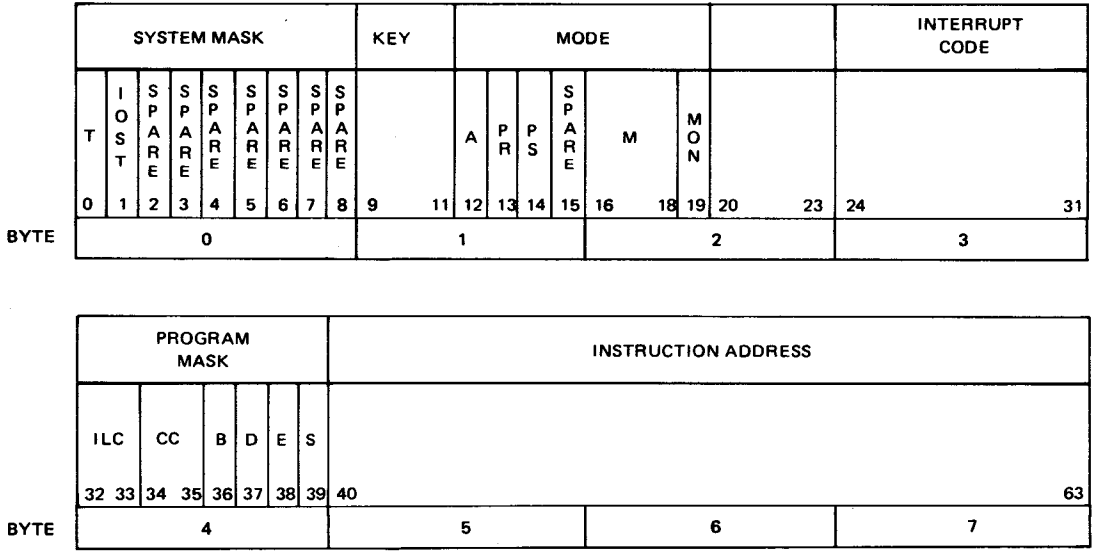


WORD
(HIGH ORDER) (LOW ORDER)



VARIABLE-LENGTH DATA
(HIGH ORDER) (LOW ORDER)




SYSTEM MASK
KEY
MODE
INTERRUPT CODE
T
IOST
SPARE
SPARE
SPARE
SPARE
SPARE
SPARE
SPARE
A
PR
PS
SPARE
M
MON
0
1
9
11
12
13
14
15
16
18
19
20
23
24
31
BYTE
0
1
2
3
PROGRAM MASK
INSTRUCTION ADDRESS
ILC
CC
B
D
E
S
32
33
34
35
36
37
38
39
40
63
BYTE
4
5
6
7

Legend for Program Status Word

Bits	Allocation	Function
0	Timer level interrupt mask	
1	I/O status tabler level interrupt mask (channel 7)	
2–8	Not used, must be zero	
9–11	Key	3-bit code assigning associated block of main storage to one of eight programs
12	ASCII mode	A = 1 (ASCII mode) A = 0 (EBCDIC mode)
13	Problem register mode	PR = 1 (problem registers selected) PR = 0 (supervisor general register selected)

Legend for Program Status Word (cont)

Bits	Allocation	Function
14	Problem mode selection	PS = 1 (problem mode) PS = 0 (supervisor mode)
15	Not used; must be zero	
16–18	Mode	000 = 90/30 native mode 001 = 9200/9300 compatibility mode 010 = IBM 360/20 compatibility mode
19	Monitor mode	MON = 1 (monitor mode) MON = 0 (normal execution)
20–23	Not used; must be zero	
24–31	Interrupt code*	

*Refer to PROGRAM EXCEPTION INTERRUPT CODES and MACHINE CHECK LEVEL INTERRUPT CODES.

32, 33	Instruction length code	00 = instruction suppressed 01 = one half word (RR) 10 = two half words (RX, RS, SI) 11 = three half words (SS)
34, 35	Condition code**	00 = test value is binary 8 (1000) 01 = test value is binary 4 (0100) 10 = test value is binary 2 (0010) 11 = test value is binary 1 (0001)
36 37 38 39	Program mask bits B = fixed-point overflow D = decimal overflow E = exponent underflow S = significance	1 = allowed 0 = inhibited
40–63	Instruction address	At interrupt, contains address of instruction following instruction causing interrupt

**Refer to CONDITION CODE SETTINGS.

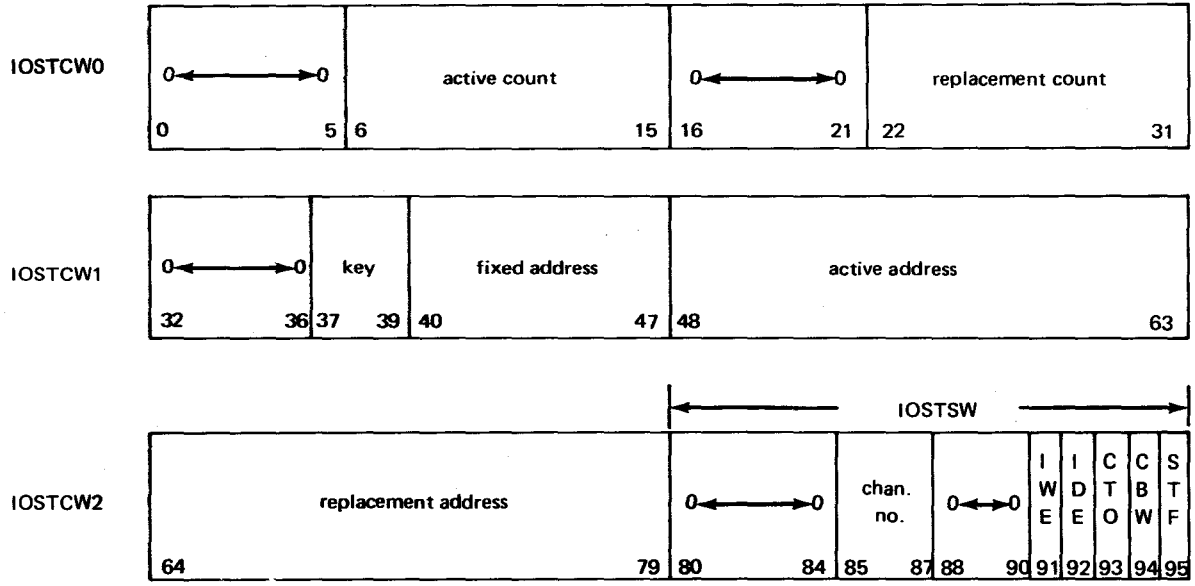
Program Exception Interrupt Codes

Code	Condition
00000001	Operation exception
00000010	Privileged operation exception
00000011	Execute exception
00000100	Protection exception
00000101	Addressing exception
00000110	Specification exception
00000111	Data exception
00001000	Fixed-point overflow exception*
00001001	Fixed-point divide exception
00001010	Decimal overflow exception*
00001011	Decimal divide exception
00001100	Exponent overflow exception
00001101	Exponent underflow exception*
00001110	Significance exception*
00001111	Floating-point divide exception

*Interrupt can be masked

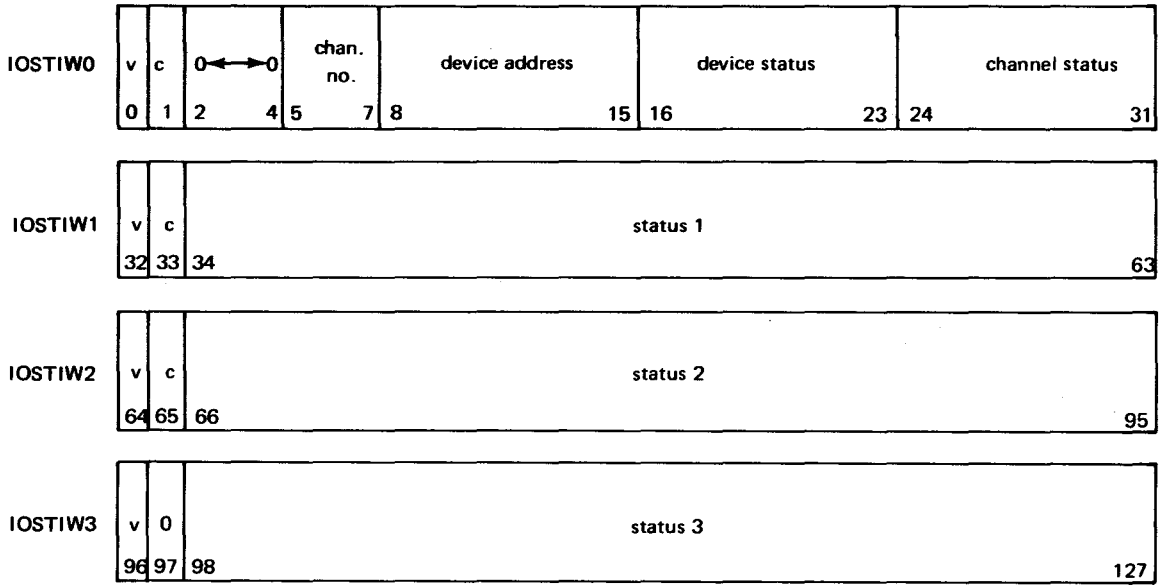
Machine Check Level Interrupt Codes

Code	Condition
Processor Machine Check Class	
11100110	Control storage write bus check
11100111	Storage parity check
11101000	Address check
11101100	Program exception interrupt request
11101111	Processor stall timer
IOST Machine Check Class	
00000101	Addressing or protection exception on accessing IOSTCW
00001000	Address check or storage parity check on accessing IOSTCW
00001111	Processor stall timer

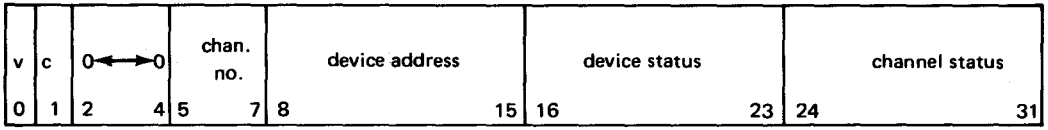

IOSTCW0
0
0
active count
0
0
replacement count
0
5
6
15
16
21
22
31
IOSTCW1
0
0
key
fixed address
active address
32
36
37
39
40
47
48
63
IOSTSW
IOSTCW2
replacement address
0
0
chan. no.
0
0
IWE
IDE
CTO
CBW
STF
64
79
80
84
85
87
88
90
91
92
93
94
95

Bits	Allocation	Function
0–5	Not used; must be zero	
6–15	Active count	Number of words remaining in status word
16–21	Not used; must be zero	
22–31	Replacement count	Replaces active count field when active count is decremented to zero
32–36	Not used; must be zero	
37–39	Key	3-bit storage protection key
40–47	Fixed address	Fixed 8-bit field of status table address
48–63	Active address	Points to most significant byte of next IOSTIW location
64–79	Replacement address	Address written into active portion of address field when active count becomes zero

80–84	Set to zero	
85–87	Channel number	Channel being serviced by IOST when error condition occurred
88–90	Not used; must be zero	
91	Interrupt word error	Protection or addressing exception error detected by IOST
92	Interrupt data error	Address or data parity check error detected by IOST
93	Channel time-out	CT0 bit set when selected channel fails to respond to status service request acknowledge (SSRA) signal within specified time.
94	Channel buffer word	Error detected by selected channel
95	Status table full	Status table location when IOSTIW to be stored is full

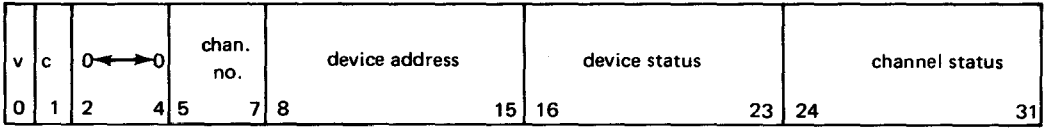

IOSTIW0
v
c
0◄►0
chan. no.
device address
device status
channel status
0
1
2
4
5
7
8
15
16
23
24
31
IOSTIW1
v
c
status 1
32
33
34
63
IOSTIW2
v
c
status 2
64
65
66
95
IOSTIW3
v
0
status 3
96
97
98
127

Multiplexer Channel



Selector Channel

IOSTIW0



IOSTIW1

IOSTIW2

Legend for I/O Status Tabler Interrupt Word Bit

Bits	Allocation	Function
bits 0, 32 64, 96	Validation bit (V)*	V = 0 (before storing IOSTIW in BCSW) V = 1 (after processing status)
bits 1, 33, 65	Continuation bit**	Length of IOSTIW C = 0 (last full word of status presented) C = 1 (more than 1 full word of status is to be presented)
2–4		Not used; must be zero
5–7	Channel number†	Number of channel presenting status
8–15	Device address	Address of device or subsystem active on channel at time status is generated

*Set to zero by multiplexer and selector channels

**Bit 1, set to 1 by selector channel and zero by multiplexer channel; bits 33 and 65, set to 1 and zero, respectively, by selector channel

†Set to 100 (channel 4) or 110 (channel 6) by selector channel and 001 (channel 1) by multiplexer channel

Bits	Allocation		Function
16	Attention		Indicates transition from the stop state to the run state
17	Status modifier		
18	Control unit end	Device status	
19	Busy	(Status byte stored here)	Indicates completion of command initiated by IPC and readiness to accept new command
20	Channel end		
21	Device end		Indicates at least one bit
22	Unit check		Set in sense byte 0
23	Unit exception		
24	*Not used		Set to zero by channel
25	*Incorrect length		
26	*Program check		
27	Invalid address	Channel status	
28	Channel data check		
29	Interface control check		
30	Channel control check		
31	*Buffer terminate		

*Bits 24–26 set to zero by multiplexer channel; bit 31 set to zero by selector channel

Legend for I/O Status Tabler Interrupt Word Bit (cont)

Bits	Allocation	Function
34–44		Set to zero by channel
45–63	Next CCW address	Value of next channel command word address present in internal hardware when status word written
66–79		Set to zero by channel
80–95	Remaining byte count	Value of byte count present in internal channel hardware when status word written
34–63	Status 1	Additional 30 bits of status may be presented by a channel
64–95	Status 2	Additional 30 bits of status may be presented by a channel
96–127	Status 3	Additional 30 bits of status may be presented by a channel

Condition Codes	0	1	2	3
Binary Codes	**1000**	**0100**	**0010**	**0001**
	Fixed-Point and Decimal Instructions			
Add (A, AH, AI, AR, AP)	=0	<0	>0	Overflow
Compare (C, CH, CR, CP)	op1=op2	<op2	>op2	Overflow
Load complement (LCR)	=0	<0	>0	Overflow
Load negative (LNR)	=0	<0	No cc	No cc
Load positive (LPR)	=0	No cc	>0	Overflow
Load and test register (LTR)	=0	<0	>0	No cc
Subtract (S, SH, SR, SP)	=0	<0	>0	Overflow
Shift left (SLA,SLDA)	=0	<0	>0	Overflow
Shift right (SRA,SRDA)	=0	<0	>0	No cc
Zero and add (ZAP)	=0	<0	>0	Overflow

Condition Codes	0	1	2	3
Binary Codes	**1000**	**0100**	**0010**	**0001**
	Floating-Point Instructions			
Add norm, long (AD, ADR) and short (AE, AER)	=0	<0	>0	No cc
Add unnorm, long (AR, AWR) and short (AU, AUR)	=0	<0	>0	No cc
Compare, long (CD, CDR) and short (CE, CER)	op1=op2	<op2	>op2	No cc
Load complement, long and short (LCDR, LCER)	=0	<0	>0	No cc
Load negative, long and short (LNDR, LNER)	=0	<0	No cc	No cc
Load positive, long and short (LPDR, LPER)	=0	No cc	>0	No cc
Load and test, long and short (LTDR, LTER)	=0	<0	>0	No cc
Sub norm, long (SD, SDR) and short (SE, SER)	=0	<0	>0	No cc
Sub unnorm, long (SW, SWR) and short (SU, SUR)	=0	<0	>0	No cc
Logical Instructions				
Add logical (AL, ALR)	=0, no carry*	≠0, no carry	=0, carry	≠0, carry

Compare logical (CL, CLC, CLI, CLR)	op1=op2	<op2	>op2	No cc
Edit (ED) Edit and mark	=0	≠0	>0	No cc
AND (N, NC, NI, NR)	=0	≠0	No cc	No cc
OR (O, OC, OI, OR)	=0	≠0	No cc	No cc
Subtract logical (SL, SLR)	No cc	≠0, no carry	=0, carry	≠0, carry
Test under mask (TM)	=0	=mixed	No cc	=1
Translate and test (TRT)	=0	≠0	Last≠0	No cc
Exclusive OR (X, XC, XI, XR)	=0	≠0	No cc	No cc

*Carry is out of the most significant bit position.

Condition Codes	0	1	2	3
Binary Codes	**1000**	**0100**	**0010**	**0001**
	Status Switching Instructions			
Load control storage (LCS)	=incomplete data transfer	=complete data transfer	No cc	=complete data transfer hash total error
Load program status word (LPSW)	Set=bit positions 34 and 35 of op1			
Set program mask (SPM)	Set=bit positions 2 and 3 of op1			
Supervisor call (SVC)	Set=bit positions 34 and 35 of SVC new PSW			
Test and set (TS)	=0	=1	No cc	No cc

Diagnostic Instructions				
Diagnose (DIAG)	=0	No cc	No cc	No cc
Soft scope forward scan (SSFS)	SYNC detected	No cc	No cc	SYN not detected
Soft scope reverse scan (SSRS)	0=SYNC detected 1st buffer iteration	1=SYNC not detected 1st buffer iteration	2=SYNC detected subsequent iteration	3=SYNC not detected subsequent iteration
Interval Timer Instruction				
Service timer register (STR)	0=ITR contents (ITR pending)	1=timer interrupt request pending, ITC contains overrun count in twos complement (overrides 0)	2=interrupt point not reached, ITR contains residue count of previously loaded value	3=timer interrupt request reset

Bit	Condition Which Sets Bits	Meaning
0	Attention	Indicates transition from stop state to run state
1		Set to zero by reader control
2		Set to zero by reader control
3		Set to zero by reader control
4		Set to zero by reader control
5	Device end	Indicates completion of command initiated by IPC and readiness to accept new command
6	Unit check*	Indicates at least one bit set in sense byte 0
7		Set to zero by reader control

*Refer to conditions listed in STATUS AND SENSE SUMMARIES.
Device status byte corresponds to bits 16–23 of IOSTIW.

Device	Command Byte	Bit Position							
		0	1	2	3	4	5	6	7
0717 Reader	Read control	A	B	X	D	E	F	1	0
	Sense*	X	X	X	X	0	1	0	0

*As a result of sense command, reader control transfers two sense-bytes to main storage.

LEGEND:

A (modifier bit)
0 = normal read operation
1 = diagnostic use only

B (modifier bit)
0 = normal read operation
1 = select read station 2 only, inhibit compare error

D & E (modifier bits)
D = ignored, E = 0 } 80-column read
D = 0, E = 1 } short-column (51 column) read
D = 1, E = 1 } short-column (66 column) read

F (modifier bit)
0 = read in translate mode
1 = read in image mode

Device	Command Byte	Bit Position							
		0	1	2	3	4	5	6	7
0605 Punch	Punch control	A	B	X	X	E	F	R	P
	Sense*	X	X	X	X	0	1	0	0

*Reader control transfers two sense bytes to main storage.

LEGEND:

A (modifier bit)
- 0 = normal punch/read operation
- 1 = diagnostic use only

B (modifier bit)
- 0 = stop on error
- 1 = sort errors and remain in run state

E (modifier bit normally for diagnostic use)
- 0 = normal punch operation
- 1 = select reject stacker, terminate data transfers, and eject the card based on the punch data

LEGEND (cont):

F (modifier bit)
- 0 = normal punch operation
- 1 = punch and/or read in image mode

P & R (modifier bits)

P = 0 and R = 0	Invalid code results in a command reject
P = 0 and R = 1	Read operation specified with no punch operation
P = 1 and R = 0	Punch operation specified with no read operation
P = 1 and R = 1	Punch and read operation specified

Device	Command Byte	Bit Position							
		0	1	2	3	4	5	6	7
UNISCOPE 100 console	Read	A	X	X	X	X	F	1	0
	Write	A	B	C	X	X	F	0	1
	Sense	X	X	X	X	0	1	0	0

LEGEND:

Read command

A (modifier bit)
- 0 = normal read operation
- 1 = diagnostic use only

F (modifier bit)
- 0 = read in translate mode; all data transferred to processor in EBCDIC
- 1 = read in ASCII mode; all data to processor in ASCII code

LEGEND (cont):

Write command

A (modifier bit)
- 0 = normal write operation
- 1 = diagnostic use only

B (modifier bit)
- 0 = keyboard lock at completion of write sequence
- 1 = keyboard unlock at completion of write sequence

C (modifier bit)
- 0 = data transfer to UNISCOPE 100 terminal only
- 1 = data transfer to UNISCOPE 100 terminal and COP

F (modifier bit)
- 0 = write in translate mode; EBCDIC data from processor translated to ASCII
- 1 = write in ASCII mode; all data from processor in ASCII

Device	Command Byte	Bit Position							
		0	1	2	3	4	5	6	7
Printer 0773	Load vertical format buffer	0	1	1	0	0	0	1	1
	Load code buffer	1	1	1	1	1	0	1	1
	Print-advance	A	C	D	E	F	0	0	1
	Advance	A	C	D	E	F	1	1	1
	Load print line buffer*	1	1	1	0	0	0	1	1
	Read print line buffer*	X	X	X	0	0	0	1	0
	Read load code buffer*	X	X	X	0	1	0	1	0
	Read vertical format buffer*	A	X	X	1	0	0	1	0
	Diagnostic*	X	X	X	X	X	1	0	1
	Sense	X	X	X	X	0	1	0	0

*These commands are normally for diagnostic use only.

0773 Printer Command Bit Legend

Detail Forms Advance Bits					
Bit A		Bits			
A=0	A=1	C	D	E	F
Advance 0 line	Filler code*	0	0	0	0
Advance 1 line	Form overflow	0	0	0	1
Advance 2 lines	Program selectable skip codes	0	0	1	0
Advance 3 lines		0	0	1	1
Advance 4 lines		0	1	0	0
Advance 5 lines		0	1	0	1
Advance 6 lines		0	1	1	0
Advance 7 lines	Home paper/end of forms	0	1	1	1

Detail Forms Advance Bits					
Bit A		Bits			
A=0	A=1	C	D	E	F
Advance 8 lines	Filler code*	1	0	0	0
Advance 9 lines	Form overflow	1	0	0	1
Advance 10 lines	Program selectable skip codes	1	0	1	0
Advance 11 lines		1	0	1	1
Advance 12 lines		1	1	0	0
Advance 13 lines		1	1	0	1
Advance 14 lines		1	1	1	0
Advance 15 lines	Home paper/end of forms	1	1	1	1

*This code should not normally be specified in the command.

8416 Disc Subsystem

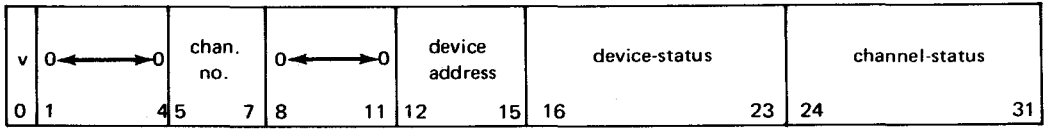
Command	Bit Positions							
	0	1	2	3	4	5	6	7
Format write	0	0	0	0	0	0	0	1
Write data	0	0	0	0	0	1	0	1
Search/read equal	0	0	0	0	1	0	0	1
Search/read high or equal	0	0	0	0	1	1	0	1
Read ID	0	0	0	0	1	1	1	0
Diagnostic	0	0	0	0	0	1	1	0
ECC diagnostic	0	0	0	0	0	1	1	1
Read data	0	0	0	0	0	0	1	0
Seek	0	0	0	0	1	0	0	0
Sense	0	0	0	0	0	1	0	0
ECC sense	0	0	0	0	0	0	1	1

8418 Disc Subsystem

Command	Bit Positions								
	0	1	2	3	4	5	6	7	
Format write	0	0	0	0	0	0	0	1	01
Write data	0	0	0	0	0	1	0	1	05
Search/read equal	0	0	0	0	1	0	0	1	09
Search/read high or equal	0	0	0	0	1	1	0	1	0D
Read ID	0	0	0	0	1	1	1	0	0E
Diagnostic	0	0	0	0	0	1	1	0	06
ECC diagnostic	0	0	0	0	0	1	1	1	07
Read data	0	0	0	0	0	0	1	0	02
Seek	0	0	0	0	1	0	0	0	08
Sense	0	0	0	0	0	1	0	0	04
ECC sense	0	0	0	0	0	0	1	1	03

BOTH TABLES ARE IDENTICAL

IOSTIW Format for IDA and IPC



Bit	Allocation	IDA Function	IPC Function
0	V	Set to zero by IDA when storing an interrupt word	Set to zero by IPC when storing IOSTIW
1–4		Set to zero	Set to zero
5–7	Channel number	Set to binary 3 by IDA	Set to zero by IPC
8–11		Set to zero	Set to zero
13–15	Device address	Binary code indicating device associated with status entry	

IOSTIW Format for IDA and IPC (cont)

Bit	Allocation	IDA Function	IPC Function
12–15	Device address		4-bit field identifying subchannel and/or device to which channel and/or device status applies
16	Attention	One of attached disc drives affected by operator-initiated transition from stop state to run state	Same as IDA
17,18		Set to zero by IDA	
17–20			Set to zero
19	Busy	Addressed device completing previously initiated seek order, programmed offset, or has pending gated attention.	
20	Channel end	IDA able to accept another command	

21	Device end	With channel end = normal end of all commands except the seek (08) Alone = accessor movement complete, disc accessible	Completion of previously initiated command by subsystem and readiness to accept new command
22	Unit check	Problem with addressed disc or IDA	Unusual condition detected at subsystem level
23	Unit exception	ECC check error on ID with read data command; ID of a record not associated with fields to be written, an ECC check error; ECC check or improper number of missing clocks detected in ID field with search read commands	Unusual condition occurred as a result of initiating operation; may not be an error
24–26		Set to zero by IDA	Set to zero
27	Invalid address	Addressing or protection exception when accessing main storage for data or BCW	Addressing or protection exception when accessing main storage except for IOSTIW references

IOSTIW Format for IDA and IPC (cont)

Bit	Allocation	IDA Function	IPC Function
28	Channel data check	Detection of storage parity check on data access to or from main storage	Same as IDA
29		Set to zero by IDA	Set to zero
30	Channel control check	Address check occurred during IPC operation with main storage excluding IOSTIW references	Bit set when storage parity check, addressing exception, or protection exception is detected when accessing portion of BCW; set on address check on BCW
31	Buffer terminate	Set to zero by IDA	IPC performed replacement operation required in data chaining operations

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Format write	0	0	0	0	0	0	0	1
Write data	0	0	0	0	0	1	0	1
Search/read equal	0	0	0	0	1	0	0	1
Search/read HI or equal	0	0	0	0	1	1	0	1
Read ID	0	0	0	0	1	1	1	0
Read data	0	0	0	0	0	0	1	0
Seek	0	0	0	0	1	0	0	0
Sense	0	0	0	0	0	1	0	0
ECC sense	0	0	0	0	0	0	1	1
Diagnostic	0	0	0	0	0	1	1	0
Ecc diagnostic	0	0	0	0	0	1	1	1

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test	X	X	1	1	0	0	0	0
				or				
	X	X	0	0	0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	0	0	0	0	0	1	0	0
Read (diagnostic)	0	0	0	0	0	0	1	0
Load code	1	1	1	1	1	0	1	1
Commands With Form Control								
Print (write)	0	C	D	E	F	0	0	1
Advance-no print (control)	0	C	D	E	F	0	1	1

LEGEND:

Bit position 7 is the least significant bit.

X may be a 1 or 0 bit and is ignored.

C D E F (detail bits)

0 0 0 0 = no advance
0 0 0 1 = advance 1 line
0 0 1 0 = advance 2 lines
0 0 1 1 = advance 3 lines
} under program control

---

0 1 0 0 = paper advanced under control of form control tape to the line corresponding to the same hole combination punched in the tape. Skip may be from 1 to 132 lines.
thru
1 1 1 Y

Y = 0 or 1

---

1 1 1 Y = home form and line selection code
when Y is 0 = 6 LPI
when Y is 1 = 8 LPI

---

1 0 0 1 = form overflow

Command	Bit Positions								
	P	0	1	2	3	4	5	6	7
Test I/O	X X	X X	X 0	1 0	1 0	0 0	0 0	0 0	0 0
Set inhibit status	X	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	X	1	0	0	0	0	0
Sense I/O	0	0	0	0	0	0	1	0	0
Print advance*	X	A	C	D	E	F	0	0	1
Diagnostic write	0	1	1	1	0	0	0	1	1
Advance only*	X	A	C	D	E	F	1	1	1

Command	Bit Positions								
	P	0	1	2	3	4	5	6	7
Load code	0	1	1	1	1	1	0	1	1
Load vertical format*	1	0	1	1	0	0	0	1	1
Fold	0	0	1	0	0	0	0	1	1
Advance print*	X	A	C	D	E	F	1	0	1
Unfold	0	0	0	1	0	0	0	1	1
Inhibit data check	0	0	1	1	1	0	0	1	1
Allow data check	1	0	1	1	1	1	0	1	1
Raise cover*	0	0	1	1	0	1	0	1	1

No operation (No-op)	1	0	0	0	0	0	0	1	1
Read print line buffer	X	X	X	X	0	0	0	1	0
Read load code buffer	X	X	X	X	0	1	0	1	0
Read vertical format buffer*	X	X	X	X	1	0	0	1	0
Check read	1	0	0	0	0	0	1	1	0
Diagnostic gate	0	1	0	1	0	1	0	1	1

*I/O channel cannot initiate these commands when printer is in stop mode, having bit 1 set in sense byte 0 (intervention required). All other commands are sent by the channel and executed normally.

LEGEND:

P is an odd parity bit.
Bit position 7 is the least significant bit.
X may be a 1 or 0 bit and is ignored.
ACDEF detailed advance bits are as follows:

0770 Modified/Detail Bits

Bit A		Bits			
A = 0	A = 1	C	D	E	F
Space 0 line (note 1)	Advance repeat	0	0	0	0
Space 1 line	Skip to code 1	0	0	0	1
Space 2 lines	Skip to code 2	0	0	1	0
Space 3 lines	Skip to code 3	0	0	1	1
Space 4 lines	Skip to code 4	0	1	0	0
Space 5 lines	Skip to code 5	0	1	0	1
Space 6 lines	Skip to code 6	0	1	1	0
Space 7 lines	Skip to code 7	0	1	1	1
Space 8 lines	Skip to code 8	1	0	0	0

Space 9 lines (note 2)	Skip to code 9	1	0	0	1
Space 10 lines	Skip to code A	1	0	1	0
Space 11 lines	Skip to code B	1	0	1	1
Space 12 lines (note 3)	Skip to code C	1	1	0	0
Space 13 lines	Skip to code D	1	1	0	1
Space 14 lines	Skip to code E	1	1	1	0
Space 15 lines	Skip to code F	1	1	1	1

LEGEND:

Code ACDEF = 10000 causes an advance in accordance with the ACDEF detail bits of the last ACDEF not equal to 10000 advance-only, print-advance, or advance-print command.

Code ACDEF = 01001 is reserved for use with code 9 (sense byte 2 bit 4) and causes a unit check status when detected in the vertical format buffer.

Code ACDEF = 01100 is reserved for use with unit exception status (forms overflow) when detected in the vertical format buffer.

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test-I/O	X	X	0	0	0	0	0	0
	X	X	1	1	0	0	0	0
Set-inhibit-status (invalid for C/SP)	X	X	0	1	0	0	0	0
Reset-inhibit-status (invalid for C/SP)	X	X	1	0	0	0	0	0
Sense	X	X	X	X	0	1	0	0
Read	A	B	C	D	E	F	1	0
Control (used for diagnostics)	X	X	X	X	X	X	1	1

LEGEND:

Bit position 7 is the least significant bit position.

X may be a 1 or 0 bit and is ignored by control unit.

A (read bit)
0 = read data

B
0 = stop on errors
1 = sort errors

D = 0, E = 0 — 80-column record

D = 0, E = 1 — short card 51-column read

D = 1, E = 1 — short card 66-column read

C = 1, F = 0 — dual translate feature

C = 0, F = 0 — read in translate mode

F (detail bit)
0 = read in translate mode
1 = read in image mode type all information in manually opposite numbers

A = 1, F = 1 — maintenance mode read

Cards are advanced but data is not read. Two bytes containing the 16 special diagnostic status bits are sent to the multiplexer channel for maintenance purposes.

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test	X	X	1	1	0	0	0	0
				or				
	X	X	0	0	0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	X	X	X	X	0	1	0	0
Control (used for nondata transfer commands)	A	X	C	D	E	X	1	1
Load buffer (write)	A	X	C	D	E	F	0	1
Unload buffer (read)	A	X	X	X	X	F	1	0

LEGEND:

X may be a 1 or 0 bit and is ignored.

A, C, D, E, F modified/detail bits are as follows.

0604 Modified/Detail Bits (Part 1 of 3)

A, C, D, E	Detail Bits for Control Command Interpretation
A=0	Denotes normal operation
A=1	Indicates transfer postpunch read data to the punch buffer (this function is a maintenance feature only)
C=1	Functions to advance the cards one station (feed a card) and to place the card punched on the previous punch order into the select stacker
D=1	Functions to feed and punch a card
E=1	Functions to feed a card

0604 Modified/Detail Bits (Part 2 of 3)

A, B, C, D, E, F	Detail Bits for Load Buffer Command Interpretation
A=0	Functions to load the punch buffer
A=1	Functions to load the read buffer (read buffer test function)
C=1	Functions to advance the cards one station (feed a card) and to place the card punched on the previous punch order into the select stacker
D=1	Functions to feed and punch a card
E=1	Functions to feed a card
F=0	Functions to cause cards to be punched in compress mode
F=1	Functions to cause cards to be punched in image mode

0604 Modified/Detail Bits (Part 3 of 3)

A,F	Detail Bits for Control Command Interpretation
A=0	Functions to unload the read buffer
A=1	Functions to unload the punch buffer
F=0	Functions to read data punched in the compressed mode
F=1	Functions to read data punched in the image mode

Command	Bit Positions							
	0	1	2	3	4	5	6	7
SEEK								
Seek	0	0	0	0	0	1	1	1
Seek-head	0	0	0	1	1	0	1	1
Seek-cylinder	0	0	0	0	1	0	1	1
WRITE								
Write-home-address	0	0	0	1	1	0	0	1
Write-TD-record	0	0	0	1	0	1	0	1
Write-count-key-and-data	0	0	0	1	1	1	0	1
Write-special-count-key-and-data	0	0	0	0	0	0	0	1
Write-data	0	0	0	0	0	1	0	1
Write-key-and-data	0	0	0	0	1	1	0	1

READ								
Read-home-address	M	0	0	1	1	0	1	0
Read-TD-record	M	0	0	1	0	1	1	0
Read-count	M	0	0	1	0	0	1	0
Read-data	M	0	0	0	0	1	1	0
Read-key-and-data	M	0	0	0	1	1	1	0
Read-count-key-and-data	M	0	0	1	1	1	1	0
Initial-program-load	M	0	0	0	0	0	1	0
SEARCH								
Search-home-address-equal	M	0	1	1	1	0	0	1
Search-ID-equal	M	0	1	1	0	0	0	1
Search-ID-high	M	1	0	1	0	0	0	1
Search-ID-equal-or-high	M	1	1	1	0	0	0	1
Search-key-equal	M	0	1	0	1	0	0	1
Search-key-high	M	1	0	0	1	0	0	1
Search-key-equal-or-high	M	1	1	0	1	0	0	1
Search-key-and-data-equal	M	0	1	0	1	1	0	1
Search-key-and-data-high	M	1	0	0	1	1	0	1
Search-key-and-data-equal-or-high	M	1	1	0	1	1	0	1
Continue-scan (See 3.2.5.9.)								

Command	Bit Positions							
	0	1	2	3	4	5	6	7
SENSE								
Sense-I/O	0	0	0	0	0	1	0	0
Sense-reserve	1	1	1	1	0	1	0	0
Sense-release	1	1	0	1	0	1	0	0
MISCELLANEOUS								
Set-file-mask	0	0	0	1	1	1	1	1
Recalibrate	0	0	0	1	0	0	1	1
No-operation	0	0	0	0	0	0	1	1
Space-count	0	0	0	0	1	1	1	1
Erase	0	0	0	1	0	0	0	1
Test-I/O	0	0	0	0	0	0	0	0

LEGEND:

Bit positions in a byte, position 7 being the least significant bit position.

The M bit, when 0, establishes normal operation mode. The M bit, when 1, establishes multiple-track mode. This bit is ignored by the control unit on an initial-program-load command. When the M bit is set to 1 in the command, the disc unit upon encountering the index mark, increments the head register to switch to the next head. This M bit when set to 1 in a search-truncated command, and the track descriptor record (TDR) is used as a data record, enables the program to cascade down the cylinder switching to the next head after reaching the index mark. If the track descriptor record is not used as a data record, and the data length is 0 along with external interrupt status containing unit exception, disc transfer terminates. If the TDR does not have a data length of 0, the data within the TDR will then be presented, and the read/write, search-truncated, jump, and chain continue.

Command	Command Code															
	Multiple Track OFF								Multiple Track ON (if applicable)							
	Bit Positions								Bit Positions							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
CONTROL																
Seek	0	0	0	0	0	1	1	1								
Seek cylinder	0	0	0	0	1	0	1	1								
Seek head	0	0	0	1	1	0	1	1								
Set sector	0	0	1	0	0	0	1	1								
Seek and set sector	0	0	1	0	0	1	1	1								
Recalibrate	0	0	0	1	0	0	1	1								
Set file mask	0	0	0	1	1	1	1	1								
Space count	0	0	0	0	1	1	1	1								
Retry restart (1)	0	0	1	1	1	0	1	1								
No operation	0	0	0	0	0	0	1	1								
Restore	0	0	0	1	0	1	1	1								

Command	Command Code															
	Multiple Track OFF								Multiple Track ON (if applicable)							
	Bit Positions								Bit Positions							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
WRITE																
Home address	0	0	0	1	1	0	0	1								
Record 0	0	0	0	1	0	1	0	1								
Erase	0	0	0	1	0	0	0	1								
Count, key and data	0	0	0	1	1	1	0	1								
Special count, key and data	0	0	0	0	0	0	0	1								
Data	0	0	0	0	0	1	0	1								
Key and data	0	0	0	0	1	1	0	1								

SEARCH																
Home address equal	0	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1
Identifier equal	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	1
Identifier high	0	1	0	1	0	0	0	1	1	1	0	1	0	0	0	1
Identifier equal or high	0	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1
Key equal	0	0	1	0	1	0	0	1	1	0	1	0	1	0	0	1
Key high	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	1
Key equal or high	0	1	1	0	1	0	0	1	1	1	1	0	1	0	0	1
READ																
Home address	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0
Count	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0
Record 0	0	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
Data	0	0	0	0	0	1	1	0	1	0	0	0	0	1	1	0
Key and data	0	0	0	0	1	1	1	0	1	0	0	0	1	1	1	0
Count, key and data,	0	0	0	1	1	1	1	0	1	0	0	1	1	1	1	0
IPL	0	0	0	0	0	0	1	0								
Sector	0	0	1	0	0	0	1	0								

Command	Command Code															
	Multiple Track OFF								Multiple Track ON (if applicable)							
	Bit Positions								Bit Positions							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
SENSE																
Sense I/O	0	0	0	0	0	1	0	0								
Command code sense	1	0	0	0	0	1	0	0								
sense (2)																
Read reset buffered log	1	0	1	0	0	1	0	0								
Release	1	0	0	1	0	1	0	0								
Reserve	1	0	1	1	0	1	0	0								
Test I/O	0	0	0	0	0	0	0	0								

NOTES:

(1) Implemented on SPERRY UNIVAC 1100 Series systems only.
(2) Implemented on SPERRY UNIVAC Series 90 systems only.

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test	X	X	0	0	0	0	0	0
				or				
	X	X	1	1	0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	0	0	0	0	0	1	0	0
Write	0	0	0	0	0	0	0	1
Read	0	0	0	X	0	0	1	0
Read backward	0	0	0	X	1	1	0	0
Control	0	0	C	C	C	1	1	1

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Mode set	D	D	M	M	M	0	1	1

LEGEND:

Bit position 7 is the least significant bit position.

X may be a 1 or 0 bit and is ignored.

CCC (control code):

000 = rewind
001 = rewind-with-interlock
010 = erase
011 = write tape mark
100 = backspace block
101 = backspace file
110 = forward space block
111 = forward space file

LEGEND: (cont)

MMM (mode modifier):

000	=	no operation
001	=	reserved for failure-finding mode (maintenance personnel only)
010	=	odd parity recording, data converter ON, density per DD
011	=	low gain (applies only to read or space operation immediately following mode set command; gain is reset to normal gain at end of operation). DD must be 01.
100	=	even parity recording, data converter OFF, density per DD
101	=	invalid
110	=	odd parity recording, data converter OFF, density per DD
111	=	invalid

DD (density set), applicable to 7-track operation only:

00	=	200 bpi
01	=	556 bpi
10	=	800 bpi
11	=	not used (invalid command)

Nine-track operation forces 800 bpi and odd vertical parity recording.
Nine-track operation overrides but does not reset 7-track mode setting.

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test	X	X	0	0	0	0	0	0
				or				
	X	X	1	1	0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	0	0	0	0	0	1	0	0
Sense/reserve	1	1	1	1	0	1	0	0
Sense/release	1	1	0	1	0	1	0	0
Write	0	0	0	0	0	0	0	1
Read	0	0	0	1	0	0	1	0

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Read/backward	0	0	0	1	1	1	0	0
Control	0	0	C	C	C	1	1	1
Mode set	D	D	M	M	M	0	1	1

LEGEND:

X may be a 1 or 0 bit and is ignored.

I = 1 – Set unit check status if bit 4 of sense data byte 3 is set.
I = 0 – Do not set unit check status if bit 4 of sense data byte 3 is set.

CCC (control command code):

000 = rewind
001 = rewind with interlock
010 = erase
011 = write tape mark
100 = backspace block

LEGEND: (cont)

101	=	backspace file
110	=	forward space block
111	=	forward space file

DDMMM (density set, mode modifier):

00011	=	request TIE (9-track NRZI)
11000	=	set 1600-bpi mode (This mode is set for 9-track operation when control unit is reset or the master unit is cleared.)
11001	=	set 800-bpi mode for 9-track
00000	=	no operation
00001	=	reset simulate mode
01001	=	set simulate mode
10001	=	set monitor mode
01011	=	set low gain (The gain condition applies to a read or space operation immediately following the mode-set command. At the end of the operation, the mode is reset to high gain.)
00MMM	=	set 200-bpi mode for 7-track
01MMM	=	set 556-bpi mode for 7-track Applies only for certain values of MMM.
10MMM	=	set 800-bpi mode for 7-track

Nine-track operation overrides, but does not reset, a 7-track mode setting. Seven-track operation overrides, but does not reset, a 9-track mode setting. Nine-track operation mode settings apply only to write, write-tape-mark, or erase commands executed from load point.

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test	X	X	0	0	0	0	0	0
				or				
	X	X	1	1	0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	0	0	0	0	0	1	0	0
Sense/reserve	1	1	1	1	0	1	0	0
Sense/release	1	1	0	1	0	1	0	0
Write	0	0	0	0	0	0	0	1

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test	X	X	0	0	0	0	0	0
				or				
	X	X	1	1	0	0	0	0
Read	0	0	0	X	0	0	1	0
Read backward	0	0	0	X	1	1	0	0
Control	0	0	C	C	C	1	1	1
Mode set	D	D	M	M	M	0	1	1

LEGEND:

Bit position 7 is the least significant bit position.

X may be either a 1 or 0 bit and is ignored.

CCC (control code):

- 000 = rewind
- 001 = rewind-with-interlock
- 010 = erase
- 011 = write tape mark
- 100 = backspace block
- 101 = backspace file
- 110 = forward space block
- 111 = forward space file

MMM (mode modifier):

- 000 = no operation, 1600 bpi if DD = 11
- 001 = failure-finding mode (maintenance personnel only), 800 bpi if DD = 11
- 010 = odd parity recording, data converter ON, translator OFF, density per DD
- 011 = low gain (applies only to read or space operation immediately following mode set command; gain is reset to normal gain at end of operation). DD must be 01. Track-in-error DD=00

100 = even parity recording, data converter OFF, density per DD, translator OFF
101 = 7-track, even parity, translator ON, data converter OFF, density per DD
110 = odd parity recording, data converter OFF, translator OFF, density per DD
111 = 7-track, odd parity, translator ON, data converter OFF, density per DD

DD (density set), applicable to 7-track operation only:

00 = 200 bpi
01 = 556 bpi
10 = 800 bpi
11 = set 9-track mode

Nine-track operation forces 800 bpi and odd vertical parity recording.
Nine-track operation overrides but does not reset 7-track mode setting.

Command	Bit Positions							
	15	14	13	12	11	10	9	8
Test	X	X	0	0 or 1	0	0	0	0
	X	X	1		0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	0	0	0	0	0	1	0	0
Sense/reserve	1	1	1	1	0	1	0	0
Sense/release	1	1	0	1	0	1	0	0
Write	0	0	0	0	0	0	0	1

Command	Bit Positions							
	15	14	13	12	11	10	9	8
Read	0	0	0	1	0	0	1	0
Read backward	0	0	0	1	1	1	0	0
Control	0	0	C	C	C	1	1	1
Mode set	D	D	M	M	M	0	1	1

LEGEND:

X may be a 1 or 0 bit and is ignored.

I = 1 — Set unit check status if bit 4 of sense data byte 3 is set.

I = 0 — Do not set unit check status if bit 4 of sense data byte 3 is set.

CCC (control code):

000	=	rewind
001	=	rewind with interlock
010	=	erase
011	=	write tape mark
100	=	backspace block
101	=	backspace file
110	=	forward space block
111	=	forward space file

DDMMM (density set, mode modifier):

00011	=	request TIE (9-track NRZI)
11000	=	set 1600-bpi mode (This mode is set for 9-track operation when control unit is reset or the master unit is cleared.)
11001	=	set 800-bpi mode for 9-track
00000	=	no operation

LEGEND: (cont)

00001 = reset simulate mode
01001 = set simulate mode
10001 = set monitor mode
01011 = set low gain (The low gain condition applies to a read or space operation immediately following the mode-set command. At the end of the operation, the mode is reset to high gain.)
00MMM = set 200-bpi mode for 7-track
01MMM = set 556-bpi mode for 7-track
10MMM = set 800-bpi mode for 7-track
} applies only for certain values of MMM.

Nine-track operation overrides, but does not reset, a 7-track mode setting. Seven-track operation overrides, but does not reset, a 9-track mode setting. Nine-track operation mode settings apply only to write, write-tape-mark, or erase commands executed from load point.

Command	Bit Positions							
	0	1	2	3	4	5	6	7
Test	X	X	0	0 or 1	0	0	0	0
	X	X	1	1	0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	0	0	0	0	0	1	0	0
Punch	0	0	0	0	0	A	0	1
Read	0	0	0	0	0	A	1	0
Control (used for nondata transfer commands)	1	0	0	0	0	0	1	1

LEGEND:

Bit position 7 is the least significant bit position.

X may be a 1 or 0 bit and is ignored.

Test-input/output-status (TIO) instruction is used with 9200/9200 II/9300/9300 II Systems only.
A = 0 indicates character recognition is operative.
A = 1 indicates operation in binary mode.

Command Byte	Bit Positions							
	0	1	2	3	4	5	6	7
Operational								
Test I/O	X	X	X	X	0	0	0	0
Set inhibit status	X	X	0	1	0	0	0	0
Reset inhibit status	X	X	1	0	0	0	0	0
Sense	X	X	X	X	0	1	0	0
Read 1 backward	X	X	0	1	1	1	0	0
Read 2 backward	X	X	1	X	1	1	0	0
Read 0 backward	X	X	0	0	1	1	0	0
Stacker 2 immediate	X	X	1	0	0	0	1	1
Stacker 3 immediate	X	X	1	1	0	0	1	1
No-op	X	X	0	X	0	0	1	1
Read Select								
OCR read	X	0	0	0	0	1	1	1
No OCR read	X	0	0	1	0	1	1	1
OCR and mark read	0	0	1	0	0	1	1	1
Mark read	0	0	1	1	0	1	1	1

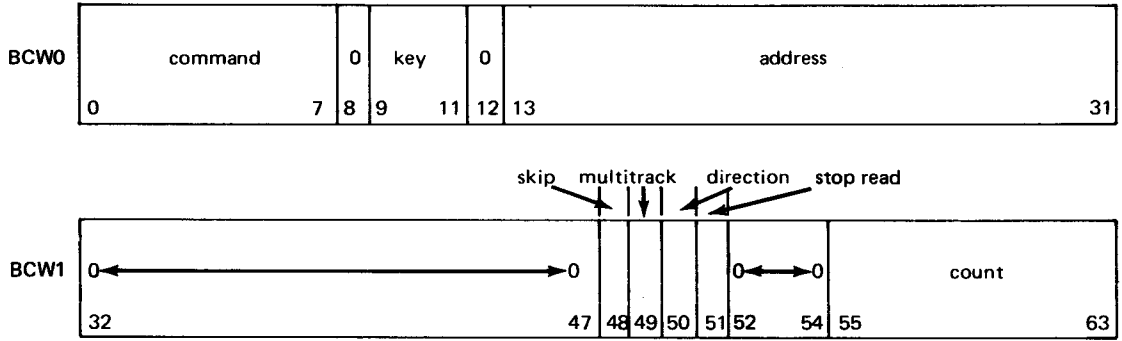
OCR and card read	0	1	X	0	0	1	1	1
Card read	0	1	X	1	0	1	1	1
OCR read and card read translate	1	1	X	0	0	1	1	1
Card read translate	1	1	X	1	0	1	1	1
Mark read translate	1	0	1	1	0	1	1	1
OCR read and mark read translate	1	0	1	0	0	1	1	1
Mode								
Stacker 2 mode preselect	0	1	0	0	1	0	1	1
Stacker 3 mode preselect	0	1	0	1	1	0	1	1
Stacker 2 preselect and modulus 10 check digit mode	1	1	0	0	1	0	1	1
Stacker 3 preselect and modulus 10 check digit mode	1	1	0	1	1	0	1	1
Modulus 10 check digit select mode	1	0	0	X	1	0	1	1
Stacker preselect mode reset	0	0	X	0	1	0	1	1
Document length								
Document length 3.00 to 3.30	1	1	1	X	1	0	1	1
Document length 3.31 to 4.00	1	0	1	X	1	0	1	1
Document length 4.01 to 5.90	0	1	1	X	1	0	1	1
Document length 5.91 to 8.75	0	0	1	X	1	0	1	1

Command Byte	Bit Positions							
	0	1	2	3	4	5	6	7
Mark read stacker								
Mark read stacker, row 0-1	0	0	0	1	1	1	1	1
Mark read stacker, row 2-3	0	0	1	1	1	1	1	1
Mark read stacker, row 4-5	0	1	0	1	1	1	1	1
Mark read stacker, row 6-7	0	1	1	1	1	1	1	1
Mark read stacker reset	0	X	X	0	1	1	1	1
Diagnostic								
Set diagnostic	1	X	X	X	1	1	1	1
Read diagnostic	X	X	X	X	X	X	1	0
Write diagnostic	X	X	X	X	X	X	0	1
Reset diagnostic	0	X	X	X	1	1	1	1

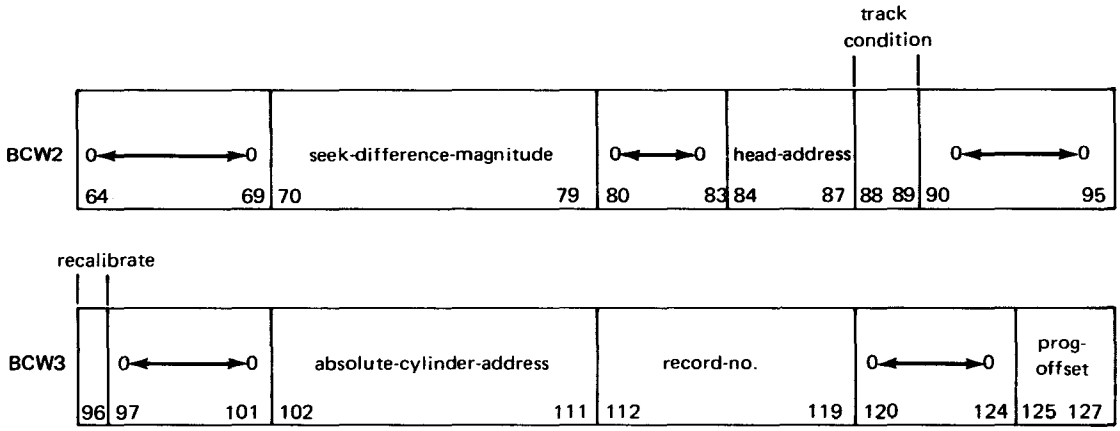
NOTES:

1. Bit positions 24 through 31 are used in XIOF and CAW, bit positions 0 through 7 are used in CCW (3.2).
2. Bit position 31 (or 7) is least significant bit position. X may be either a 1 or 0 bit.

Buffer Control Word (BCW) IDA Format



Buffer Control Word (BCW) IDA Format (cont)



Buffer Control Word (BCW) IDA Format (cont)

Bits	Allocation	Function
0–7	Command	Command code to be executed by IDA; bits 0–3 must be zero
8		Unassigned; must be set to zero
9–11	Key	3-bit field containing storage protection key
12		Unassigned; must be set to zero
13–31	Address	Storage address on which command operates
32–47		Unassigned; must be set to zero
48	Skip sentinel	Set with read data command to indicate data transfers inhibited to main storage; set with search/read commands to indicate search begins at index

Buffer Control Word (BCW) IDA Format (cont)

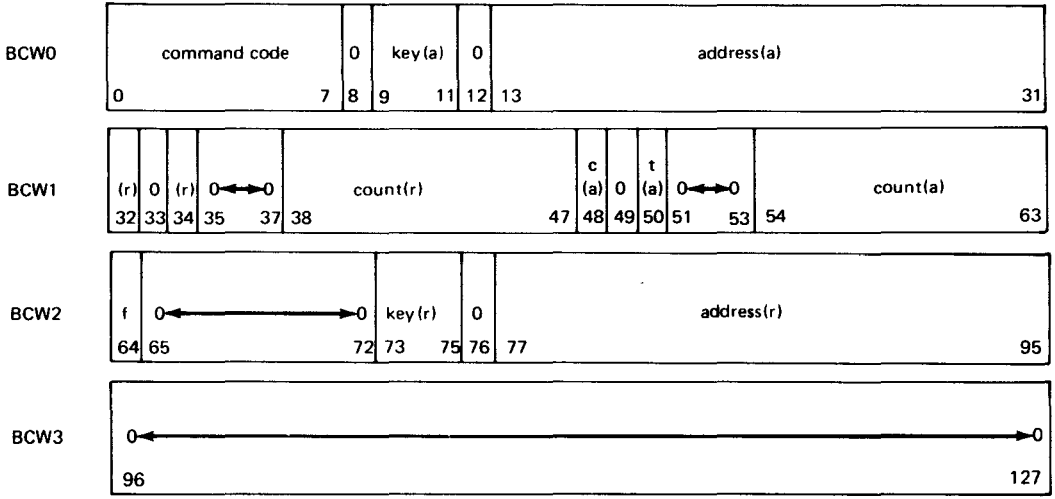
Bits	Allocation	Function
49	Multitrack sentinel	Set to 1 with search/read command to indicate search limited to cylinder boundaries rather than single track
50	Direction sentinel	If 1, specifies accessor moves in direction of decreasing cylinder numbers
51	Stop read	Stop read command on record which causes error
52–54		Unassigned; must be set to zero
55–63	Count	On search/read commands = number of bytes to be searched On data read or write commands = number of records to be processed

64–69		Unassigned; must be zero
70–79	Seek difference magnitude	During seek operation, specifies magnitude of difference between accessor present position and desired position
80–83		Unassigned; must be set to zero
84–87	Head address	4-bit field specifying current operation head address
88,89	Track condition	Condition of track where operation acts
90–95		Unassigned; must be set to zero
96	Recalibrate	Set to 1 = accessor reoriented and moved to cylinder 0; overrides bits 71–79 and 50
97–101		Unassigned; must be set to zero
102–111	Absolute cylinder address	Final position of accessor after completed seek or recalibrate

Buffer Control Word (BCW) IDA Format (cont)

Bits	Allocation	Function
112–119	Record number	Number of record where operation is performed or initiated
120–124		Unassigned; must be set to zero
125–127	Programmed offset	Bit 125 = 1 programmed offset used for command Bit 125 = 0 programmed offset not used; bits 126 and 127 ignored Bit 126 = 1 major offset Bit 126 = 0 minor offset Bit 127 = 1 offset away from hub Bit 127 = 0 offset toward hub

Buffer Control Word (BCW) IPC Format



Buffer Control Word (BCW) IPC Format (cont)

Bits	Allocation	Function
0–7	Command code	Field accessed by IPC during SIO instruction
8		Unassigned; must be set to zero
9–11 and 73–75	Key (a, r)	3-bit field containing I/O storage protection key
12		Unassigned; must be set to zero
13–31 and 77–95	Address (a, r)	Allows IPC to reference any byte in main storage during data transfer sequences Bits 31 and 95 = 0 most significant byte of addressed half word Bits 31 and 95 = 1 least significant byte of addressed half word

32 and 48	c (r, a)	Specifies data chaining operations when set to 1
33		Unassigned; must be set to zero
34 and 50	t(r) and t(a)	Single control bit used with c(a) bit: c(a) = 0 and t = 0 — means use A fields for current data transfer sequence (no data chaining) c(a) = 0 and t = 1 — terminates control c(a) = 1 and t = 0 — use A fields for current data transfer sequence (data chaining initial A and R setting) c(a) = 1 and t = 1 — A fields depleted; replacement operation required

LEGEND

a = active
c = chaining
f = flag
r = replacement
t = transfer

Buffer Control Word (BCW) IPC Format (cont)

Bits	Allocation	Function
34 and 50 (cont)	t(r) and t(a) (cont)	If t(a) and c(a) = 1: f = 0 — terminates with buffer wraparound error f = 1, c(r) = 1 or 0, t(r) = 1 — terminates normally f = 1, c(r) = 0, t(r) = 0 — normal data transfer; no chaining f = 1, c(r) = 1, t(r) = 0 — normal data transfer with chaining
35–37		Unassigned; must be set to zero

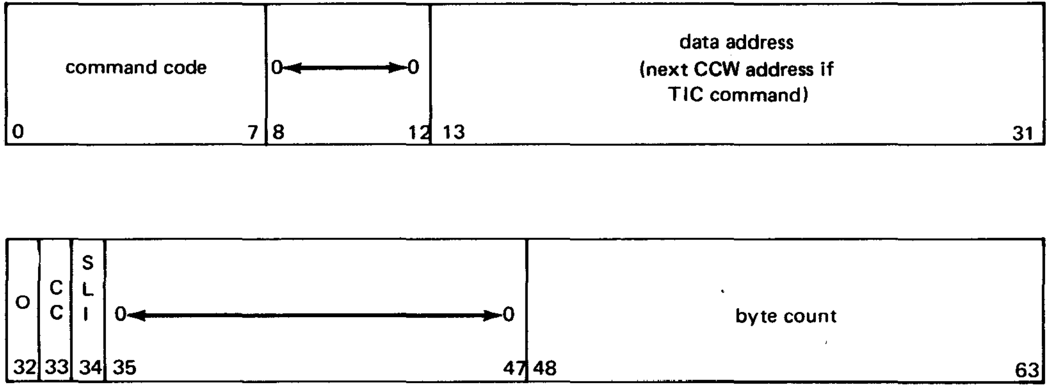
38–47 and 54–63	Count (r) and count (a)	Byte count required for all data transfer operations
49		Unassigned; must be set to zero
51–53		Unassigned; must be set to zero
64	f (flag bit)	Indicates to IPC that current contents of r fields are valid for replacement operation
65–72, 76, and 96–127		Unassigned; must be set to zero

LEGEND

a = active
c = chaining
f = flag
r = replacement
t = transfer

Bits	Allocation	Function
0–2	Key	I/O storage protection key used by channel for all storage accesses of data and CCWs
3–12		Bits set to zero
13–31	First CCW address instruction	Controls I/O operation initiated by SIO

CHANNEL COMMAND WORD (CCW) SELECTOR CHANNEL

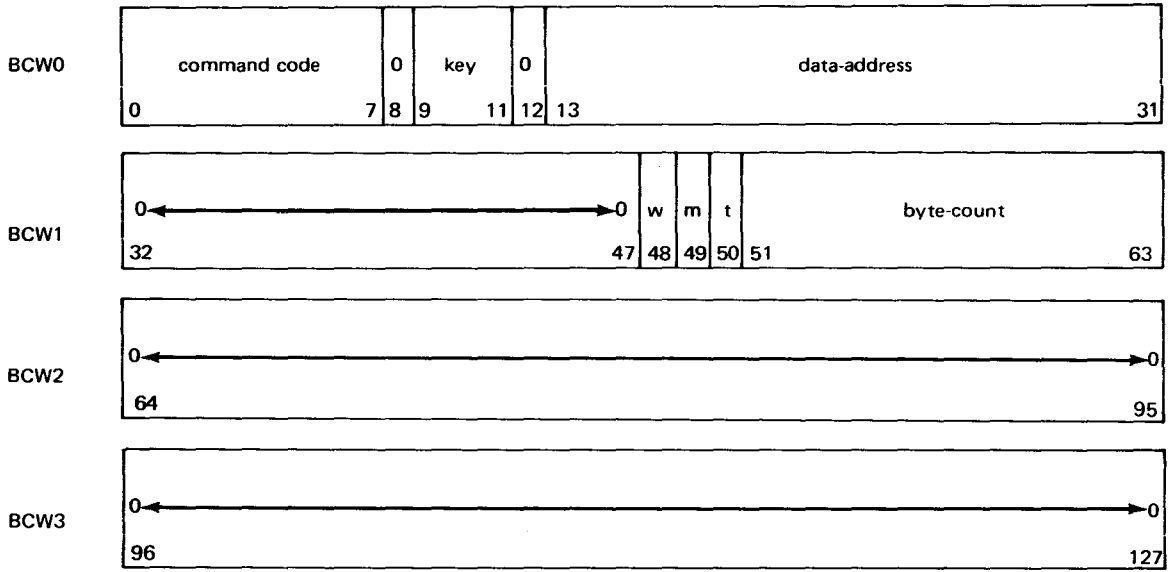


Bits	Allocation	Function
0–7	Command code*	Specifies operation to be performed by device and channel
8–12		Bits set to zero
13–31	Data address	Address of location in main storage into or from which first byte of data is transferred
32		Bit set to zero
33	CC (chain command flag)	When valid ending device status received, new CCW fetched and operation specified by new command code initiated
34	SLI (suppress length indication flag)	If set to 1, incorrect length condition not indicated to program; if CC = 1 also, command chaining not suppressed
35–47		Unassigned; must be set to zero

48–63	Byte count	Byte count required for all data transfer operations

***Command codes:**

Code Bits								Command
0	1	2	3	4	5	6	7	
M	M	M	M	1	0	0	0	Transfer in channel (TIC)
M	M	M	M	0	1	0	0	Sense
M	M	M	M	M	M	0	1	Write
M	M	M	M	M	M	1	0	Read
M	M	M	M	1	1	0	0	Read backward
M	M	M	M	M	M	1	1	Control
M	M	M	M	0	0	0	0	Test


BCW0
command code
0
key
0
data-address
0
7
8
9
11
12
13
31
BCW1
0
0
w
m
t
byte-count
32
47
48
49
50
51
63
BCW2
0
0
64
95
BCW3
0
0
96
127

Bits	Allocation	Function
0–7	command code*	Specifies operation to be performed by device and channel
8		Unassigned; must be set to zero
9–11	key	Contains I/O storage protection key
12		Unassigned; must be set to zero
13–31	data address	Allows multiplexer channel to reference any byte in main storage during data transfer sequences
32–47		Unassigned; must be set to zero
48	w	w = 0 input operation (read) w = 1 output operation (write)

Bits	Allocation	Function
49	m	m = 0 ascending address (forward sequence) m = 1 descending address (reverse sequence)
50	t	t = 0 transfer data t = 1 termination of data transfer
51–63	byte count	Contains byte count required for all data transfers
64–127		Unassigned; must be set to zero

***Command codes:**

Code Bits								Command
0	1	2	3	4	5	6	7	
M	M	M	M	0	1	0	0	Sense
M	M	M	M	M	M	0	1	Write
M	M	M	M	M	M	1	0	Read
M	M	M	M	1	1	0	0	Read backward
M	M	M	M	M	M	1	1	Control
M	M	M	M	0	0	0	0	Test
M	M	M	M	1	0	0	0	Transfer in channel (TIC)

Device	Address Field Bits*							
	24	25	26	27	28	29	30	31
Console (UNISCOPE 100)	0	0	0	0	0	0	0	0
0717 reader	0	0	0	0	0	0	0	1
0773 printer	0	0	0	0	0	0	1	0
0605 punch	0	0	0	0	0	0	1	1
LA – 0 (CA – 1)	0	0	0	0	0	1	0	0
LA – 6 (CA – 1)	0	0	0	0	0	1	0	1
LA – 1 (CA – 1)	0	0	0	0	0	1	1	0
LA – 7 (CA – 1)	0	0	0	0	0	1	1	1
LA – 2 (CA – 1)	0	0	0	0	1	0	0	0
LA – 8 (CA – 1)	0	0	0	0	1	0	0	1

Device	Address Field Bits*							
	24	25	26	27	28	29	30	31
LA – 3 (CA – 1)	0	0	0	0	1	0	1	0
LA – 9 (CA – 1)	0	0	0	0	1	0	1	1
LA – 4 (CA – 1)	0	0	0	0	1	1	0	0
LA – 10 (CA – 1)	0	0	0	0	1	1	0	1
LA – 5 (CA – 1)	0	0	0	0	1	1	1	0
LA – 11 (CA – 1)	0	0	0	0	1	1	1	1
LA – 0 (CA – 2)	0	0	0	1	0	1	0	0
LA – 6 (CA – 2)	0	0	0	1	0	1	0	1
LA – 1 (CA – 2)	0	0	0	1	0	1	1	0
LA – 7 (CA – 2)	0	0	0	1	0	1	1	1

Device	Address Field Bits*							
	24	25	26	27	28	29	30	31
LA – 2 (CA – 2)	0	0	0	1	1	0	0	0
LA – 8 (CA – 2)	0	0	0	1	1	0	0	1
LA – 3 (CA – 2)	0	0	0	1	1	0	1	0
LA – 9 (CA – 2)	0	0	0	1	1	0	1	1
LA – 4 (CA – 2)	0	0	0	1	1	1	0	0
LA – 10 (CA – 2)	0	0	0	1	1	1	0	1
LA – 5 (CA – 2)	0	0	0	1	1	1	1	0
LA – 11 (CA – 2)	0	0	0	1	1	1	1	1

*Address field bits correspond to bits 24–31 of SIO instruction.

LEGEND:

LA = line adapter

CA = communication adapter

I/O Channel Number Assignment

Channel 0 =	Integrated Peripheral Channel (IPC)
Channel 1 =	Multiplexer channel
Channel 2 =	Unnassigned
Channel 3 =	Integrated Disc Adapter (IDA)

Channel 4 =	Selector channel 1
Channel 5 =	Communications Intelligence Channel (CIC)
Channel 6 =	Selector channel 2
Channel 7 =	I/O Status tabler

IDA Device Addresses

Device	Address Field Bits*							
	24	25	26	27	28	29	30	31
Disc drive 0	0	0	0	0	0	0	0	0
Disc drive 1	0	0	0	0	0	0	0	1
Disc drive 2	0	0	0	0	0	0	1	0
Disc drive 3	0	0	0	0	0	0	1	1
Disc drive 4**	0	0	0	0	0	1	0	0
Disc drive 5**	0	0	0	0	0	1	0	1
Disc drive 6**	0	0	0	0	0	1	1	0
Disc drive 7**	0	0	0	0	0	1	1	1

*Address field bits correspond to bits 24–31 of SIO instruction.

**Requires expansion feature for eight disc drives.

Low-Order Main Storage

Byte Address (Hexadecimal)	0 1 2 3	4 5 6 7	8 9 A B	C D E F
00X				
01X	IOSTCW0	IOSTCW1	IOSTCW2	(Reserved)
02X	IOST	Old PSW	IOST	New PSW
03X	Machine Check	Old PSW	Machine Check	New PSW
04X	Program Exception	Old PSW	Program Exception	New PSW
05X	Supervisor Call	Old PSW	Supervisor Call	New PSW
06X	Interval Timer	Old PSW	Interval Timer	New PSW
07X	(Reserved)		(Reserved)	
08X	Monitor	Old PSW	Monitor	New PSW

09X	(Reserved)					(Reserved)		
0AX	CAW	MM ①	LL ②	RR ③	00 ④	(Reserved)		
0BX	Rel. Reg. 0		Rel. Reg. 1			Rel. Reg. 2	Rel. Reg. 3	
0CX	Rel. Reg. 4		Rel. Reg. 5			Rel. Reg. 6	Rel. Reg. 7	
0DX	(Reserved)					(Reserved)		
0EX	IOST BCSW					IOST BCSW		
0FX	Int. Disc BCW0		Int. Disc BCW1			Int. Disc BCW2	Int. Disc BCW3	
10X	Console BCW0		Console BCW1			Console BCW2	Console BCW3	
11X	Reader BCW0		Reader BCW1			Reader BCW2	Reader BCW3	
12X	Printer BCW0		Printer BCW1			Printer BCW2	Printer BCW3	
13X	Punch BCW0		Punch BCW1			Punch BCW2	Punch BCW3	

Byte Address (Hexadecimal)	0 1 2 3	4 5 6 7	8 9 A B	C D E F
14X	LA0 BCW0*	LA0 BCW1*	LA0 BCW2*	LA0 BCW3*
15X	LA6 BCW0*	LA6 BCW1*	LA6 BCW2*	LA6 BCW3*
16X	LA1 BCW0*	LA1 BCW1*	LA1 BCW2*	LA1 BCW3*
17X	LA7 BCW0*	LA7 BCW1*	LA7 BCW2*	LA7 BCW3*
18X	LA2 BCW0*	LA2 BCW1*	LA2 BCW2*	LA2 BCW3*
19X	LA8 BXW0*	LA8 BCW1*	LA8 BCW2*	LA8 BCW3*
1AX	LA3 BCW0*	LA3 BCW1*	LA3 BCW2*	LA3 BCW3*
1BX	LA9 BCW4*	LA9 BCW1*	LA9 BCW2*	LA9 BCW3*
1CX	LA4 BCW0*	LA4 BCW1*	LA4 BCW2*	LA4 BCW3*

1DX	LA10 BCW4*	LA10 BCW1*	LA10 BCW2*	LA10 BCW3*
1EX	LA5 BCW0*	LA5 BCW1*	LA5 BCW2*	LA5 BCW3*
1FX	LA11 BCW4*	LA11 BCW1*	LA11 BCW2*	LA11 BCW3*
20X	Mux. Subch. 0 BCW0	Mux. Subch. 0 BCW1	Mux. Subch. 0 BCW2	Mux. Subch. 0 BCW3
21X	Mux. Subch. 1 BCW0	Mux. Subch. 1 BCW1	Mux. Subch. 1 BCW2	Mux. Subch. 1 BCW3
22X	Mux. Subch. 2 BCW0	Mux. Subch. 2 BCW1	Mux. Subch. 2 BCW2	Mux. Subch. 2 BCW3
23X	Mux. Subch. 3 BCW0	Mux. Subch. 3 BCW1	Mux. Subch. 3 BCW2	Mux. Subch. 3 BCW3
24X	Mux. Subch. 4 BCW0	Mux. Subch. 4 BCW1	Mux. Subch. 4 BCW2	Mux. Subch. 4 BCW3
25X	Mux. Subch. 5 BCW0	Mux. Subch. 5 BCW1	Mux. Subch. 5 BCW2	Mux. Subch. 5 BCW3
26X	Mux. Subch. 6 BCW0	Mux. Subch. 6 BCW1	Mux. Subch. 6 BCW2	Mux. Subch. 6 BCW3
27X	Mux. Subch. 7 BCW0	Mux. Subch. 7 BCW1	Mux. Subch. 7 BCW2	Mux. Subch. 7 BCW3

Byte Address (Hexadecimal)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
28X	(Unused)															
29X																
2AX																
2BX																
2CX																
2DX																
2EX																
2FX																

30X	(Reserved)			
31X				
32X				
33X				
34X	LA0 BCW0**	LA0 BCW1**	LA0 BCW2**	LA0 BCW3**
35X	LA6 BCW0**	LA6 BCW1**	LA6 BCW2**	LA6 BCW3**
36X	LA1 BCW0**	LA1 BCW1**	LA1 BCW2**	LA1 BCW3**
37X	LA7 BCW0**	LA7 BCW1**	LA7 BCW2**	LA7 BCW3**
38X	LA2 BCW0**	LA2 BCW1**	LA2 BCW2**	LA2 BCW3**
39X	LA8 BCW0**	LA8 BCW1**	LA8 BCW2**	LA8 BCW3**
3AX	LA3 BCW0**	LA3 BCW1**	LA3 BCW2**	LA3 BCW3**

Byte Address (Hexadecimal)	0 1 2 3	4 5 6 7	8 9 A B	C D E F
3BX	LA9 BCW0**	LA9 BCW1**	LA9 BCW2**	LA9 BCW3**
3CX	LA4 BCW0**	LA4 BCW1**	LA4 BCW2**	LA4 BCW3**
3DX	LA10 BCW0**	LA10 BCW1**	LA10 BCW2**	LA10 BCW3**
3EX	LA5 BCW0**	LA5 BCW1**	LA5 BCW2**	LA5 BCW3**
3FX	LA11 BCW0**	LA11 BCW1**	LA11 BCW2**	LA11 BCW3**

*Communication adapter-1

**Communication adapter-2

NOTES:

① MM = machine microcode ID field

② LL = load microcode ID field

③ RR = revision level microcode ID field

④ 00 = zero filler

Character	Printed Symbol	Card Punches	ASCII		EBCDIC	
			Hexadecimal	Decimal	Hexadecimal	Decimal
Letters						
A	A	12–1	41	65	C1	193
B	B	12–2	42	66	C2	194
C	C	12–3	43	67	C3	195
D	D	12–4	44	68	C4	196
E	E	12–5	45	69	C5	197
F	F	12–6	46	70	C6	198
G	G	12–7	47	71	C7	199
H	H	12–8	48	72	C8	200

Character	Printed Symbol	Card Punches	ASCII		EBCDIC	
			Hexadecimal	Decimal	Hexadecimal	Decimal
I	I	12–9	49	73	C9	201
J	J	11–1	4A	74	D1	209
K	K	11–2	4B	75	D2	210
L	L	11–3	4C	76	D3	211
M	M	11–4	4D	77	D4	212
N	N	11–5	4E	78	D5	213
O	O	11–6	4F	79	D6	214
P	P	11–7	50	80	D7	215
Q	Q	11–8	51	81	D8	216
R	R	11–9	52	82	D9	217

S	S	0–2	53	83	E2	226
T	T	0–3	54	84	E3	227
U	U	0–4	55	85	E4	228
V	V	0–5	56	86	E5	229
W	W	0–6	57	87	E6	230
X	X	0–7	58	88	E7	231
Y	Y	0–8	59	89	E8	232
Z	Z	0–9	5A	90	E9	233
a	a	12–0–1	61	97	81	129
b	b	12–0–2	62	98	82	130
c	c	12–0–3	63	99	83	131

Character	Printed Symbol	Card Punches	ASCII		EBCDIC	
			Hexadecimal	Decimal	Hexadecimal	Decimal
d	d	12–0–4	64	100	84	132
e	e	12–0–5	65	101	85	133
f	f	12–0–6	66	102	86	134
g	g	12–0–7	67	103	87	135
h	h	12–0–8	68	104	88	136
i	i	12–0–9	69	105	89	137
j	j	12–11–1	6A	106	91	145
k	k	12–11–2	6B	107	92	146
l	l	12–11–3	6C	108	93	147

m	m	12–11–4	6D	109	94	148
n	n	12–11–5	6E	110	95	149
o	o	12–11–6	6F	111	96	150
p	p	12–11–7	70	112	97	151
q	q	12–11–8	71	113	98	152
r	r	12–11–9	72	114	99	153
s	s	11–0–2	73	115	A2	162
t	t	11–0–3	74	116	A3	163
u	u	11–0–4	75	117	A4	164
v	v	11–0–5	76	118	A5	165
w	w	11–0–6	77	119	A6	166

Character	Printed Symbol	Card Punches	ASCII		EBCDIC	
			Hexadecimal	Decimal	Hexadecimal	Decimal
x	x	11–0–7	78	120	A7	167
y	y	11–0–8	79	121	A8	168
z	z	11–0–9	7A	122	A9	169
Numerals						
0	0	0	30	48	F0	240
1	1	1	31	49	F1	241
2	2	2	32	50	F2	242
3	3	3	33	51	F3	243
4	4	4	34	52	F4	244

5	5	5	35	53	F5	245
6	6	6	36	54	F6	246
7	7	7	37	55	F7	247
8	8	8	38	56	F8	248
9	9	9	39	57	F9	249
Symbols						
Exclamation point	!	12–8–7	21	33	4F	79
Quotation mark, dieresis	"	8–7	22	34	7F	127
Number sign, pound sign	#	8–3	23	35	7B	123
Dollar sign	$	11–8–3	24	36	5B	91
Percent sign	%	0–8–4	25	37	6C	108

Character	Printed Symbol	Card Punches	ASCII		EBCDIC	
			Hexadecimal	Decimal	Hexadecimal	Decimal
Ampersand	&	12	26	38	50	80
Apostrophe, acute accent	'	8–5	27	39	7D	125
Opening parenthesis	(	12–8–5	28	40	4D	77
Closing parenthesis	)	11–8–5	29	41	5D	93
Asterisk	*	11–8–4	2A	42	5C	92
Plus sign	+	12–8–6	2B	43	4E	78
Comma, cedilla	,	0–8–3	2C	44	6B	107
Minus sign, hyphen	–	11	2D	45	60	96
Period, decimal point	.	12–8–3	2E	46	4B	75

Slash, virgule, solidus	/	0–1	2F	47	61	97
Colon	:	8–2	3A	58	7A	122
Semicolon	;	11–8–6	3B	59	5E	94
Less than	<	12–8–4	3C	60	4C	76
Equal sign	=	8–6	3D	61	7E	126
Greater than	>	0–8–6	3E	62	6E	110
Question mark	?	0–8–7	3F	63	6F	111
Commercial at symbol	@	8–4	40	64	7C	124
Opening bracket	[	12–8–2	5B	91	4A	74
Closing bracket	]	11–8–2	5D	93	5A	90
Reverse slash	\	0–8–2	5C	92	E0	224

Character	Printed Symbol	Card Punches	ASCII		EBCDIC	
			Hexadecimal	Decimal	Hexadecimal	Decimal
Circumflex	^	11–8–7	5E	94	5F	95
Underline	—	0–8–5	5F	95	6D	109
Grave accent	`	8–1	60	96	79	121
Opening brace	{	12–0	7B	123	C0	192
Closing brace	}	11–0	7D	125	D0	208
Vertical line	|	12–11	7C	124	6A	106
Overline, tilde	~	11–0–1	7E	126	A1	161

Character	Card Punches	ASCII		EBCDIC	
		Hexadecimal	Decimal	Hexadecimal	Decimal
Nonprintable Characters					
ACK (acknowledge)	0–9–8–6	06	6	2E	46
BEL (bell)	0–9–8–7	07	7	2F	47
BS (backspace)	11–9–6	08	8	16	22
CAN (cancel)	11–9–8	18	24	18	24
CR (carriage return)	12–9–8–5	0D	13	0D	13
DC1 (device control 1)	11–9–1	11	17	11	17
DC2 (device control 2)	11–9–2	12	18	12	18
DC3 (device control 3)	11–9–3	13	19	13	19

Character	Card Punches	ASCII		EBCDIC	
		Hexadecimal	Decimal	Hexadecimal	Decimal
DC4 (device control 4)	9–8–4	14	20	3C	60
DEL (delete)	12–9–7	7F	127	07	7
DLE (data link escape)	12–11–9–8–1	10	16	10	16
DS (digit select)	11–0–9–8–1	80	128	20	32
EM (end of medium)	11–9–8–1	19	25	19	25
ENQ (enquiry)	0–9–8–5	05	5	2D	45
EOT (end of transmission)	9–7	04	4	37	55
ESC (escape)	0–9–7	1B	27	27	39
ETB (end of transmission block)	0–9–6	17	23	26	38

ETX (end of text)	12–9–3	03	3	03	3
FF (form feed)	12–9–8–4	0C	12	0C	12
FS (file separator)	11–9–8–4	1C	28	1C	28
FS (field separator)	0–9–2	82	130	22	34
GS (group separator)	11–9–8–5	1D	29	1D	29
HT (horizontal tabulation)	12–9–5	09	9	05	5
LF (line feed)	0–9–5	0A	10	25	37
NAK (negative acknowledge)	9–8–5	15	21	3D	61
NUL (null)	12–0–9–8–1	00	0	00	0
RS (record separator)	11–9–8–6	1E	30	1E	30
SI (shift in)	12–9–8–7	0F	15	0F	15

Character	Card Punches	ASCII		EBCDIC	
		Hexadecimal	Decimal	Hexadecimal	Decimal
SO (shift out)	12–9–8–6	0E	14	0E	14
SOH (start of heading)	12–9–1	01	1	01	1
SOS (significance start)	0–9–1	81	129	21	33
SP (space)		20	32	40	64
STX (start of text)	12–9–2	02	2	02	2
SUB (substitute)	9–8–7	1A	26	3F	63
SYN (synchronous idle)	9–2	16	22	32	50
US (unit separator)	11–9–8–7	1F	31	1F	31
VT (vertical tabulation)	12–9–8–3	0B	11	0B	11

Hexadecimal to Decimal:

Working from right to left with the hexadecimal digits to be converted, select the decimal number from the digit position column corresponding to each hexadecimal digit. Add the selected decimal numbers to complete the conversion.

Decimal to Hexadecimal:

1. Select the highest decimal number from the table that is less than the decimal number to be converted.
2. Subtract this number from the number to be converted.
3. Note the corresponding hexadecimal digit, its digit position, and the difference.
4. Substitute the difference for the decimal number to be converted and repeat steps 1 and 2 until a zero difference is obtained.
5. Include a 0 for each unused digit position.

The resulting hexadecimal number is the conversion.

Hexadecimal Digit Positions											
6		5		4		3		2		1	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15

HEXADECIMAL ADDITION AND SUBTRACTION TABLE

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

SIGN CONVENTIONS

Hexadecimal Representation		Binary Representation	Sign	
Generation	Digit		Value	Mode
External	A	1010	Positive	ASCII
	B	1011	Negative	
Processor	C	1100	Positive	EBCDIC
	D	1101	Negative	
External	E	1110	Positive	
	F	1111	Positive	

LINKAGE REGISTER CONVENTIONS

Register	Contents
0	Parameter register
1	Parameter/list register
2-12	Free registers
13	Save area register
14	Return address register
15	Entry point register

ASSEMBLER OPERATORS

Class	Operator	Description	Hierarchy
Arithmetic	*/	A*/B is equivalent to $A*2^B$.	6
	//	Covered quotient; A//B is equivalent to (A + B − 1)/B.	5
	/	A/B means arithmetic quotient of A and B.	5
	*	A*B means arithmetic product of A and B.	5
	−	A−B means arithmetic difference of A and B.	4
	+	A+B means arithmetic sum of A and B.	4
Logical	**	A**B means logical product AND of A and B.	3
	++	A++B means logical sum OR of A and B.	2
	−−	A− −B means logical differences XOR of A and B.	2
Relational	=	A=B has value 1 if true; has value 0 if false.	1
	>	A>B has value 1 if true; has value 0 if false.	1
	<	A<B has value 1 if true; has value 0 if false.	1

NOTE:

The hierarchy numbers mean that operations with the higher numbers are performed first. Operations are performed from left to right.

STATEMENT CONVENTIONS

Capital letters, parentheses, and punctuation marks	Must be coded exactly as shown
Lowercase letters and terms	Represent information supplied by the programmer
Braces { }	Necessary entries from which one must be chosen
Brackets [ ]	Optional entries
Ellipsis . . .	Indefinite number of entries
Shading	Default option

Operation	Operand	Comments
Fixed Point		
DC	[d] t[L_n] $\left\{\begin{matrix}'c'\\(c)\end{matrix}\right\}$	d = duplication factor in decimal t = type constant* L_n = explicit length factor in decimal c = the constant specification for data (c) = the constant specification for an address
DS	[d] t[L_n] $\left[\begin{matrix}'c'\\(c)\end{matrix}\right]$	
Floating Point		
DC	[d] t[L_n] [S+n] 'c[E±n]'	S+n = scale modifier c[E±n] = the constant specification with optional exponent

*See assembler DEFINE CONSTANT (DC) AND DEFINE STORAGE (DS) TYPES.

Type Code	Constant or Storage Type	Alignment	Source Code Specification		Storage Format	Truncation or Padding	Length in Bytes		
							Implied	Minimum Explicit	Maximum Explicit *
C	Character	None	Characters	C' '	Character	Right	Variable	1	256 (DC) 65,535 (DS)
X	Hexadecimal	None	Hexadecimal digits	X' '	Hexadecimal	Left	Variable	1	256 (DC) 65,535 (DS)
B	Binary	None	Binary digits	B' '	Binary	Left	Variable	1	256
P	Packed decimal	None	Decimal digits	P' '	Packed decimal	Left	Variable	1	16
Z	Zoned decimal	None	Decimal digits	Z' '	Character	Left	Variable	1	16
H	Half word, fixed point	Half word	Decimal digits	H' '	Fixed-point binary	Left	2	1	8
F	Full word, fixed point	Full word	Decimal digits	F' '	Fixed-point binary	Left	4	1	8

Y	Half-word address	Half word	Expression	Y( )	Binary	Left	2	1	2
A	Full-word address	Full word	Expression	A( )	Binary	Left	4	1	4
S	Base and displacement	Half word	One or two expressions	S( )	Base and displacement	None	2	2	2
V	External address	Full word	Relocatable symbol	V( )	Binary	Left	4	3	4
E	Full word, floating point	Full word	Decimal digits	E' '	Floating-point binary normalized	Right	4	1	8
D	Double word, floating point	Double word	Decimal digits	D' '	Floating-point binary normalized	Right	8	1	8

*Maximum explicit lengths consider duplication factors.

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125

		4	194	304	22	0.000	000	238	418	579	101	562	5					
		8	388	608	23	0.000	000	119	209	289	550	781	25					
		16	777	216	24	0.000	000	059	604	644	775	390	625					
		33	554	432	25	0.000	000	029	802	322	387	695	312	5				
		67	108	864	26	0.000	000	014	901	161	193	847	656	25				
		134	217	728	27	0.000	000	007	450	580	596	923	828	125				
		268	435	456	28	0.000	000	003	725	290	298	461	914	062	5			
		536	870	912	29	0.000	000	001	862	645	149	230	957	031	25			
	1	073	741	824	30	0.000	000	000	931	322	574	615	478	515	625			
	2	147	483	648	31	0.000	000	000	465	661	287	307	739	257	812	5		
	4	294	967	296	32	0.000	000	000	232	830	643	653	869	628	906	25		
	8	589	934	592	33	0.000	000	000	116	415	321	826	934	814	453	125		
	17	179	869	184	34	0.000	000	000	058	207	660	913	467	407	226	562	5	
	34	359	738	368	35	0.000	000	000	029	103	830	456	733	703	613	281	25	
	68	719	476	736	36	0.000	000	000	014	551	915	228	366	851	806	640	625	
	137	438	953	472	37	0.000	000	000	007	275	957	614	183	425	903	320	312	5
	274	877	906	944	38	0.000	000	000	003	637	978	807	091	712	951	660	156	25
	549	755	813	888	39	0.000	000	000	001	818	989	403	545	856	475	830	078	125
1	099	511	627	776	40	0.000	000	000	000	909	494	701	772	928	237	915	039	062	5

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

These powers of 16 are especially useful in determining the value of floating-point numbers.

Label	Operation	Operand	Description
unused	CNOP	a_1,a_2	Adjusts the location counter to a half-word, full-word, or double-word main storage boundary
[symbol]	COM	unused	Defines a control section which is a main storage area common to two or more separately assembled routines
unused	COPY	SYMBOL	Source module, indentified by the operand field symbol, is taken from a library and included in the source program being assembled.
[symbol]	CSECT	unused	Indicates that the following source statements belong to a new control section
unused	DROP	r_1[,...,r_n]	Informs the assembler that the specified registers are not available for base register assignment

Label	Operation	Operand	Description
[symbol]	DSECT	unused	Indicates to the assembler that the statements which follow (dummy control section) are used to redefine a data storage area reserved in the modules being programmed or in another separately assembled module
unused	EJECT	unused	Advances the printer form to the next page for continued listing
unused	END	[e]	Indicates the end of a source module or a macro definition being assembled
unused	ENTRY	symbol[,...symbol]	Declares the symbols defined within the module to which reference is made by other modules
symbol	EQU	e[,a]	Defines symbols (primarily length and value of a symbol)

unused	EXTRN	symbol[,...symbol]	Specifies symbols referred to in the module being assembled but defined in some other module
unused	ICTL	[beginning column] [,ending column] [,continuation column]	Specifies new values for the beginning, ending, and continuation coding columns
unused	ISEQ	[leftmost column ,rightmost column]	Specifies the columns of the source statement which contain the field used for checking the sequence of statements
[symbol]	LTORG	unused	Generates all literals previously defined, but not generated, in the source module
[symbol]	ORG	[e]	Sets or resets the location counter to a specified value

Label	Operation	Operand	Description
unused	PRINT	[{**ON** / OFF}] [,{**GEN** / NOGEN}] [,{DATA / **NODATA**}] [,{**SINGLE** / DOUBLE}]	Enables the programmer to control the contents of the assembly listing
unused	PUNCH	'c_1,...,c_{80}'	Produces a record from a library, at assembly time, by inserting the needed job control commands

unused	REPRO	unused	Reproduces a record in its entirety (columns 1 through 80) at assembly time; the record precedes or follows the object module.
unused	SPACE	[i]	Advances the paper in the printer a specified number of lines
[symbol]	START	a	Defines the name of the first control section, the program name, and the initial value of the location counter
[symbol]	TITLE	'C'	Provides data for the heading which appears at the top of each page of the assembler listing and advances the printer form to a new page

Label	Operation	Operand	Description
unused	USING	$v,r_1[,...,r_n]$	Informs the assembler that a specific register is available for base register assignment in operand addresses and that it will contain a specific value at execution time

Legend:

a = an absolute or relocatable expression
e = a relocatable expression
c = a character string
i = an unsigned decimal integer
r = a register
v = a relocatable or absolute value

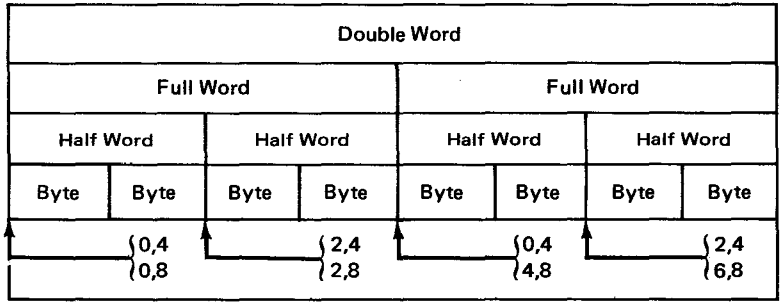
Label	Operation	Operand	Description
	ACTR	se	Used to limit the number of AGO, AIF, and DO directives that may be processed by the assembler within a macro definition or source program
[s_1]	AGO	[s_2]	Unconditionally alters the sequence of source statement processing
s_1	AIF	(b) s_2	Conditionally alters the sequence of source statement processing
s_1	ANOP	unused	Facilitates branching to a point in a program when a statement is unavailable to define the branch destination
[variable symbol]	DO	se	Defines the start of a range of code to be generated repetitively and specifies the number of times it is to be generated
unused	ENDO	unused	Signals the end of range of a DO directive

Label	Operation	Operand	Description
unused	{GBL / GBLA / GBLB / GBLC}	ss[,...ss]	Declares general-purpose, arithmetic, Boolean, or character global set symbols, respectively
unused	{LCL / LCLA / LCLB / LCLC}	ss[,...ss]	Declares general-purpose, arithmetic, Boolean, or character local set symbols, respectively
unused	MEXIT	unused	Assembler terminates processing macro definition and processes statement in source program following macro call instruction that called macro definition containing MEXIT.
unused	MNOTE	{'m' / ,'m' / s,'m'}	Generates error message or comments on printer output listing
unused	PNOTE	* or 'mc', 'com'	Indicates asterisk to be printed in error flag field, or a message or comments character string is enclosed in apostrophes

&s	SET	{ae / ce}	Assigns arithmetic or character-string value to variable symbol declared by LCL or GBL directive
&s	SETA	a	Assigns arithmetic value to variable symbol declared by LCLA or GBLA directive
&s	SETB	b	Assigns binary value of 0 or 1 to variable symbol declared by LCLB or GBLB directive
&s	SETC	c	Assigns character value to variable symbol declared by LCLC or GBLC directive

LEGEND:

a = a valid SETA term or arithmetic combination of valid SETA terms
b = a valid SETB logical expression, a 0, or a 1 that must be enclosed in parentheses
c = a valid SETC operand
ae = an arithmetic expression
ce = a character expression
com = a comments character string
m = a message
mc = a message character
s_1 = a sequence
s_2 = a sequence symbol defined in a following source code statement
sc = a severity code
se = a set expression
ss = a set symbol
&s = a set symbol declared by LCL, or GBL
* = no error statement to be generated


Double Word
Full Word
Full Word
Half Word
Half Word
Half Word
Half Word
Byte
Byte
Byte
Byte
Byte
Byte
Byte
Byte
0,4
0,8
2,4
2,8
0,4
4,8
2,4
6,8

Assembler EBCDIC Character Codes

Bit Positions 4, 5, 6, 7	Bit Positions 0, 1, 2, 3															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DLE	DS(1)		SP	&	–						{(4)	}(4)	\(4)	0
0001	SOH	DC1	SOS(1)				/		a(4)	j	~(4)		A	J		1
0010	STX	DC2	FS(1)	SYN					b	k	s		B	K	S	2
0011	ETX	DC3							c	l	t		C	L	T	3
0100									d	m	u		D	M	U	4
0101	HT		LF						e	n	v		E	N	V	5
0110		BS	ETB						f	o	w		F	O	W	6
0111	DEL		ESC	EOT					g	p	x		G	P	X	7
1000		CAN							h	q	y		H	Q	Y	8
1001		EM						`(4)	i	r	z		I	R	Z	9

Assembler EBCDIC Character Codes (cont)

		Bit Positions 0, 1, 2, 3															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	1010					[	]	¦③	:								
	1011	VT				.	$	.	#								
	1100	FF	FS		DC4	<	*	%	@								
	1101	CR	GS	ENQ	NAK	(	)	—	'								
	1110	SO	RS	ACK		+	;	>	=								
	1111	SI	US	BEL	SUB	¦②	¬②	?	"								

NOTES:

EBCDIC bits are numbered from the left in ascending numerical order:

0 1 2 3 4 5 6 7

① DS, SOS, FS are the control characters for the EDIT instruction and have been assigned for ASCII mode processing so as not to conflict with the corresponding character positions previously assigned in the EBCDIC chart. As these characters are not outside the range as defined in ANSI X3.4 – 1968, they must not appear in external storage media, such as ANSI standard tapes. This presents no difficulty due to the nature of the EDIT instruction.

② The following optional graphics can be substituted in the character set:

v for ¬
¦ for !

③ For 63-character printers, the following substitution is made:

\ for ¦

④ The lowercase alphabet and indicated graphics are introduced by use of the type 0768–02 printer, which prints a 94-character set.

Assembler ASCII Character Codes
(American Standard Code for Information Interchange)

		Bit Positions 7, 6, 5							
		000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Bit Positions 4, 3, 2, 1	**0000**	NUL	DLE	SP	0	@	P	`	p
	0001	SOH	DC1	! ①	1	A	Q	a	q
	0010	STX	DC2	"	2	B	R	b	r
	0011	ETX	DC3	#	3	C	S	c	s
	0100	EOT	DC4	$	4	D	T	d	t
	0101	ENQ	NAK	%	5	E	U	e	u
	0110	ACK	SYN	&	6	F	V	f	v
	0111	BEL	ETB	'	7	G	W	g	w

Assembler ASCII Character Codes (cont)
(American Standard Code for Information Interchange)

		Bit Positions 7, 6, 5							
	1000	BS	CAN	(	8	H	X	h	x
	1001	HT	EM	)	9	I	Y	i	y
	1010	LF	SUB	*	:	J	Z	j	z
	1011	VT	ESC	+	;	K	[	k	{
	1100	FF	FS	,	<	L	\	l	| ②
	1101	CR	GS	-	=	M	]	m	}
	1110	SO	RS	.	>	N	^ ①	n	~
	1111	SI	US	/	?	O	_	o	DEL

③ ④ ⑤

NOTES:

ASCII bits are numbered from the left in descending numerical sequence:

7 6 5 4 3 2 1

① The following optional graphics can be substituted in the character set:

¬ for ^

for |

② For 63-character printers, the following substitution is made:

\ for |

③ Sixty-three printable character set.

④ Graphics available by use of the type 0768-02 printer, which prints a 94-character set (DEL is not a graphic).

⑤ Ninety-four printable character set.

ACK – Acknowledge
BEL – Bell
BS – Backspace
CAN – Cancel
CR – Carriage return
DC1 – Device control 1
DC2 – Device control 2
DC3 – Device control 3
DC4 – Device control 4
DEL – Delete
DLE – Data link escape
DS – Digit select
EM – End of medium
ENQ – Enquiry
EOT – End of transmission
ESC – Escape
ETB – End of transmission block
ETX – End of text
FF – Form feed
FS – Field separator
GS – Group separator
HT – Horizontal tab
LF – Line field
NAK – Negative acknowledge
NUL – Null
RS – Record separator
SI – Shift in
SO – Shift out
SOH – Start of heading
SOS – Start of significance
SP – Space
STX – Start of text
SUB – Substitute
SYN – Synchronous idle
US – Unit separator
VT – Vertical tab

Label	Operation	Operand	Description
name	BCW	device-cmd-code [,data-addr] [,data-flag] [,data-byte-count] [,repl-addr] [,repl-flag] [,repl-byte-count] [,control-flag]	Generates buffer control word
name	CCB	PIOCB-name, {BCW-name / CCW-name} [, {PUB-entry / 0}] [, {error-option / X'00'}]	Generates command control block
name	CCW	device-cmd-code [,data-addr] [,flag] [,data-byte-count]	Generates channel control word
[name]	EXCP	{CCB-name / (1)} [,C]	Executes channel program

name	PIOCB	[{FCB-length / MAX / 16}]	Generates input/output control block
[name]	RDFCB	{PIOCB-name / (1)} [, {error-addr / (r)}]	Reads file control block
[name]	SWAP	{CCB-name / (1)}	Accesses next physical input/output device

NOTE:
Shaded operands indicate default options.

Label	Operation	Operand	Description
[name]	WAIT	{ ALL / CCB-name / (1) } [, { branch-addr / (15) }]	Waits for one or all input/output requests to complete
[name]	WAITM	{ CCB-name-1, CCB-name-2[,...,CCB-name-n] / list-name / (1) }	Waits for one of several input/output requests to complete

Label	Operation	Operand	Description
[name]	ALLOC	{FCB-name / filename-addr / (1)} [, {error-addr / (r)}] [, {vol-seq-no,OLD,NOFCB / (0)}]	Assigns space to a new disc file or to an existing disc file
[name]	EXTEND	{filename-addr / (1)} [, {error-addr / (r)}] [, {01 / 80} ,vol-seq-no / (0)]	Assigns additional space to an existing disc file

Label	Operation	Operand	Description
[name]	OBTAIN	{param-list / (1)} [,{error-addr / (r)}] [,{vol-seq-no / (1)}] [,FCBCORE]	Accesses VTOC user block
[name]	RENAME	{param-list / (1)} [,{error-addr / (r)}] [{vol-seq-no / (1)}]	Renames a disc file
[symbol]	SCRTCH	{FCB-name / (1)} [,{PREFIX / ALL / (0)}] [,{error-addr / (r)}]	Releases all or part of disc file

NOTE:
Shaded operands indicate default options.

Label	Operation	Operand	Description
[name]	CLOSE	{ filename-1 [,...,filename-n] / (1) }	Closes disc file
[symbol]	CNTRL	{ filename / (1) } ,code	Initiates nondata operations on a tape unit
filename	DTFPF	PCA1 = partition-name [. . . ,PCA7 = partition-name] [,ALINE=YES] [,ERROR=symbol] [,EXTENTS=n] [,FCB=YES] [,LIBUP-YES] [,WAIT=YES]	Defines partitioned file
[name]	GET	{ file-name / (1) } , { TCA-name / (0) }	Retrieves next logical record

System Access Technique (SAT)

Label	Operation	Operand	Description
[name]	OPEN	{file-name-1[,...,file-name-n] / (1)}	Opens disc file
partition-name	PCA	BLKSIZE=n [,EODADDR=symbol] [,FORMAT=NO] IOAREA1=symbol [,KEYLEN=n] [,LACE=n] [,LBLK=n] [,SEQ=YES] [,SIZE=n] [,UOS=n] [,VERIFY=YES]	Defines partition control appendage
[name]	PUT	{file-name / (1)} , {TCA-name / (0)}	Outputs logical record
[name]	READE READH	{file-name / (1)} , {PCA-name / (0)}	Searches track by key equal Searches track by key equal or higher
[name]	SEEK	{file-name / (1)} , {PCA-name / (0)}	Accesses physical record
[name]	WAITF	{file-name / (1)}	Waits for record transfer

Label	Operation	Operand	Description
		Task Management	
[name]	ATTACH	{ECB-name / (1)}, {entry-point-name / (0)} [, {error-addr / (r)}] [,n]	Creates and activates additional tasks
[name]	AWAKE	[{ECB-name / (1)}]	Reactivates existing, nonactive task
[name]	CHAP	{n / (1)}	Changes priority of task
[name]	DETACH	{ECB-name / (1)}, {error-addr / (r)}	Terminates task normally

Multitasking (cont)

Label	Operation	Operand	Description
		Task Management	
		[,{error-addr / (r)}] [,n]	
[name]	ECB		Generates event control block
[name]	TYIELD		Deactivates a task
		Task Synchronization	
[symbol]	POST		Activates waiting task
[name]	WAIT	{ECB-name / (1)}	Waits for task request to complete
[name]	WAITM	{ECB-name-1,ECB-name-2 [,...,ECB-name-n] / list-name / (1)}	Waits for one of several task requests to complete

Label	Operation	Operand	Description
		Program Loader	
[name]	FETCH	{phase-name / (1)} [, {entry-point-name / (0)}] [,R]	Loads program phase and branches
[name]	LOAD	{phase-name / (1)} [, {load-addr / (0)}] [, {error-addr / (r)}] [,R]	Loads program and returns control
[name]	LOADI	{phase-name / (1)}, {work-area-addr / (0)}	Locates program phase and stores its phase header in work area

Label	Operation	Operand	Description
		Program Loader	
		[, {work-area-length / 13}] [, {error-addr / (r)}] [,R]	
[name]	LOADR	{phase-name / (1)} [, {load-addr / (0)}] [, {error-addr / (r)}] [,R]	Loads program phase, relocates address constants, and returns control
		Job and Task Termination	
[name]	CANCEL	[{error-code / (0)}]	Terminates job abnormally
[name]	EOJ		Terminates job step normally

Timer Services			
[name]	GETIME	[{M | S}]	Obtains current time and date
[name]	SETIME	{time-interval | (1)} [,WAIT] [, {M | S}]	Sets elapsed time counter for the requesting task
Island Code Linkage			
[name]	EXIT	{IT | OC | PC}	Exits from island code subroutine

NOTE:
Shaded operands indicate default options.

Program Management (cont)

Label	Operation	Operand	Description
		Island Code Linkage	
[name]	STXIT	{AB / IT / PC} [, {entry-point / (1)} [, {save-area / (0)}]]	Links island code subroutine when used for program check, abnormal termination, or interval timer island code linkage
[name]	STXIT	OC [, {entry-point,save-area,msg-area,length / (1)}]	Links island code subroutine when used for unsolicited operator communications linkage
		System Information Control	
[name]	GETCOM	{to-addr / (1)}	Retrieves data from job common area
[name]	GETINF	{PRE / PUB / SIB / TCB}, {work-area / (1)}, bytes, displacement	Retrieves data from system control tables
[name]	PUTCOM	{from-addr / (1)}	Places data in job common area

Control Stream Reader			
[name]	GETCS	{ input-area / (1) } [, { number-of-records / 0 / (1) }] [, { error-addr / (r) }]	Retrieves embedded data file submitted in job control stream
[name]	SETCS	{ NEXT / data-set-no / pointer / (1) } [{ R / S }] [, { error-addr / (r) }]	Resets pointer to embedded data file

NOTE:

Shaded operands indicate default options.

Label	Operation	Operand	Description
Storage Displays			
[name]	DUMP	[{completion-code / (0)}]	Prints out job in main storage and terminates job step
[name]	SNAP	{start-addr-1,end-addr-1 [...,start-addr-n,end-addr-n] / (1)}	Prints out portions of main storage and returns control

Label	Operation	Operand	Description
[name]	GETMSG	{buff-addr-1 / (1)} [,{msg-length / (0) / 60}] [,{error-addr / (r)3}] [,not applicable] [,{buff-addr-2 / (r)4}] [,{buff-length-2 / (r)5}]	Retrieves message from canned message file
[symbol]	WTL	{buff-addr / (1)} [,{msg-length / (0) / 60}] [,{error-addr / (r)3}]	Writes message to the system log file

Message Retrieval, Logging, and Display (cont)

Label	Operation	Operand	Description
[symbol]	WTLD	{buff-addr-1 / (1)} [,{msg-length / (0) / 60}] [,{error-addr / (r)3}] [,REPLY] [,{buff-addr-2 / (r)4}] ,{buff-length-2 / (r)5}	Writes message into system log file after displaying on system console

NOTE:

Shaded operands indicate default options.

Label	Operation	Operand	Description
[symbol]	OPR	{ buff-addr-1 / (1) } ,[{ msg-length / (0) / 60 }] ,[{ error-addr / (r)3 }] [,REPLY],[{ buff-addr-2 / (r)4 }] [{ buff-length-2 / (r)5 / 60 }]	Displays message to operator on system console
Tape SAT File Interface			
filename	SAT	TCA=TCA-name [,CKPTREC=YES] [,ERROR=error-addr] [,FCB=YES] [,WAIT=YES]	Defines a magnetic tape file to be processed by SAT

Label	Operation	Operand	Description
		Tape SAT File Interface	
TCA-name	TCA	IOAREA1=area-name ,BLKSIZE=n [,CLRW={NORWD / RWD}] [,EOFADDR=end-of-data-error] [,FILABL={STD / NSTD / NO}] [,LBLK=n] [,OPRW=NORWD] [,READ={FORWARD / BACK}] [,REWIND={UNLOAD / NORWD}] [,TPMARK=NO] [,TYPEFLE=OUTPUT]	Defines the logical attributes of a magnetic tape file to be processed by TSAT

Program Linkage			
[symbol]	{ CALL / VCALL }	{ entry-point / (15) } [, { (param-1,...,param-n) / list-address / (1) }]	Pass control from a program to a specified entry point in another program
[symbol]	RETURN	[(r1,r2)] [,T] [,SA= { savearea-name / * }]	Marks the exit point of the called program
[symbol]	SAVE	[(r1,r2)] [,T] [,COVER= { r / (r1,r2,...,rn) }] [,COVADR= { base-addr / * }] [,SA=savearea-name]	Marks the entry point of the called program

NOTE:
Shaded operands indicate default options.

User-Operator Communications (cont)

Label	Operation	Operand	Description
		Checkpoint Facility	
[symbol]	CHKPT	filename [,restart-addr] [,list-name] [,error-addr]	Writes a series of checkpoint records to a specified checkpoint file
filename	DDCPF		Defines a disc file to which checkpoint records are to be written
[symbol]	DCPOPN	{ filename (1) }	Opens a disc checkpoint file defined by a DDCPF macro instruction
[symbol]	DCPCLS	{ filename (1) }	Closes a disc checkpoint file defined by a DDCPF macro instruction

list-name	DCFLT	{ (disc-PIOCB-1) } { (tape-PIOCB-1,tmc-1,bc-1) } [{ ,(...),(disc-PIOCB-n) } { (...,...,...),(tape-PIOCB-n,tmc-n,bc-n) }]	Generates a table of IOCS files

Label △ Operation △ Operand	Description
//[symbol] ALTER [phase-name] [,address] [,change] [RESET | ORG]	Introduces load module alterations at execution time
//[symbol] CAT lfdname[,catpw] [,SCR] [,GEN=nn]	Causes a file to be cataloged
//[symbol] CR	Allows input from card reader to be inserted in control stream
//ignored DATA FILEID= file-identifier	Loads card data to a spool file
//[symbol] DECAT lfdname[,catpw] [,SCR] [,GEN]	Causes a file to be removed from the catalog
//[symbol] DVC {nnn | RES | RUN} [{addr | ALT | IGNORE | OPT}]	Requests assignment of peripheral devices to a job
//[symbol] DST dest-1[,dest-2,...,dest-n]	Supplies the destination identification of a remote device for spool output

//[symbol] EQU lun-1,type-1[,lun-2,type-2,...,lun-n,type-n]	Equates logical unit numbers to specific code for device type
//[symbol] EXEC program-name [,{library-name / YRUN / **YLOD**}] [,switch-priority]	Provides the name of the load module to be executed
//[symbol] EXT [{DA / IS / NI / **SQ** / ST / nn(id)}] [,{C / CF / F}] [,{inc / 0 / **1**}] { [[,{addr / **BLK** / CYL / PRI / SUB}]] [,{mi / (bi[,ai]) / pi%[,ci])}] [,{mj / (bj[,aj]) / (pj%[,cj])} ,...] [,OLD] / [,OLD] }	Obtains disc space; provides information needed to create new files or extend existing files stored on discs

Label △ Operation △ Operand	Description
//[symbol] FIN	Terminates card reader operation
//[symbol] FREE lfdname-1 [[(DEV)],...,lfdname-n[(DEV)]]	Releases peripheral devices assigned to job and not required in later job steps
//[symbol] GBL set-id-1[=init-1] [,set-id-2[=init-2] ,...,set-id-n[=init-n]]	Assigns global status to a set symbol
//[symbol] GO destination	Causes an unconditional branch, in a foward direction, to another job control statement
//[symbol] IF (a op b) destination	Causes a conditional branch, in a forward direction, to another job control statement
//[symbol] JOB jobname [,{P / H / N}] [,min] [,max] [,{tasks / 1}] [,max-time] [,(op-list-1,...,op-list-n)] [,acct-no] [,nXm]	Indicates the beginning of a job

//[symbol] JSET value	Assigns local status to a set symbol
//[symbol] LBL { file-identifier / 'file-identifier' } [, { file-serial-number / VCHECK }] [,expiration-date] [,creation-date] [, { file-sequence-number / 1 }] [, { generation-number / 1 }] [, { version-number / 1 }]	Supplies label information for files on disc and tape volumes for use by data management
//[symbol] LBL { [qual/] levelid-1 [.levelid-2...[.levelid-n]] [{ nn / +m / -n }] [(rpw/wpw)] / '[qual/] levelid-1 [.levelid-2...[.levelid-n]] [{ nn / +m / -n }] [(rpw/wpw)]' } [,expiration-date] [,creation-date] [, { file-sequence-number / 1 }] [, { generation-number / 1 }] [, { version-number / 1 }]	Supplies file catalog information for files on disc and tape volumes

Label △ Operation △ Operand	Description
//[symbol] LCB {C'char-string-1' / X'hex-string-1'} [,{C'char-string-2' / X'hex-string-2'},...,{C'char-string-n' / X'hex-string-n'}] [,NUMBCHAR=n] [,CARTID={C'c' / X'aa'}] [,SPACE={C'c' / X'aa' / **X'40'**}] [,TYPE={**0773** / 0768 / 0770}] [,MISM={**IGNORE** / REPORT}] [,DUAL={C'abababab' / C'cccc' / X'yyyyyyyy' / X'xxyyxxyyxxyyxxyy'}] [,MISMCHAR={C'c' / X'aa' / **X'40'**}] [,CARTNAME=symbol] [NAME=stand]	Allows overriding of SYSGEN load code buffer parameters

//[symbol] LFD {filename / *filename} [,{n / 0}] [{ACCEPT / EXTEND / INIT / RELOD}]	Links file information in control stream with data management file definition
//[symbol] MTC lfdname,{BB,nn / BM,nn / FB,nn / FM,nn / WM,nn / RL / RU}	Positions tape volumes prior to or after the execution of a job step
//[symbol] NOP	Inserts labels to be used as targets of branch statements
//[symbol] OPR comment-line (1)	Displays messages at the system console
//[symbol] OPTION p-1[,...,p-n] [BOF,DOF,DUMP,GO,JOBDUMP, LINK,NODUMP,REPEAT, SCAN,SIG,SUB,SYSDUMP,TRACE,XUF] (2)	Allows user to specify certain optional software features and operating environment

Label △ Operation △ Operand	Description
//[symbol] PARAM operand-1,...,operand-n	Makes available to user program parameters required at job step execution time
//[symbol] QUAL qualifier	Appends a qualifier to subsequent file identifiers in the job
//[symbol] RST file-name,checkpoint-id,number [,jobname[(rename)] [,pri]] [,key-1=val-1,...,key-n=val-n]	Restarts a program from a checkpoint
//[symbol] RUN jobname[(rename)] [,pri] [,key-1=val-1,...,key-n=val-n]	Initiates the named job
//[symbol] SCR lfdname [, { DATE[,yyddd] / EXT,extent-number / PRE[,aaaa] }]	Scratches files and extents

//[symbol] SET DATE,yy/mm/dd[,t-date] [,d-date] //[symbol] SET UPSI,switch-setting //[symbol] SET COMREG,char-string	Sets or modifies date field, user program switch indicator, or communications region in the job preamble
//[symbol] SFT module-1[,...,module-n]	Specifies data management common code modules or symbionts supporting user interfaces and required by job
//[symbol] SKIP target-label[,mask]	Identifies the target control statement of a branch control statement
//[symbol] SPL [{ HOLD / DUMP / RETAIN }] [,nXm] [, { no-cop / **1** }] [, { no-skpcode / **7** }] [, { max-rec / **5000** }] [,forms]	Controls the spool output

Label △ Operation △ Operand	Description
//[symbol] VFB LENGTH=lines [,DENSITY={6 / 8}] [FORMNAME=symbol] [,USE=stand] [,TYPE={9300 / 0768 / 0770 / 0773}] [,HP={n / 1}] [,OVF=(line-1,...,line-n)] [,OVf2=(line-1,...,line-n)] [,CD1-(line-1,...,line-n0),...[,CD15=(line-1,...,line-n)]	Overrides system vertical format buffer
//[symbol] VOL {C / CMcc / Mcc / SCR / volsn-1 [{(S) / (NS) / (NOV) / (PREP)}]} [{volsn-1 [{(S) / (NS) / (NOV) / (PREP)}] / volsn-2 [{(S) / (NS)}]}]	Supplies volume serial numbers of data and program volumes to be accessed by the job

$\left[\left\{\begin{array}{l}\text{volsn-2}\left[\left\{\begin{array}{l}\underline{\text{(S)}}\\(\text{NS})\end{array}\right\}\right]\\\text{volsn-3}\left[\left\{\begin{array}{l}\underline{\text{(S)}}\\(\text{NS})\end{array}\right\}\right]\end{array}\right\},\ldots\right]$	
/$	Start of data
/*	End of data
/&	End of job

NOTES:

1. If comment includes embedded blanks, it must be enclosed by apostrophes.

2. Parameters inside the brackets may be specified in any order or combination for $p_1^n \ldots, p_n^u$ except the REPEAT parameter should not be used when LINK is used with GO.

Characters Used To Specify Mode Setting on VOL Statement

Used with UNISERVO 12/16 Magnetic Tape Volumes					
Tape	cc	Bytes per Inch	Parity	Translate Feature	Convert Feature
7-track	10	200	Odd	Off	On
	20	200	Even	Off	Off
	28	200	Even	On	Off
	30	200	Odd	Off	Off
	38	200	Odd	On	Off
	50	556	Odd	Off	On
	60	556	Even	Off	Off
	68	556	Even	On	Off
	70	556	Odd	Off	Off
	78	556	Odd	On	Off
	90	800	Odd	Off	On
	A0	800	Even	Off	Off
	A8	800	Even	On	Off
	B0	800	Odd	Off	Off
	B8	800	Odd	On	Off
9-track	C8	800	Odd	Off	Off
	C0*	1600	Odd	Off	Off

Characters Used To Specify Mode Setting on VOL Statement (cont)

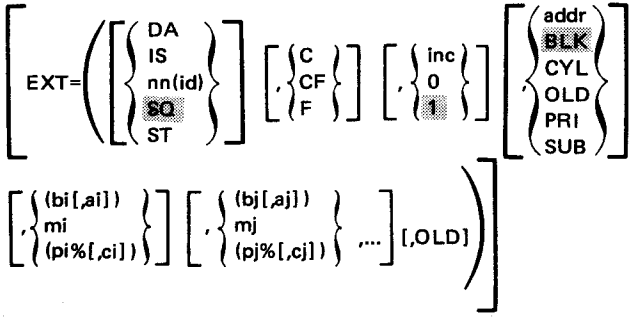
Used with UNISERVO VI-C Magnetic Tape Volumes					
Tape	cc	Bytes per Inch	Parity	Translate Feature	Convert Feature
7-track	10	200	Odd		On
	20	200	Even		Off
	30	200	Odd		Off
	50	556	Odd		On
	60	556	Even		Off
	70	556	Odd		Off
	90	800	Odd		On
	A0	800	Even		Off
	B0	800	Odd		Off
9-track	C0	800	Odd		Off

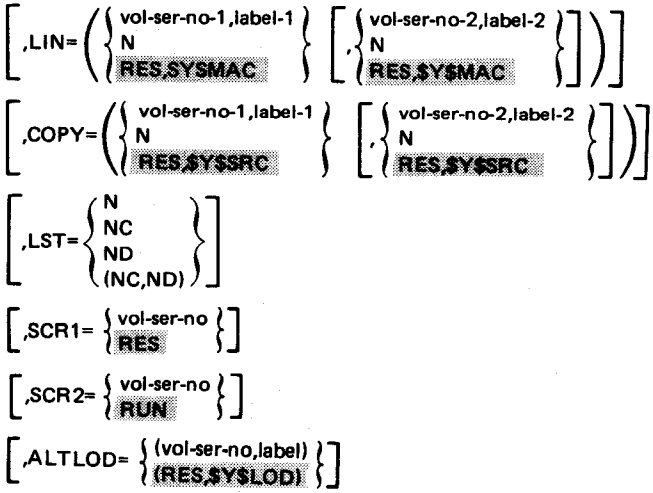
*Also applies to the UNISERVO 20 Magnetic Tape Subsystem

NOTE:

The mode always must be specified for tape devices with phase-encoded capability.

Label △ Operation △ Operand	Description
//symbol procname p_1,p_2,...,p_n k_i=v_i,k_j=r_j,...,k_m=v_m	Format of all user job control procedure call statements
// lfdname ACCESS { lblname (lblname [,{n / S}] [,{ACCEPT / EXTEND / INIT / RELOD}]) } [,{DVC=nn,VOL=volsn / VOL=volsn}]	Generates the job control statements required to assign a device to a job step so that the file on that device can be accessed at job execution time
// lfdname ALLOC { lblname (lblname [,{noext / S}] [,{ACCEPT / EXTEND / INIT / RELOD}]) } [,{DVC=nn,VOL=volsn / VOL=volsn}]	Generates job control statements required to assign a disc to a job step and allocates space on that disc for the file

	
//[symbol] ASM / ASML / ASMLG [PRNTR= lun / 20] [,IN= (vol-ser-no,label) / (RES) / (RES,label) / (RUN,label)] [,OUT= (vol-ser-no,label) / (RES,label) / (RUN,label) / (N) / (RUN,YRUN)]	Generates the job control statements needed to run the assembler language translator

Label △ Operation △ Operand	Description
(cont) 	Generates the job control statements needed to run the assembler language translator

```
//[symbol] { COBOLB   }
           { COBOLBL  }
           { COBOLBLG }
           { COBOL    }
           { COBOLL   }
           { COBOLLG  }

[PRNTR= { lun }]  [,IN= { (vol-ser-no,label) }]
        { 20  }        { (RES)              }
                       { (RES,label)        }
                       { (RUN,label)        }

[,OBJ= { (vol-ser-no,label) }]
       { (RES,label)        }
       { (RUN,label)        }
       { (RUN,$Y$RUN)       }

[,LIN= { (vol-ser-no,label) }]
       { (RES,$Y$SRC)       }

[,OUT=(p-1,...,p-n)]  [,LST=p-1,...,p-n)]

[,SCR1= { vol-ser-no }]  [,SCR2= { vol-ser-no }]
        { RES        }           { RES        }

[,SCR3= { vol-ser-no }]
        { RUN        }

[,ALTOD= { (vol-ser-no label) }]
         { (RES,$Y$LOD)       }
```

Generates the job control statements needed to run the COBOL language translator

Label △ Operation △ Operand	Description
//[symbol] DVCVOL { vol-ser-no / RES / RUN } [,lun]	Assigns the same logical unit number to a disc volume having different files used in a job
//[symbol] DVCVTP vol-ser-no[,lun]	Assigns the same logical unit number to a tape volume having different files used in a job
//[symbol] { FORT / FORTL / FORTLG } [PRNTR= { lun / **20** }] [,IN= { (vol-ser-no,label) / (RES) / (RES,label) / (RUN,label) }] [,OUT= { (vol-ser-no,label) / (RES,label) / (RUN,label) / NO / **(RUN,YRUN)** }] [,LST=k] [,SCR1= { vol-ser-no / **RES** }] [,ALTLOD= { (vol-ser-no,label) / **(RES,YLOD)** }]	Generates the job control statements needed to run the FORTRAN language translator

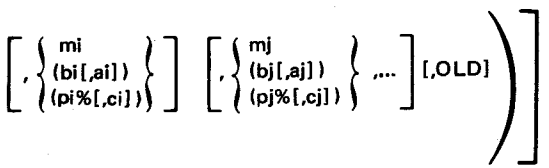
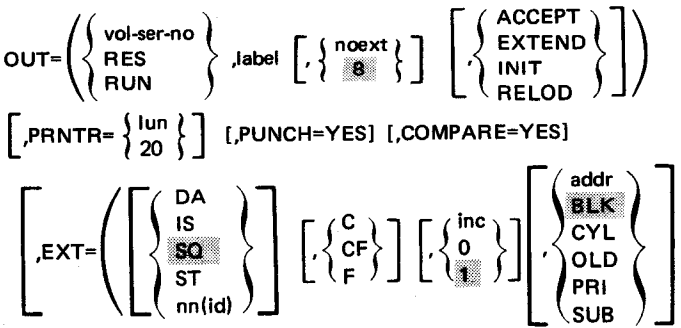
```
//[symbol] { LINK  } [input-module-name] [,PRNTR= { lun }]
           { LINKG }                             { 20  }
   [,IN= { (vol-ser-no,label) }]
         { (RES)              }
         { (RES,label)        }
         { (RUN,label)        }
         { (RUN,$Y$RUN)       }
   [,OUT= { (vol-ser-no,label) }]
          { (RES,label)        }
          { (RUN,label)        }
          { (N)                }
          { (RUN,$Y$RUN)       }
   [,RLIB= { (vol-ser-no,label) }]
           { (RES,label)        }
           { (RUN,label)        }
   [,ALIB= { (vol-ser-no,label) }]
           { (RES,label)        }
           { (RUN,label)        }
   [,SCR1= { vol-ser-no } ] [,STD= { YES } ]
           { RES        }          { NO  }
   [,ALTLOD=(vol-ser-no,label)]
```

Generates the job control statements needed to execute the linkage editor

Label △ Operation △ Operand	Description
//[symbol] { RPG / RPGL / RPGLG } [PRNTR= { lun / **20** }] [,IN= { (vol-ser-no,label) / (RES) / (RES,label) / (RUN,label) }] [,OUT= { (vol-ser-no,label) / (RES,label) / (RUN,label) / (N) / **(RUN,YRUN)** }] [,LST= { K / M / N / S }]	Generates the job control statements needed to run the RPG language translator

[,SCR1= { vol-ser-no / **RES** }] [,SCR2= { vol-ser-no / **RUN** }]	
//ignored UDT IN= ({ vol-ser-no / RES / RUN } ,label [, { noext / **8** }] [,ACCEPT]) ,OUT=(vol-ser-no,label) [,PRNTR= { lun / **20** } [,PUNCH=YES] [,COMPARE=YES]	Generates the job control statements for the device assignment sets needed by the data utility routine to copy or compare a disc file to a tape file

Label △ Operation △ Operand	Description
// ignored UDD IN=({ vol-ser-no / RES / RUN } ,label [,{ noext / **8** }] [,ACCEPT]) ,OUT=({ vol-ser-no / RES / RUN } ,label [,{ noext / **8** }] [,{ ACCEPT / EXTEND / INIT / RELOD }]) [,PRNTR= { lun / **20** }] [,PUNCH=YES] [,COMPARE=YES] [,EXT=([{ DA / IS / **SQ** / ST / nn(id) }] [,{ C / CF / F }] [,{ inc / 0 / **1** }] [,{ addr / **BLK** / CYL / OLD / PRI / SUB }]]	Generates the job control statements for the device assignment sets needed by the data utility routine to copy or compare one disc file to another disc file

	
//ignored UTD IN=(vol-ser-no,label), 	Generates the job control statements for the device assignment set needed by the data utility routine to copy or compare a tape file to a disc file

Label △ Operation △ Operand	Description
(cont) [,{mi / (bi[,ai]) / (pi%[,ci])}] [,{mj / (bj[,aj]) / (pj%[,cj])},...] [,OLD])]	
//[symbol] WRTBIG 'block-1' [,'block-2',...,'block-8'] [,IN={(vol-ser-no,label) / (RES,label) / (RUN,label) / **(RES,YLOD)**}] [,LUN={[nn] [,lfdname] / **(20) [,PRNTR]**}]	Prints block letters before the print output of a job
//[label] WORKn [{DVC=nn,VOL=volsn / VOL=volsn}] [,BLK=bb] [,EXTSP={nn / **16**}] [,SECALL={nn / **1**}]	Sets up temporary files for a job step

Label	Operation	Operand	Description
[symbol]	PROC	[pos] $\left[,\left\{\begin{matrix}n\\0\end{matrix}\right\}\right]$ [,k_1,...k_n]	Signals start of procedure definition
symbol	NAME	parameter	Supplies name by which procedure is referenced or called
[symbol]	END		Indicates end of procedure definition

For sense byte information see the 90/30 I/O sense data byte definitions summary, UP-8176 (current version).