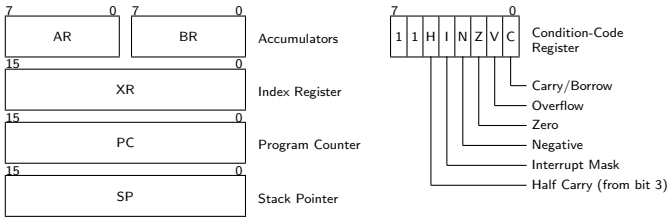
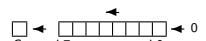
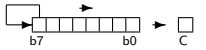
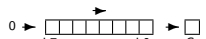
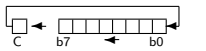
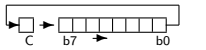


ARIZONA STATE UNIVERSITY

Embedded–Systems Laboratory

Motorola M6800 Microprocessor

ACCUMULATOR and MEMORY		ADDRESSING MODES														BOOL/ARITH OPERATION	CONDITION CODES[1]					
		IMMEDIATE			DIRECT			INDEXED			EXTENDED			INHERENT			(Each register label refers	5	4	3	2	1	0
OPERATIONS	MNEM.	OP	∼	#	OP	∼	#	OP	∼	#	OP	∼	#	OP	∼	#	to contents of register)	H	I	N	Z	V	C
Add Accumulators	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A ∧ M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B ∧ M → B	•	•	↕	↕	R	•
Arithmetic Shift Left	ASL							68	7	2	78	6	3					•	•	↕	↕	2	↕
	ASLA													48	2	1		•	•	↕	↕	2	↕
	ASLB													58	2	1		•	•	↕	↕	2	↕
Arithmetic Shift Right	ASR							67	7	2	77	6	3					•	•	↕	↕	2	↕
	ASRA													47	2	1		•	•	↕	↕	2	↕
	ASRB													57	2	1		•	•	↕	↕	2	↕
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A ∧ M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B ∧ M	•	•	↕	↕	R	•
Compare Accumulators	CBA													11	2	1	A – B	•	•	↕	↕	↕	↕
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A – M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B – M	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				$\overline{M} \rightarrow M$	•	•	↕	↕	R	S
	COMA													43	2	1	$\overline{A} \rightarrow A$	•	•	↕	↕	R	S
	COMB													53	2	1	$\overline{B} \rightarrow B$	•	•	↕	↕	R	S
Decimal Adjust, A	DAA													19	2	1	Convert Binary Addition of BCD	•	•	↕	↕	↕	3
Decrement	DEC							6A	7	2	7A	6	3				M – 1 → M	•	•	↕	↕	↕	•
	DECA													4A	2	1	A – 1 → A	•	•	↕	↕	↕	•
	DECB													5A	2	1	B – 1 → B	•	•	↕	↕	↕	•
Exclusive Or	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	↕	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	↕	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Logical Shift Right	LSR							64	7	2	74	6	3					•	•	R	↕	2	↕
	LSRA													44	2	1		•	•	R	↕	2	↕
	LSRB													54	2	1		•	•	R	↕	2	↕
Negate	NEG							60	7	2	70	6	3				00 – M → M	•	•	↕	↕	↕	↕
	NEGA													40	2	1	00 – A → A	•	•	↕	↕	↕	↕
	NEGB													50	2	1	00 – B → B	•	•	↕	↕	↕	↕
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A ∨ M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B ∨ M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	$A \rightarrow M_{SP}$, SP – 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	$B \rightarrow M_{SP}$, SP – 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, $M_{SP} \rightarrow A$	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, $M_{SP} \rightarrow B$	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3					•	•	↕	↕	2	↕
	ROLA													49	2	1		•	•	↕	↕	2	↕
	ROLB													59	2	1		•	•	↕	↕	2	↕
Rotate Right	ROR							66	7	2	76	6	3					•	•	↕	↕	2	↕
	RORA													46	2	1		•	•	↕	↕	2	↕
	RORB													56	2	1		•	•	↕	↕	2	↕
Subtract Accumulators	SBA													10	2	1	A – B → A	•	•	↕	↕	↕	↕
Subtract with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A – M – C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B – M – C → B	•	•	↕	↕	↕	↕
Store Accumulator	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A – M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B – M → B	•	•	↕	↕	↕	↕
Transfer Accumulator	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test Value	TST							6D	7	2	7D	6	3				M – 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A – 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B – 00	•	•	↕	↕	R	R

January 15, 2003

XR and SP		ADDRESSING MODES: IMMEDIATE			DIRECT			INDEXED			EXTENDED			INHERENT			BOOL/ARITH OPERATION	CONDITION CODES[1]					
OPERATIONS	MNEM.	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	(Each register label refers to contents of register)	5 H	4 I	3 N	2 Z	1 V	0 C
Compare XR	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_{MS} - M, X_{LS} - (M + 1)$	•	•	4	↕	5	•
Decrement SP	DES													34	4	1	SP – 1 → SP	•	•	•	•	•	•
Decrement XR	DEX													09	4	1	X – 1 → X	•	•	•	↕	•	•
Increment SP	INS													31	4	1	SP + 1 → SP	•	•	•	•	•	•
Increment XR	INX													08	4	1	X + 1 → X	•	•	•	↕	•	•
Load SP	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_{MS}, (M + 1) \rightarrow SP_{LS}$	•	•	↕	↕	R	•
Load XR	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_{MS}, (M + 1) \rightarrow X_{LS}$	•	•	↕	↕	R	•
Store SP	STS				9F	5	2	AF	7	2	BF	6	3				$SP_{MS} \rightarrow M, SP_{LS} \rightarrow (M + 1)$	•	•	↕	↕	R	•
Store XR	STX				DF	5	2	EF	7	2	FF	6	3				$X_{MS} \rightarrow M, X_{LS} \rightarrow (M + 1)$	•	•	↕	↕	R	•
SP + 1 → XR	TSX													30	4	1	SP + 1 → X	•	•	•	•	•	•
XR – 1 → SP	TXS													35	4	1	X – 1 → SP	•	•	•	•	•	•

JUMP and BRANCH		RELATIVE			INDEXED			EXTENDED			INHERENT				5	4	3	2	1	0
OPERATIONS	MNEM.	OP	~	#	OP	~	#	OP	~	#	OP	~	#	BRANCH TEST	H	I	N	Z	V	C
Branch if Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch if Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch if Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch if Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch if Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch if Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch if Equal	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch if Not Equal	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch if < (Signed)	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch if ≤ (Signed)	BLE	2F	4	2										Z ∨ (N ⊕ V) = 1	•	•	•	•	•	•
Branch if ≥ (Signed)	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch if > (Signed)	BGT	2E	4	2										Z ∨ (N ⊕ V) = 0	•	•	•	•	•	•
Branch if Lower or Same (Unsigned)	BLS	23	4	2										C ∨ Z = 1	•	•	•	•	•	•
Branch if Higher (Unsigned)	BHI	22	4	2										C ∨ Z = 0	•	•	•	•	•	•
Branch Always	BRA	20	4	2										Branch Relative	•	•	•	•	•	•
Branch to Subroutine	BSR	8D	8	2										Push PC; Branch Relative	•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				Jump Absolute	•	•	•	•	•	•
Jump to Subroutine	JSR				AD	8	2	BD	9	3				Push PC; Jump Absolute	•	•	•	•	•	•
No Operation	NOP										01	2	1	Only Advance Program Counter	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1	Pull Interrupt Stack Frame	↕	↕	↕	↕	↕	↕
Return From Subroutine	RTS										39	5	1	Pull PC	•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	Push Interrupt Stack Frame; Vector	•	S	•	•	•	•
Wait for Interrupt	WAI										3E	9	1	Push Interrupt Stack Frame; Wait	•	6	•	•	•	•

CONDITION-CODE		INHERENT			BOOLEAN	5	4	3	2	1	0
OPERATIONS	MNEM.	OP	~	#	OPERATION	H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
AR → CC	TAP	06	2	1	A → CC	↕	↕	↕	↕	↕	↕
CC → AR	TPA	07	2	1	CC → A	•	•	•	•	•	•

Condition-Code Notes:

1. Bits 7 and 6 of CC are always set.
2. Sets CC.V = N ⊕ C after shift has occurred.
3. CC.C = 1 if BCD result > 99_{10}; otherwise, CC.C = 0.
4. CC.N = Sign bit from subtraction of MS bytes.
5. CC.V = Two's-complement overflow from subtraction of MS bytes.
6. Sets CC.I when interrupt occurs. If previously set, a NonMaskable Interrupt is required to exit from the wait state.

Interrupt Vectors

FFF8	IRQ	MS
FFF9	IRQ	LS
FFFA	SWI	MS
FFFB	SWI	LS
FFFC	NMI	MS
FFFD	NMI	LS
FFFE	Reset	MS
FFFF	Reset	LS

Interrupt Stack

SP	
SP + 1	CC
SP + 2	BR
SP + 3	AR
SP + 4	XR_{MS}
SP + 5	XR_{LS}
SP + 6	PC_{MS}
SP + 7	PC_{LS}

Legend:

OP	Operation Code (Hexadecimal)	$\overline{M}$	One's Complement of M
~	Number of MPU Cycles	0	Bit = Zero
#	Number of Program Bytes	00	Byte = Zero
+	Arithmetic Plus	CC	Condition-Code register
–	Arithmetic Minus	↕	Set if true, cleared otherwise
×	Arithmetic Multiply	•	Not Affected
∧	Boolean AND	R	Reset Always
∨	Boolean Inclusive OR	S	Set Always
⊕	Boolean Exclusive OR	H	Half Carry from bit 3
→	Transfer Into	I	Interrupt Mask
LS	Least Significant	N	Negative (sign bit)
MS	Most Significant	Z	Zero (byte)
M	Memory Operand	V	Overflow, Two's Complement
M_{SP}	Mem. byte that SP addresses	C	Carry from bit 7

Powers of Two

n	2^n	$\$2^n$
0	1	$01
1	2	$02
2	4	$04
3	8	$08
4	16	$10
5	32	$20
6	64	$40
7	128	$80

n	2^n	$\$2^n$
8	256	$0100
9	512	$0200
10	1,024	$0400
11	2,048	$0800
12	4,096	$1000
13	8,192	$2000
14	16,384	$4000
15	32,768	$8000

n	2^n	$\$2^n$
16	65,536	$01,0000
17	131,072	$02,0000
18	262,144	$04,0000
19	524,288	$08,0000
20	1,048,576	$10,0000
21	2,097,152	$20,0000
22	4,194,304	$40,0000
23	8,388,608	$80,0000

n	2^n	$\$2^n$
24	16,777,216	$0100,0000
25	33,554,432	$0200,0000
26	67,108,864	$0400,0000
27	134,217,728	$0800,0000
28	268,435,456	$1000,0000
29	536,870,912	$2000,0000
30	1,073,741,824	$4000,0000
31	2,147,483,648	$8000,0000

n	2^n	$\$2^n$
32	4,294,967,296	$01,0000,0000
33	8,589,934,592	$02,0000,0000
34	17,179,869,184	$04,0000,0000
35	34,359,738,368	$08,0000,0000
36	68,719,476,736	$10,0000,0000
37	137,438,953,472	$20,0000,0000
38	274,877,906,944	$40,0000,0000
39	549,755,813,888	$80,0000,0000

Operation Codes, Numerical Listing

OP	MNEM	MODE	~	#	OP	MNEM	MODE	~	#	OP	MNEM	MODE	~	#	OP	MNEM	MODE	~	#	OP	MNEM	MODE	~	#	OP	MNEM	MODE	~	#
01	NOP	INHER	2	1	30	TSX	INHER	4	1	60	NEG	INDXD	7	2	8C	CPX	IMMED	3	3	B0	SUBA	EXTND	4	3	D8	EORB	DIR	3	2
06	TAP	↑	2	1	31	INS	↑	4	1	63	COM	↑	7	2	8D	BSR	REL	8	2	B1	CMPA	↑	4	3	D9	ADCB	↑	3	2
07	TPA	↕	2	1	32	PULA	↕	4	1	64	LSR	↕	7	2	8E	LDS	IMMED	3	3	B2	SBCA	↕	4	3	DA	ORAB	↕	3	2
08	INX	↕	4	1	33	PULB	↕	4	1	66	ROR	↕	7	2	90	SUBA	DIR	3	2	B4	ANDA	↕	4	3	DB	ADDB	↕	3	2
09	DEX	↕	4	1	34	DES	↕	4	1	67	ASR	↕	7	2	91	CMPA	↑	3	2	B5	BITA	↕	4	3	DE	LDX	↓	4	2
0A	CLV	↕	2	1	35	TXS	↕	4	1	68	ASL	↕	7	2	92	SBCA	↕	3	2	B6	LDAA	↕	4	3	DF	STX	DIR	5	2
0B	SEV	↕	2	1	36	PSHA	↕	4	1	69	ROL	↕	7	2	94	ANDA	↕	3	2	B7	STAA	↕	5	3	E0	SUBB	INDXD	5	2
0C	CLC	↕	2	1	37	PSHB	↕	4	1	6A	DEC	↕	7	2	95	BITA	↕	3	2	B8	EORA	↕	4	3	E1	CMPB	↑	5	2
0D	SEC	↕	2	1	39	RTS	↕	5	1	6C	INC	↕	7	2	96	LDAA	↕	3	2	B9	ADCA	↕	4	3	E2	SBCB	↕	5	2
0E	CLI	↕	2	1	3B	RTI	↕	10	1	6D	TST	↕	7	2	97	STAA	↕	4	2	BA	ORAA	↕	4	3	E4	ANDB	↕	5	2
0F	SEI	↕	2	1	3E	WAI	↕	9	1	6E	JMP	↓	4	2	98	EORA	↕	3	2	BB	ADDA	↕	4	3	E5	BITB	↕	5	2
10	SBA	↕	2	1	3F	SWI	↕	12	1	6F	CLR	INDXD	7	2	99	ADCA	↕	3	2	BC	CPX	↕	5	3	E6	LDAB	↕	5	2
11	CBA	↕	2	1	40	NEGA	↕	2	1	70	NEG	EXTND	6	3	9A	ORAA	↕	3	2	BD	JSR	↕	9	3	E7	STAB	↕	6	2
16	TAB	↕	2	1	43	COMA	↕	2	1	73	COM	↕	6	3	9B	ADDA	↕	3	2	BE	LDS	↓	5	3	E8	EORB	↕	5	2
17	TBA	↕	2	1	44	LSRA	↕	2	1	74	LSR	↕	6	3	9C	CPX	↕	4	2	BF	STS	EXTND	6	3	E9	ADCB	↕	5	2
19	DAA	↓	2	1	46	RORA	↕	2	1	76	ROR	↕	6	3	9E	LDS	↓	4	2	C0	SUBB	IMMED	2	2	EA	ORAB	↕	5	2
1B	ABA	INHER	2	1	47	ASRA	↕	2	1	77	ASR	↕	6	3	9F	STS	DIR	5	2	C1	CMPB	↑	2	2	EB	ADDB	↕	5	2
20	BRA	REL	4	2	48	ASLA	↕	2	1	78	ASL	↕	6	3	A0	SUBA	INDXD	5	2	C2	SBCB	↕	2	2	EE	LDX	↓	6	2
22	BHI	↑	4	2	49	ROLA	↕	2	1	79	ROL	↕	6	3	A1	CMPA	↑	5	2	C4	ANDB	↕	2	2	EF	STX	INDXD	7	2
23	BLS	↕	4	2	4A	DECA	↕	2	1	7A	DEC	↕	6	3	A2	SBCA	↕	5	2	C5	BITB	↕	2	2	F0	SUBB	EXTND	4	3
24	BCC	↕	4	2	4C	INCA	↕	2	1	7C	INC	↕	6	3	A4	ANDA	↕	5	2	C6	LDAB	↕	2	2	F1	CMPB	↑	4	3
25	BCS	↕	4	2	4D	TSTA	↕	2	1	7D	TST	↕	6	3	A5	BITA	↕	5	2	C8	EORB	↕	2	2	F2	SBCB	↕	4	3
26	BNE	↕	4	2	4F	CLRA	↕	2	1	7E	JMP	↓	3	3	A6	LDAA	↕	5	2	C9	ADCB	↕	2	2	F4	ANDB	↕	4	3
27	BEQ	↕	4	2	50	NEGB	↕	2	1	7F	CLR	EXTND	6	3	A7	STAA	↕	6	2	CA	ORAB	↕	2	2	F5	BITB	↕	4	3
28	BVC	↕	4	2	53	COMB	↕	2	1	80	SUBA	IMMED	2	2	A8	EORA	↕	5	2	CB	ADDB	↓	2	2	F6	LDAB	↕	4	3
29	BVS	↕	4	2	54	LSRB	↕	2	1	81	CMPA	↑	2	2	A9	ADCA	↕	5	2	CE	LDX	IMMED	3	3	F7	STAB	↕	5	3
2A	BPL	↕	4	2	56	RORB	↕	2	1	82	SBCA	↕	2	2	AA	ORAA	↕	5	2	D0	SUBB	DIR	3	2	F8	EORB	↕	4	3
2B	BMI	↕	4	2	57	ASRB	↕	2	1	84	ANDA	↕	2	2	AB	ADDA	↕	5	2	D1	CMPB	↑	3	2	F9	ADCB	↕	4	3
2C	BGE	↕	4	2	58	ASLB	↕	2	1	85	BITA	↕	2	2	AC	CPX	↕	6	2	D2	SBCB	↕	3	2	FA	ORAB	↕	4	3
2D	BLT	↕	4	2	59	ROLB	↕	2	1	86	LDAA	↕	2	2	AD	JSR	↕	8	2	D4	ANDB	↕	3	2	FB	ADDB	↕	4	3
2E	BGT	↕	4	2	5A	DECB	↕	2	1	88	EORA	↕	2	2	AE	LDS	↓	6	2	D5	BITB	↕	3	2	FE	LDX	↓	5	3
2F	BLE	REL	4	2	5C	INCB	↕	2	1	89	ADCA	↕	2	2	AF	STS	INDXD	7	2	D6	LDAB	↓	3	2	FF	STX	EXTND	6	3
					5D	TSTB	↓	2	1	8A	ORAA	↓	2	2						D7	STAB	DIR	4	2					
					5F	CLRB	INHER	2	1	8B	ADDA	IMMED	2	2															

ASCII Character Set (7-Bit Code)

M.S. NIBBLE / L.S. NIBBLE	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL (null)	DLE (data-link escape)	SPC (space)	0	@ (at sign)	P	` (grave accent)	p
1 0001	SOH (start of header)	DC1 (XON)	! (exclamation)	1	A	Q	a	q
2 0010	STX (start of text)	DC2 (direct control 2)	" (quotation mark)	2	B	R	b	r
3 0011	EXT (end of text)	DC3 (XOFF)	# (number sign)	3	C	S	c	s
4 0100	EOT (end of transmission)	DC4 (direct control 4)	$ (dollar sign)	4	D	T	d	t
5 0101	ENQ (enquiry)	NAK (negative acknowledge)	% (percent sign)	5	E	U	e	u
6 0110	ACK (acknowledge)	SYN (synchronous idle)	& (ampersand)	6	F	V	f	v
7 0111	BEL (bell)	ETB (end transmission block)	' (apostrophe)	7	G	W	g	w
8 1000	BS (backspace)	CAN (cancel)	((left parenthesis)	8	H	X	h	x
9 1001	HT (horizontal tab)	EM (end of medium)	) (right parenthesis)	9	I	Y	i	y
A 1010	LF (line feed)	SUB (substitute)	* (asterisk)	: (colon)	J	Z	j	z
B 1011	VT (vertical tab)	ESC (escape)	+ (plus sign)	; (semicolon)	K	[(left bracket)	k	{ (left brace)
C 1100	FF (form feed)	FS (file separator)	, (comma)	< (less)	L	\ (backslash)	l	| (pipe)
D 1101	CR (carriage return)	GS (group separator)	– (hyphen)	= (equal)	M	] (right bracket)	m	} (right brace)
E 1110	SO (shift out)	RS (record separator)	. (period)	> (greater)	N	^ (circumflex)	n	~ (tilde)
F 1111	SI (shift in)	US (unit separator)	/ (forward slash)	? (question)	O	_ (underscore)	o	DEL (delete)

MUDBUG Command Summary

Parameters: START, STOP, KEY, and MASK.

Continuation Terminators: Next = <CR> or <ENTER> Same = , (comma)
Previous = ∧ (circumflex) Terminate = . (period)

Cmd	Prm	Description
*	0	An asterisk introduces a comment line.
?	0	Display interactive help. (Same as H command.)
A	0	Display the value of the AR.
A	1	Set AR to START.
B	0	Display the value of the BR.
B	1	Set BR to START.
CA	1	CAlculate expression START; display answer in hex.
CB	1	Change Byte. Display and open byte location START for change, and allow continuation.
CB	2	Change Byte. Set byte location START to STOP.
CC	0	Display the value of the CC.
CC	1	Set CC to START.
CI	1	Change Instruction. Display and open instruction at START for change in assembly language, and allow continuation.
CV	0	Change Vectors. Display and open interrupt vectors for change as a circular list; order is IRQ ↔ SWI ↔ NMI.
CW	1	Change Word. Display and open word location START for change, and allow continuation.
CW	2	Change Word. Set word location START to STOP.
DB	3	Display memory Bytes START through STOP if START ≤ STOP. Display STOP memory bytes beginning at START if START > STOP. Display KEY lines on each screen.
DI	3	Display Instructions. Like DB, but display instructions in assembly language.
DR	0	Display the value of the DR (AR:BR).
DR	1	Set DR (AR:BR) to START.
DV	0	Display interrupt Vectors.
DW	3	Display Word values. Like DB, but group bytes into 16-bit word values.
FB	4	Find Bytes from START through STOP with the value KEY considering only the bits in MASK if MASK ≠ 0. Find bytes not equal to KEY if MASK = 0.
G	1	Go to location START (default = PC) to execute the user's program.
H	0	Display interactive Help. (Same as ? command.)
IB	3	Initialize Bytes from START through STOP to the 8-bit value KEY. Analogous to DB for START > STOP.
IO	0	Switch I/O control between User and System.
IS	0	Initialize System. Reset system, and set registers to default values.
IW	3	Initialize Words from START through STOP to the 16-bit value KEY. Analogous to DW for START > STOP.
LD	1	LoaD an S-record object module into memory. START = offset. For example, "LD 0 DOWNLOAD PROGRAM.OBJ"
LM	0	Go to Line Mode; initialize and erase the screen.
LQ	0	Load Query. Verify that MUDBUG correctly downloaded or compared the last object file.
MB	1	Monitor Byte. Add byte location START to monitor window (visible in screen mode).
MB	2	Monitor Bytes from START through STOP. Like DB for START > STOP.
MD	0	Module Deselect. Deselect user module with local symbols.
MS	1	Module Select. Select user module with local symbols.
MW	1	Monitor Word. Add word location START to the monitor window (visible in screen mode).
MW	2	Monitor Words from START through STOP. Like DW for START > STOP.
N	1	N-step. Execute next START program instructions, beginning at the PC; enter step mode.
O	1	One-step. Execute one instruction at location START (default = PC); enter step mode.
PA	1	Switch Port Assignments. Reverse host and terminal port assignments.
PC	0	Display the value of the PC.
PC	1	Set PC to START.
PE	1	PEek at memory. Display byte at location START; allow continuation.
PO	1	POke memory. Change byte location START (without reading), and allow continuation.
PO	2	POke memory. Set byte START to STOP without reading.
PM	1	Enter Port transparent Mode with exit character START (default = ^X).
Q	0	Query MPU state. Display registers and next instruction.
RA	0	Remove All variables from monitor window (visible in screen mode).
RD	1	Relative Displacement. Display the byte-relative destination address of a branch, and then accept a new destination.
RM	1	Remove Monitor variable START from the monitor window (visible in screen mode).
RM	2	Remove Monitor items from START through STOP.
SB	1	Set host Baud rate to START.
SM	0	Go to Screen Mode; initialize and display all windows.
SP	0	Display the value of the SP.
SP	1	Set SP to START.
T	4	Trap. Trace program flow from START through KEY times the program reaches location STOP, printing the register values every MASK times the program reaches location STOP. T can trap through programs that reside in RAM or Pseudo ROM, but not ROM.
V	4	Verify ROM program. Like T, but works even in ROM.
W	2	Write memory locations START through STOP to the host port as an S-record object file.
X	0	Display the value of the XR.
X	1	Set XR to START.
Z	0	Zero AR, BR, XR, and CC, and initialize PC and SP.