

* MONITOR PROGRAM FOR THE SOUTHWEST TECHNICAL
 * PRODUCTS MP-09 CPU BOARD AS COMMENTED BY....

* ALLEN CLARK WALLACE WATSON
 * 2502 REGAL OAKS LANE 4815 EAST 97th AVE.
 * LUTZ, FLA. 33549 TEMPLE TERRACE, FLA. 33617
 * PH. 813-977-0347 PH. 813-985-1359

* MODIFIED TO SBUG09 VER 1.8 BY: RANDY JARRETT
 * 2561 NANTUCKET DR APT. E
 * ATLANTA, GA 30345
 * PH. 404-320-1043

* *** COMMANDS ***

* CONTROL A = ALTER THE "A" ACCUMULATOR
 * CONTROL B = ALTER THE "B" ACCUMULATOR
 * CONTROL C = ALTER THE CONDITION CODE REGISTER
 * CONTROL D = ALTER THE DIRECT PAGE REGISTER
 * CONTROL P = ALTER THE PROGRAM COUNTER
 * CONTROL U = ALTER USER STACK POINTER
 * CONTROL X = ALTER "X" INDEX REGISTER
 * CONTROL Y = ALTER "Y" INDEX REGISTER
 * B hhhh = SET BREAKPOINT AT LOCATION \$hhhh
 * D = BOOT A SWTPC 8 INCH FLOPPY SYSTEM
 * U = BOOT A SWTPC 5 INCH FLOPPY SYSTEM
 * E ssss-eeee = EXAMINE MEMORY FROM STARTING ADDRESS ssss
 * -TO ENDING ADDRESS eeee.
 * G = CONTINUE EXECUTION FROM BREAKPOINT OR SWI
 * L = LOAD TAPE
 * M hhhh = EXAMINE AND CHANGE MEMORY LOCATION hhhh
 * P ssss-eeee = PUNCH TAPE, START ssss TO END eeee ADDR.
 * Q ssss-eeee = TEST MEMORY FROM ssss TO eeee
 * R = DISPLAY REGISTER CONTENTS
 * S = DISPLAY STACK FROM ssss TO \$DFC0
 * X = REMOVE ALL BREAKPOINTS

55AA TSTPAT EQU \$55AA TEST PATTERN

DFC0		ORG	\$DFC0	
DFC0	STACK	RMB	2	TOP OF INTERNAL STACK / USER VECTOR
DFC2	SWI3	RMB	2	SOFTWARE INTERRUPT VECTOR #3
DFC4	SWI2	RMB	2	SOFTWARE INTERRUPT VECTOR #2
DFC6	FIRQ	RMB	2	FAST INTERRUPT VECTOR
DFC8	IRQ	RMB	2	INTERRUPT VECTOR
DFCA	SWI	RMB	2	SOFTWARE INTERRUPT VECTOR
DFCC	SVCV0	RMB	2	SUPERVISOR CALL VECTOR ORIGIN
DFCE	SVCVL	RMB	2	SUPERVISOR CALL VECTOR LIMIT

DFD0	LRARAM	RMB	16	LRA ADDRESSES
DFE0	CPORT	RMB	2	RE-VECTORABLE CONTROL PORT
DFE2	ECHO	RMB	1	ECHO FLAG
DFE3	BPTBL	RMB	24	BREAKPOINT TABLE BASE ADDR
	E004	ACIAS	EQU	\$E004 CONTROL PORT
	E018	Comreg	EQU	\$E018 COMMAND REGISTER
	E014	Drvreg	EQU	\$E014 DRIVE REGISTER
	E01A	Secreg	EQU	\$E01A SECTOR REGISTER
	E01B	Datreg	EQU	\$E01B DATA REGISTER

F000	ADDREG	EQU	\$F000	ADDRESS REGISTER
F002	CNTREG	EQU	\$F002	COUNT REGISTER
F010	CCREG	EQU	\$F010	CHANNEL CONTROL REGISTER
F014	PRIREG	EQU	\$F014	DMA PRIORITY REGISTER
F015	AAAREG	EQU	\$F015	???
F016	BBBREG	EQU	\$F016	???
F020	COMREG	EQU	\$F020	1791 COMMAND REGISTER
F022	SECREG	EQU	\$F022	SECTOR REGISTER
F024	DRVREG	EQU	\$F024	DRIVE SELECT LATCH
F040	CCCREG	EQU	\$F040	???

FFF0	IC11	EQU	\$FFF0	DAT RAM CHIP
------	------	-----	--------	--------------

F800		ORG	\$F800
F800 F814		FDB	MONITOR
F802 F861		FDB	NEXTCMD
F804 FDCF		FDB	INCH
F806 FDC9		FDB	INCHE
F808 FDDF		FDB	INCHEK
F80A FDEE		FDB	OUTCH
F80C FDBD		FDB	PDATA
F80E FDB1		FDB	PCRLF
F810 FDAD		FDB	PSTRNG
F812 FB81		FDB	LRA

* MONITOR

* VECTOR ADDRESS STRING IS.....

* \$F8A1-\$F8A1-\$F8A1-\$F8A1-\$F8A1-\$FAB0-\$FFFF-\$FFFF

F814 8E	FE4F	MONITOR	LDX	#RAMVEC	POINT TO VECTOR ADDR. STRING
F817 108E	DFC0		LDY	#STACK	POINT TO RAM VECTOR LOCATION
F81B C6	10		LDB	#\$10	BYTES TO MOVE = 16
F81D A6	80	LOOPA	LDA	,X+	GET VECTOR BYTE
F81F A7	A0		STA	,Y+	PUT VECTORS IN RAM / \$DFC0-\$DFCF
F821 5A			DECB		SUBTRACT 1 FROM NUMBER OF BYTES TO MOVE
F822 26	F9		BNE	LOOPA	CONTINUE UNTIL ALL VECTORS MOVED

* CONTENTS	FROM	TO	FUNCTION
* \$F8A1	\$FE40	\$DFC0	USER-V
* \$F8A1	\$FE42	\$DFC2	SWI3-V
* \$F8A1	\$FE44	\$DFC4	SWI2-V
* \$F8A1	\$FE46	\$DFC6	FIRQ-V
* \$F8A1	\$FE48	\$DFC8	IRQ-V

```

* $FAB0      $FE4A      $DFCA      SWI-V
* $FFFF      $FE4C      $DFCC      SVC-V0
* $FFFF      $FE4E      $DFCE      SVC-VL

```

```

F824 8E E004          LDX #ACIAS GET CONTROL PORT ADDR.
F827 BF DFE0          STX CPORT STORE ADDR. IN RAM
F82A 17 027A          LBSR XBKPNT CLEAR OUTSTANDING BREAKPOINTS
F82D C6 0C            LDB #12 CLEAR 12 BYTES ON STACK
F82F 6F E2          CLRSTK CLR , -S
F831 5A              DECB
F832 26 FB            BNE CLRSTK
F834 30 8C DD          LEAX MONITOR,PCR SET PC TO SBUG-E ENTRY
F837 AF 6A            STX 10,S ON STACK
F839 86 D0            LDA #$D0 PRESET CONDITION CODES ON STACK
F83B A7 E4            STA ,S
F83D 1F 43            TFR S,U
F83F 17 05BE          LBSR ACINIZ INITIALIZE CONTROL PORT
F842 8E FE5F          LDX #MSG1 POINT TO 'SBUG 1.8' MESSAGE
F845 17 0575          LBSR PDATA PRINT MSG
F848 8E DFD0          LDX #LRARAM POINT TO LRA RAM STORAGE AREA
F84B 4F              CLRA START TOTAL AT ZERO
F84C C6 0D            LDB #13 TOTAL UP ALL ACTIVE RAM MEMORY
F84E 6D 85          FNDREL TST B,X TEST FOR RAM AT NEXT LOC.
F850 27 03            BEQ RELPAS IF NO RAM GO TO NEXT LOC.
F852 8B 04            ADDA #4 ELSE ADD 4K TO TOTAL
F854 19              DAA ADJ. TOTAL FOR DECIMAL
F855 5A          RELPAS DECB SUB. 1 FROM LOCS. TO TEST
F856 2A F6            BPL FNDREL PRINT TOTAL OF RAM
F858 17 0526          LBSR OUT2H OUTPUT HEX BYTE AS ASCII
F85B 8E FE74          LDX #MSG2 POINT TO MSG 'K' CR/LF + 3 NULS
F85E 17 055C          LBSR PDATA PRINT MSG

```

***** NEXTCMD *****

```

F861 8E FE7B          NEXTCMD LDX #MSG3 POINT TO MSG ">"
F864 17 0546          LBSR PSTRNG PRINT MSG
F867 17 0565          LBSR INCH GET ONE CHAR. FROM TERMINAL
F86A 84 7F            ANDA #$7F STRIP PARITY FROM CHAR.
F86C 81 0D            CMPA #$0D IS IT CARRIAGE RETURN ?
F86E 27 F1            BEQ NEXTCMD IF CR THEN GET ANOTHER CHAR.
F870 1F 89            TFR A,B PUT CHAR. IN "B" ACCUM.
F872 81 20            CMPA #$20 IS IT CONTROL OR DATA CHAR ?
F874 2C 09            BGE PRTCMD IF CMD CHAR IS DATA, PRNT IT
F876 86 5E            LDA #'^ ELSE CNTRL CHAR CMD SO...
F878 17 0573          LBSR OUTCH PRINT "^"
F87B 1F 98            TFR B,A RECALL CNTRL CMD CHAR
F87D 8B 40            ADDA #$40 CONVERT IT TO ASCII LETTER
F87F 17 056C          PRTCMD LBSR OUTCH PRNT CMD CHAR
F882 17 0567          LBSR OUT1S PRNT SPACE
F885 C1 60            CMPB #$60
F887 2F 02            BLE NXTCH0
F889 C0 20            SUBB #$20

```

***** DO TABLE LOOKUP *****
 * FOR COMMAND FUNCTIONS

F88B	8E	FE13	NXTCH0	LDX	#JMPTAB	POINT TO JUMP TABLE
F88E	E1	80	NXTCHR	CMPB	,X+	DOES COMMAND MATCH TABLE ENTRY ?
F890	27	0F		BEQ	JMPCMD	BRANCH IF MATCH FOUND
F892	30	02		LEAX	2,X	POINT TO NEXT ENTRY IN TABLE
F894	8C	FE4F		CMPX	#TABEND	REACHED END OF TABLE YET ?
F897	26	F5		BNE	NXTCHR	IF NOT END, CHECK NEXT ENTRY
F899	8E	FE7D		LDX	#MSG4	POINT TO MSG "WHAT?"
F89C	17	051E		LBSR	PDATA	PRINT MSG
F89F	20	C0		BRA	NEXTCMD	IF NO MATCH, PRMPT FOR NEW CMD
F8A1	AD	94	JMPCMD	JSR	[,X]	JUMP TO COMMAND ROUTINE
F8A3	20	BC		BRA	NEXTCMD	PROMPT FOR NEW COMMAND

*
 * "G" GO OR CONTINUE

F8A5	1F	34	GO	TFR	U,S
F8A7	3B		RTI	RTI	

* "R" DISPLAY REGISTERS

F8A8	8E	FE83	REGSTR	LDX	#MSG5	POINT TO MSG " - "
F8AB	17	04FF		LBSR	PSTRNG	PRINT MSG
F8AE	17	0411		LBSR	PRTSP	\$FCBF
F8B1	17	0419		LBSR	PRTUS	\$FCCA
F8B4	17	0421		LBSR	PRTDP	\$FCD5
F8B7	17	0429		LBSR	PRTIX	\$FCE0
F8BA	17	0431		LBSR	PRTIY	\$FCEB
F8BD	8E	FE83		LDX	#MSG5	POINT TO MSG " - "
F8C0	17	04EA		LBSR	PSTRNG	PRINT MSG
F8C3	17	0433		LBSR	PRTPC	\$FCF5
F8C6	17	043A		LBSR	PRTA	\$FCFF
F8C9	17	0441		LBSR	PRTB	\$FD09
F8CC	16	0448		LBRA	PRTCC	\$FD13

* ALTER "PC" PROGRAM COUNTER

F8CF	17	0427	ALTRPC	LBSR	PRTPC	\$FCF5 PRINT MSG " PC = "
F8D2	17	0517		LBSR	OUT1S	OUTPUT SPACE
F8D5	17	0457		LBSR	IN1ADR	GET NEW CONTENTS FOR "PC"
F8D8	29	02		BVS	ALTPCD	EXIT IF INVALID HEX
F8DA	AF	4A		STX	10,U	POKE IN NEW CONTENTS
F8DC	39		ALTPCD	RTS		

* ALTER "U" USER STACK POINTER

F8DD	17	03ED	ALTRU	LBSR	PRTUS	\$FCCA PRINT MSG " US = "
F8E0	17	0509		LBSR	OUT1S	OUTPUT SPACE

```

F8E3 17 0449      LBSR  IN1ADR
F8E6 29 02        BVS   ALTUD
F8E8 AF 48        STX   8,U
F8EA 39          ALTUD RTS

```

*

* ALTER "Y" INDEX REGISTER

```

F8EB 17 0400      ALTRY  LBSR  PRTIY      PRINT MSG " IY = "
F8EE 17 04FB      LBSR  OUT1S      OUTPUT SPACE
F8F1 17 043B      LBSR  IN1ADR
F8F4 29 02        BVS   ALTUD
F8F6 AF 46        STX   6,U        $F8F0
F8F8 39          ALTUD RTS

```

* ALTER "X" INDEX REGISTER

```

F8F9 17 03E7      ALTRX  LBSR  PRTIX      $FCE0 PRINT MSG " IX = "
F8FC 17 04ED      LBSR  OUT1S      OUTPUT SPACE
F8FF 17 042D      LBSR  IN1ADR
F902 29 02        BVS   ALTXD
F904 AF 44        STX   4,U
F906 39          ALTXD RTS

```

* ALTER "DP" DIRECT PAGE REGISTER

```

F907 17 03CE      ALTRDP LBSR  PRTDP      $FCD5 PRINT MSG " DP = "
F90A 17 04DF      LBSR  OUT1S      OUTPUT SPACE
F90D 17 0430      LBSR  BYTE      INPUT BYTE (2 HEX CHAR)
F910 29 02        BVS   ALTDPD
F912 A7 43        STA   3,U
F914 39          ALTDPD RTS

```

* ALTER "B" ACCUMULATOR

```

F915 17 03F5      ALTRB  LBSR  PRTB      $FD09 PRINT MSG " B = "
F918 17 04D1      LBSR  OUT1S      OUTPUT SPACE
F91B 17 0422      LBSR  BYTE      INPUT BYTE (2 HEX CHAR)
F91E 29 02        BVS   ALTBD
F920 A7 42        STA   2,U
F922 39          ALTBD RTS        $F91C

```

* ALTER "A" ACCUMULATOR

*

```

F923 17 03DD      ALTRA  LBSR  PRTA      $FCFF RINT MSG " A = "

```

F926	17	04C3		LBSR	OUT1S	OUTPUT SPACE
F929	17	0414		LBSR	BYTE	INPUT BYTE (2 HEX CHAR)
F92C	29	02		BVS	ALTAD	
F92E	A7	41		STA	1,U	
F930	39		ALTAD	RTS		

* ALTER "CC" REGISTER

F931	17	03E3	ALTRCC	LBSR	PRTCC	\$FD13 PRINT MSG " CC: "
F934	17	04B5		LBSR	OUT1S	OUTPUT SPACE
F937	17	0406		LBSR	BYTE	INPUT BYTE (2 HEX CHAR)
F93A	29	04		BVS	ALTCCD	
F93C	8A	80		ORA	#\$80	SETS "E" FLAG IN PRINT LIST
F93E	A7	C4		STA	,U	
F940	39		ALTCCD	RTS		

***** "M" MEMORY EXAMINE AND CHANGE *****

F941	17	03EB	MEMCHG	LBSR	IN1ADR	INPUT ADDRESS
F944	29	2D		BVS	CHRTN	IF NOT HEX, RETURN
F946	1F	12		TFR	X,Y	SAVE ADDR IN "Y"
F948	8E	FE83	MEMC2	LDX	MSG5	POINT TO MSG " - "
F94B	17	045F		LBSR	PSTRNG	PRINT MSG
F94E	1F	21		TFR	Y,X	FETCH ADDRESS
F950	17	0426		LBSR	OUT4H	PRINT ADDR IN HEX
F953	17	0496		LBSR	OUT1S	OUTPUT SPACE
F956	A6	A4		LDA	,Y	GET CONTENTS OF CURRENT ADDR.
F958	17	0426		LBSR	OUT2H	OUTPUT CONTENTS IN ASCII
F95B	17	048E		LBSR	OUT1S	OUTPUT SPACE
F95E	17	03DF		LBSR	BYTE	LOOP WAITING FOR OPERATOR INPUT
F961	28	11		BVC	CHANGE	IF VALID HEX GO CHANGE MEM. LOC.
F963	81	08		CMPA	#8	IS IT A BACKSPACE (CNTRL H)?
F965	27	E1		BEQ	MEMC2	PROMPT OPERATOR AGAIN
F967	81	18		CMPA	#\$18	IS IT A CANCEL (CNTRL X)?
F969	27	DD		BEQ	MEMC2	PROMPT OPERATOR AGAIN
F96B	81	5E		CMPA	#'^	IS IT AN UP ARROW?
F96D	27	17		BEQ	BACK	DISPLAY PREVIOUS BYTE
F96F	81	0D		CMPA	#\$D	IS IT A CR?
F971	26	0F		BNE	FORWRD	DISPLAY NEXT BYTE
F973	39		CHRTN	RTS		EXIT ROUTINE

F974	A7	A4	CHANGE	STA	,Y	CHANGE BYTE IN MEMORY
F976	A1	A4		CMPA	,Y	DID MEMORY BYTE CHANGE?
F978	27	08		BEQ	FORWRD	\$F972
F97A	17	046F		LBSR	OUT1S	OUTPUT SPACE
F97D	86	3F		LDA	#'?	LOAD QUESTION MARK
F97F	17	046C		LBSR	OUTCH	PRINT IT
F982	31	21	FORWRD	LEAY	1,Y	POINT TO NEXT HIGHER MEM LOCATION
F984	20	C2		BRA	MEMC2	PRINT LOCATION & CONTENTS
F986	31	3F	BACK	LEAY	-1,Y	POINT TO LAST MEM LOCATION
F988	20	BE		BRA	MEMC2	PRINT LOCATION & CONTENTS

* "S" DISPLAY STACK
 * HEX-ASCII DISPLAY OF CURRENT STACK CONTENTS FROM
 * CURRENT STACK POINTER TO INTERNAL STACK LIMIT.

F98A	17	0335	DISSTK	LBSR	PRTSP	PRINT CURRENT STACK POINTER
F98D	1F	32		TFR	U,Y	
F98F	8E	DFC0		LDX	#STACK	LOAD INTERNAL STACK AS UPPER LIMIT
F992	30	1F		LEAX	-1,X	POINT TO CURRENT STACK
F994	20	05		BRA	MDUMP1	ENTER MEMORY DUMP OF STACK CONTENTS

* "E" DUMP MEMORY FOR EXAMINE IN HEX AND ASCII
 * AFTER CALLING 'IN2ADR' LOWER ADDRESS IN Y-REG.
 * UPPER ADDRESS IN X-REG.
 * IF HEX ADDRESSES ARE INVALID (V)=1.

F996	17	038B	MEMDUMP	LBSR	IN2ADR	INPUT ADDRESS BOUNDRIES
F999	29	06		BVS	EDPRTN	NEW COMMAND IF ILLEGAL HEX
F99B	34	20	MDUMP1	PSHS	Y	COMPARE LOWER TO UPPER BOUNDS
F99D	AC	E1		CMPX	,S++	LOWER BOUNDS > UPPER BOUNDS?
F99F	24	01		BCC	AJDUMP	IF NOT, DUMP HEX AND ASCII
F9A1	39		EDPRTN	RTS		

* ADJUST LOWER AND UPPER ADDRESS LIMITS
 * TO EVEN 16 BYTE BOUNDRIES.

* IF LOWER ADDR = \$4532
 * LOWER BOUNDS WILL BE ADJUSTED TO = \$4530.

* IF UPPER ADDR = \$4567
 * UPPER BOUNDS WILL BE ADJUSTED TO = \$4570.

* ENTER WITH LOWER ADDRESS IN X-REG.
 * -UPPER ADDRESS ON TOP OF STACK.

F9A2	1F	10	AJDUMP	TFR	X,D	GET UPPER ADDR IN D-REG
F9A4	C3	0010		ADDD	#\$10	ADD 16 TO UPPER ADDRESS
F9A7	C4	F0		ANDB	#\$F0	MASK TO EVEN 16 BYTE BOUNDRY
F9A9	34	06		PSHS	A,B	SAVE ON STACK AS UPPER DUMP LIMIT
F9AB	1F	20		TFR	Y,D	\$F9A5 GET LOWER ADDRESS IN D-REG
F9AD	C4	F0		ANDB	#\$F0	MASK TO EVEN 16 BYTE BOUNDRY
F9AF	1F	01		TFR	D,X	PUT IN X-REG AS LOWER DUMP LIMIT
F9B1	AC	E4	NXTLIN	CMPX	,S	COMPARE LOWER TO UPPER LIMIT
F9B3	27	05		BEQ	SKPDMP	IF EQUAL SKIP HEX-ASCII DUMP
F9B5	17	0427		LBSR	INCHEK	CHECK FOR INPUT FROM KEYBOARD
F9B8	27	03		BEQ	EDUMP	IF NONE, CONTINUE WITH DUMP
F9BA	32	62	SKPDMP	LEAS	2,S	READJUST STACK IF NOT DUMPING
F9BC	39			RTS		

* PRINT 16 HEX BYTES FOLLOWED BY 16 ASCII CHARACTERS
 * FOR EACH LINE THROUGHOUT ADDRESS LIMITS.

F9BD	34	10	EDUMP	PSHS	X	PUSH LOWER ADDR LIMIT ON STACK
F9BF	8E	FE83		LDX	#MSG5	POINT TO MSG " - "

```

F9C2 17 03E8      LBSR  PSTRNG  PRINT MSG
F9C5 AE E4        LDZ    ,S      LOAD LOWER ADDR FROM TOP OF STACK
F9C7 17 03AF      LBSR  OUT4H    PRINT THE ADDRESS LBSR OUT2S PRINT 2
SPACES
F9CA C6 10        LDB    #$10    LOAD COUNT OF 16 BYTES TO DUMP
F9CC A6 80        EL0OP LDA    ,X+  GET FROM MEMORY HEX BYTE TO PRINT
F9CE 17 03B0      LBSR  OUT2H    OUTPUT HEX BYTE AS ASCII
F9D1 17 0418      LBSR  OUT1S    OUTPUT SPACE
F9D4 5A          DECB          $F9D1 DECREMENT BYTE COUNT
F9D5 26 F5        BNE    EL0OP    CONTINUE TIL 16 HEX BYTES PRINTED

```

```

* PRINT 16 ASCII CHARACTERS
* IF NOT PRINTABLE OR NOT VALID
* ASCII PRINT A PERIOD (.)

```

```

F9D7 17 0410      LBSR  OUT2S    2 SPACES
F9DA AE E1        LDZ    ,S++    GET LOW LIMIT FRM STACK - ADJ STACK
F9DC C6 10        LDB    #$10    SET ASCII CHAR TO PRINT = 16
F9DE A6 80        EDPASC LDA    ,X+  GET CHARACTER FROM MEMORY
F9E0 81 20        CMPA   #$20    IF LESS THAN $20, NON-PRINTABLE?
F9E2 25 04        BCS    PERIOD  IF S0, PRINT PERIOD INSTEAD
F9E4 81 7E        CMPA   #$7E    IS IT VALID ASCII?
F9E6 23 02        BLS    PRASC    IF S0 PRINT IT
F9E8 86 2E        PERIOD LDA    #'  LOAD A PERIOD (.)
F9EA 17 0401      PRASC  LBSR  OUTCH PRINT ASCII CHARACTER
F9ED 5A          DECB          DECREMENT COUNT
F9EE 26 EE        BNE    EDPASC
F9F0 20 BF        BRA    NXTLIN

```

***** "Q" MEMORY TEST *****

```

F9F2 6F E2        MEMTST CLR    , -S  CLEAR BYTE ON STACK
F9F4 6F E2        CLR    , -S  CLEAR ANOTHER BYTE
F9F6 17 032B      LBSR  IN2ADR  GET BEGIN(Y) & END(X) ADDR. LIMITS
F9F9 34 30        PSHS   X,Y    SAVE ADDRESSES ON STACK
F9FB 29 7B        BVS    ADJSK6  EXIT IF NOT VALID HEX
F9FD AC 62        CMPX   2,S    COMPARE BEGIN TO END ADDR.
F9FF 25 77        BCS    ADJSK6  EXIT IF BEGIN > END ADDR.
FA01 17 03E8      LBSR  OUT1S    OUTPUT SPACE
FA04 1F 20        MEMSET TFR    Y,D  PUT BEGIN ADDR. IN 'D'-ACCUM.
FA06 E3 64        ADDD   4,S    ADD PASS COUNT TO BEGIN ADDR
FA08 34 04        PSHS   B      ADD LS BYTE TO MS BYTE OF BEGIN ADDR
FA0A AB E0        ADDA   ,S+
FA0C A7 A0        STA    ,Y+    SAVE THIS DATA BYTE AT BEGIN ADDR
FA0E 10AC E4      CMPY   ,S      COMPARE END TO BEGIN ADDR
FA11 25 F1        BCS    MEMSET  IF BEGIN LOWER, CONTINUE TO SET MEMORY
FA13 10AE 62      LDY    2,S    RELOAD BEGIN ADDRESS
FA16 1F 20        TEST1 TFR    Y,D  PUT BEGIN ADDR IN 'D'-ACC.
FA18 E3 64        ADDD   4,S    ADD PASS COUNT TO ADDRESS
FA1A 34 02        PSHS   A      ADD MS BYTE TO LS BYTE OF ADDRESS
FA1C EB E0        ADDB   ,S+
FA1E E8 A0        EORB   ,Y+    EX-OR THIS DATA WITH DATA IN MEMORY LOC.
FA20 27 3C        BEQ    GUDPAS  IF (Z) SET, MEMORY BYTE OK
FA22 8E FE83      LDZ    #MSG5   POINT TO MSG " - "
FA25 17 0385      LBSR  PSTRNG  PRINT MSG
FA28 30 3F        LEAX   -1,Y    GET ERROR ADDRESS IN X-REG

```

FA2A 17	034C		LBSR	OUT4H	OUTPUT IT
FA2D 34	10		PSHS	X	PUSH ERROR ADDR ON STACK
FA2F 8E	FEA1		LDX	#MSG8	POINT TO MSG " =>"
FA32 17	0388		LBSR	PDATA	PRINT MSG
FA35 35	10		PULS	X	POP ERROR ADDR FROM STACK
FA37 17	0147		LBSR	LRA	GET PHYSICAL ADDR FROM LRA
FA3A 17	0350		LBSR	XASCII	OUTPUT EXTENDED 4 BITS OF PHYSICAL ADDR
FA3D 17	0339		LBSR	OUT4H	OUTPUT LS 16 BITS OF PHYSICAL ADDR
FA40 8E	FE87		LDX	#MSG6	POINT TO MSG ", PASS "
FA43 17	0377		LBSR	PDATA	PRINT MSG
FA46 AE	64		LDX	4,S	LOAD PASS COUNT
FA48 17	032E		LBSR	OUT4H	OUTPUT IT
FA4B 8E	FE8F		LDX	#MSG7	POINT TO MSG ", BITS IN ERROR
FA4E 17	036C		LBSR	PDATA	PRINT MSG
FA51 1F	98		TFR	B,A	GET ERROR BYTE INTO A-ACC
FA53 8E	FEA6		LDX	#MSG9	POINT TO MSG "76543210"
FA56 17	033E		LBSR	BIASCI	OUTPUT IN BINARY/ASCII FORMAT
FA59 17	0383		LBSR	INCHEK	CHECK FOR INPUT FROM KEYBOARD \$FA56
FA5C 26	1A		BNE	ADJSK6	IF S0, EXIT MEMORY TEST
FA5E 10AC	E4	GUDPAS	CMPY	,S	COMPARE END ADDR TO BEGIN ADDR
FA61 25	B3		BCS	TEST1	
FA63 86	2B		LDA	#'+	GET "PASS" SYMBOL IF MEMORY PASS OK
FA65 17	0386		LBSR	OUTCH	OUTPUT SYMBOL TO TERMINAL
FA68 17	0374		LBSR	INCHEK	INPUT FROM KEYBOARD?
FA6B 26	0B		BNE	ADJSK6	IF S0, EXIT MEMORY TEST
FA6D 10AE	62		LDY	2,S	LOAD BEGIN ADDRESS
FA70 6C	65		INC	5,S	INCREMENT LS BYTE OF PASS COUNT
FA72 26	90		BNE	MEMSET	IF NOT ZERO, SET NEXT MEMORY BYTE
FA74 6C	64		INC	4,S	INCREMENT MS BYTE OF PASS COUNT
FA76 26	8C		BNE	MEMSET	DONE WITH 65,535 PASSES OF MEMORY?
FA78 32	66	ADJSK6	LEAS	6,S	ADJ STACK POINTER BY 6
FA7A 39			RTS		

***** "B" SET BREAKPOINT *****

FA7B 17	02B1	BRKPNT	LBSR	IN1ADR	GET BREAKPOINT ADDRESS
FA7E 29	1E		BVS	EXITBP	EXIT IF INVALID HEX ADDR.
FA80 8C	DFC0		CMPX	#STACK	ADDRESS ILLEGAL IF >=\$DFC0
FA83 24	1A		BCC	BPERR	IF ERROR PRINT (?), EXIT
FA85 34	10		PSHS	X	\$FA82 PUSH BP ADDRESS ON STACK
FA87 8E	FFFF		LDX	#\$FFFF	LOAD DUMMY ADDR TO TEST BP TABLE
FA8A 8D	55		BSR	BPTST	TEST BP TABLE FOR FREE SPACE
FA8C 35	10		PULS	X	POP BP ADDRESS FROM STACK
FA8E 27	0F		BEQ	BPERR	(Z) SET, OUT OF BP TABLE SPACE
FA90 A6	84		LDA	,X	GET DATA AT BREAKPOINT ADDRESS
FA92 81	3F		CMPA	#\$3F	IS IT A SWI?
FA94 27	09		BEQ	BPERR	IF SWI ALREADY, INDICATE ERROR
FA96 A7	A0		STA	,Y+	SAVE DATA BYTE IN BP TABLE
FA98 AF	A4		STX	,Y	SAVE BP ADDRESS IN BP TABLE
FA9A 86	3F		LDA	#\$3F	LOAD A SWI (\$3F)
FA9C A7	84		STA	,X	SAVE SWI AT BREAKPOINT ADDRESS
FA9E 39		EXITBP	RTS		

* INDICATE ERROR SETTING BREAKPOINT

FA9F 17	034A	BPERR	LBSR	OUT1S	OUTPUT SPACE
FAA2 86	3F		LDA	#'?	LOAD (?), INDICATE BREAKPOINT ERROR
FAA4 16	0347		LBRA	OUTCH	PRINT "?"

*** "X" CLEAR OUTSTANDING BREAKPOINTS ***

FAA7 108E	DfE3	XBKPNt	LDY	#BPTBL	POINT TO BREAKPOINT TABLE
FAAB C6	08		LDB	#8	LOAD BREAKPOINT COUNTER
FAAD 8D	18	XBPLP	BSR	RPLSWI	REMOVE USED ENTRY IN BP TABLE
FAAF 5A			DECB		\$FAAC DECREMENT BP COUNTER
FAB0 26	FB		BNE	XBPLP	END OF BREAKPOINT TABLE?
FAB2 39			RTS		

***** SWI ENTRY POINT *****

FAB3 1F	43	SWIE	TFR	S,U	TRANSFER STACK TO USER POINTER
FAB5 AE	4A		LDX	10,U	LOAD PC FROM STACK INTO X-REG
FAB7 30	1F		LEAX	-1,X	ADJUST ADDR DOWN 1 BYTE.
FAB9 8D	26		BSR	BPTESr	FIND BREAKPOINT IN BP TABLE
FABB 27	04		BEQ	REGPR	IF FOUND, REPLACE DATA AT BP ADDR
FABD AF	4A		STX	10,U	SAVE BREAKPOINT ADDR IN STACK
FABF 8D	06		BSR	RPLSWI	GO REPLACE SWI WITH ORIGINAL DATA
FAC1 17	FDE4	REGPR	LBSR	REGSTR	GO PRINT REGISTERS
FAC4 16	FD9A		LBRA	NEXTCMD	GET NEXT COMMAND
FAC7 AE	21	RPLSWI	LDX	1,Y	LOAD BP ADDRESS FROM BP TABLE
FAC9 8C	DFC0		CMPX	#STACK	COMPARE TO TOP AVAILABLE USER MEMORY
FACC 24	0A		BCC	FFSTBL	GO RESET TABLE ENTRY TO \$FF'S
FACE A6	84		LDA	,X	GET DATA FROM BP ADDRESS
FAD0 81	3F		CMPA	#\$3F	IS IT SWI?
FAD2 26	04		BNE	FFSTBL	IF NOT, RESET TABLE ENTRY TO \$FF'S
FAD4 A6	A4		LDA	,Y	GET ORIGINAL DATA FROM BP TABLE
FAD6 A7	84		STA	,X	\$FAD3 RESTORE DATA AT BP ADDRESS
FAD8 86	FF	FFSTBL	LDA	#\$FF	LOAD \$FF IN A-ACC
FADA A7	A0		STA	,Y+	RESET BREAKPOINT TABLE DATA TO \$FF'S
FADC A7	A0		STA	,Y+	RESET BREAKPOINT TABLE ADDR TO \$FF'S
FADE A7	A0		STA	,Y+	
FAE0 39			RTS		

** SEARCH BREAKPOINT TABLE FOR MATCH **

FAE1 108E	DfE3	BPTESr	LDY	#BPTBL	POINT TO BREAKPOINT TABLE
FAE5 C6	08		LDB	#8	LOAD BREAKPOINT COUNTER
FAE7 A6	A0	FNDBP	LDA	,Y+	LOAD DATA BYTE
FAE9 AC	A1		CMPX	,Y++	COMPARE ADDRESS, IS IT SAME?
FAEB 27	04		BEQ	BPADJ	IF SO, ADJUST POINTER FOR TABLE ENTRY
FAED 5A			DECB		IF NOT, DECREMENT BREAKPOINT COUNTER
FAEE 26	F7		BNE	FNDBP	AND LOOK FOR NEXT POSSIBLE MATCH
FAF0 39			RTS		

FAF1 31	3D	BPADJ	LEAY	-3,Y	MOVE POINTER TO BEGIN OF BP ENTRY
FAF3 39			RTS		

*** "D" DISK BOOT FOR DMAF2 ***

FAF4	86	DE	DBOOT	LDA	#\$DE	
FAF6	B7	F024		STA	DRVREG	
FAF9	86	FF		LDA	#\$FF	
FAFB	B7	F014		STA	PRIREG	\$FAF8
FAFE	B7	F010		STA	CCREG	
FB01	B7	F015		STA	AAAREG	
FB04	B7	F016		STA	BBBREG	
FB07	7D	F010		TST	CCREG	
FB0A	86	D8		LDA	#\$D8	
FB0C	B7	F020		STA	COMREG	
FB0F	17	0097		LBSR	DLY	
FB12	B6	F020	DBOOT0	LDA	COMREG	
FB15	2B	FB		BMI	DBOOT0	
FB17	86	09		LDA	#\$09	
FB19	B7	F020		STA	COMREG	
FB1C	17	008A		LBSR	DLY	
FB1F	B6	F020	DISKWT	LDA	COMREG	FETCH DRIVE STATUS
FB22	85	01		BITA	#1	TEST BUSY BIT
FB24	26	F9		BNE	DISKWT	LOOP UNTIL NOT BUSY
FB26	85	10		BITA	#\$10	
FB28	26	CA		BNE	DBOOT	
FB2A	8E	C000		LDX	#\$C000	LOGICAL ADDR. = \$C000
FB2D	8D	52		BSR	LRA	GET 20 BIT PHYSICAL ADDR. OF LOG. ADDR.
FB2F	8A	10		ORA	#\$10	
FB31	B7	F040		STA	CCCREG	
FB34	1F	10		TFR	X, D	
FB36	43			COMA		
FB37	53			COMB		
FB38	FD	F000		STD	ADDREG	
FB3B	8E	FEFF		LDX	#\$FEFF	LOAD DMA BYTE COUNT = \$100
FB3E	BF	F002		STX	CNTREG	STORE IN COUNT REGISTER
FB41	86	FF		LDA	#\$FF	LOAD THE CHANNEL REGISTER
FB43	B7	F010		STA	CCREG	
FB46	86	FE		LDA	#\$FE	SET CHANNEL 0
FB48	B7	F014		STA	PRIREG	
FB4B	86	01		LDA	#1	SET SECTOR TO "1"
FB4D	B7	F022		STA	SECREG	ISSUE COMMAND
FB50	86	8C		LDA	#\$8C	SET SINGLE SECTOR READ
FB52	B7	F020		STA	COMREG	ISSUE COMMAND
FB55	8D	52		BSR	DLY	

* THE FOLLOWING CODE TESTS THE STATUS OF THE
 * CHANNEL CONTROL REGISTER. IF "D7" IS NOT
 * ZERO THEN IT WILL LOOP WAITING FOR "D7"
 * TO GO TO ZERO. IF AFTER 65,536 TRIES IT
 * IS STILL A ONE THE BOOT OPERATION WILL
 * BE STARTED OVER FROM THE BEGINING.

FB57 5F

CLRB

FB58	34	04	DB00T1	PSHS	B	\$FB55
FB5A	5F			CLRB		
FB5B	7D	F010	DB00T2	TST	CCREG	
FB5E	2A	0A		BPL	DB00T3	
FB60	5A			DECB		
FB61	26	F8		BNE	DB00T2	
FB63	35	04		PULS	B	
FB65	5A			DECB		
FB66	26	F0		BNE	DB00T1	
FB68	20	8A		BRA	DB00T	
FB6A	35	04	DB00T3	PULS	B	
FB6C	B6	F020		LDA	COMREG	
FB6F	85	1C		BITA	#\$1C	
FB71	27	01		BEQ	DB00T4	
FB73	39			RTS		

FB74	C6	DE	DB00T4	LDB	#\$DE	
FB76	F7	F024		STB	DRVREG	
FB79	8E	C000		LDX	#\$C000	
FB7C	AF	4A		STX	10,U	
FB7E	1F	34		TFR	U,S	\$FB7B
FB80	3B			RTI		

***** LRA LOAD REAL ADDRESS *****

* THE FOLLOWING CODE LOADS THE 20-BIT
 * PHYSICAL ADDRESS OF A MEMORY BYTE
 * INTO THE "A" AND "X" REGISTERS. THIS
 * ROUTINE IS ENTERED WITH THE LOGICAL
 * ADDRESS OF A MEMORY BYTE IN THE "IX"
 * REGISTER. EXIT IS MADE WITH THE HIGH-
 * ORDER FOUR BITS OF THE 20-BIT PHYSICAL
 * ADDRESS IN THE "A" REGISTER, AND THE
 * LOW-ORDER 16-BITS OF THE 20-BIT
 * PHYSICAL ADDRESS IN THE "IX" REGISTER.
 * ALL OTHER REGISTERS ARE PRESERVED.
 * THIS ROUTINE IS REQUIRED SINCE THE
 * DMAF1 AND DMAF2 DISK CONTROLLERS MUST
 * PRESENT PHYSICAL ADDRESSES ON THE
 * SYSTEM BUS.

FB81	34	36	LRA	PSHS	A,B,X,Y	PUSH REGISTERS ON STACK
FB83	A6	62		LDA	2,S	GET MSB LOGICAL ADDR FRM X REG ON STACK
FB85	44			LSRA		
FB86	44			LSRA		ADJ FOR INDEXED INTO
FB87	44			LSRA		CORRESPONDING LOCATION
FB88	44			LSRA		IN LRA TABLE
FB89	108E	DFD0		LDY	#LRARAM	LOAD LRA TABLE BASE ADDRESS
FB8D	E6	A6		LDB	A,Y	GET PHYSICAL ADDR. DATA FROM LRA TABLE
FB8F	54			LSRB		ADJ. REAL ADDR. TO REFLECT EXTENDED
FB90	54			LSRB		PHYSICAL ADDRESS.
FB91	54			LSRB		EXTENDED MS 4-BITS ARE RETURNED
FB92	54			LSRB		IN THE "A" ACCUMULATOR

FB93	E7	E4	STB	,S	MS 4 BITS IN A ACCUM. STORED ON STACK
FB95	E6	A6	LDB	A,Y	LOAD REAL ADDRESS DATA FROM LRA TABLE
FB97	53		COMB		COMP TO ADJ FOR PHYSICAL ADDR. IN X REG
FB98	58		ASLB		ADJ DATA FOR RELOCATION IN X REG
FB99	58		ASLB		
FB9A	58		ASLB		\$FB97
FB9B	58		ASLB		
FB9C	A6	62	LDA	2,S	GET MS BYTE OF LOGICAL ADDR.
FB9E	84	0F	ANDA	#\$0F	MASK MS NIBBLE OF LOGICAL ADDRESS
FBA0	A7	62	STA	2,S	SAVE IT IN X REG ON STACK
FBA2	EA	62	ORB	2,S	SET MS BYTE IN X REG TO ADJ PHY ADDR.

* PLUS LS NIBBLE OF LOGICAL ADDRESS

FBA4	E7	62	STB	2,S	SAVE AS LS 16 BITS OF PHY ADDR IN X REG
* ON STACK					
FBA6	35	36	PULS	A,B,X,Y	POP REGS. FROM STACK
FBA8	39		RTS		

* DELAY LOOP

FBA9	34	04	DLY	PSHS	B	SAVE CONTENTS OF "B"
FBAB	C6	20		LDB	#\$20	GET LOOP DELAY VALUE
FBAD	5A		SUB1	DECB		SUBTRACT ONE FROM VALUE
FBAE	26	FD		BNE	SUB1	LOOP UNTIL ZERO
FBF0	35	04		PULS	B	RESTORE CONTENTS OF "B"
FBF2	39			RTS		

***** "U" MINIDISK BOOT *****

FBF3	7D	E018	MINBOOT	TST	Comreg	
FBF6	7F	E014		CLR	Drvreg	SELECT DRIVE 0

* DELAY BEFORE ISSUING RESTORE COMMAND

FBF9	C6	03		LDB	#3	
FBFB	8E	0000		LDX	#0	
FBFE	30	01	LOOP	LEAX	1,X	\$FBFB
FBC0	8C	0000		CMPX	#0	
FBC3	26	F9		BNE	LOOP	
FBC5	5A			DECB		\$FBC2
FBC6	26	F6		BNE	LOOP	

FBC8	86	0F		LDA	#\$0F	*LOAD HEAD, VERIFY, 20msec/step
FBCA	B7	E018		STA	Comreg	ISSUE RESTORE COMMAND
FBCD	8D	37		BSR	DELAY	
FBCF	F6	E018	LOOP1	LDB	Comreg	\$FBCC
FBD2	C5	01		BITB	#1	
FBD4	26	F9		BNE	LOOP1	LOOP UNTIL THRU
FBD6	86	01		LDA	#1	
FBD8	B7	E01A		STA	Secreg	SET SECTOR REGISTER TO ONE
FBD8	8D	29		BSR	DELAY	
FBD8	86	8C		LDA	#\$8C	LOAD HEAD, DELAY 10msec,
FBD8	B7	E018		STA	Comreg	AND READ SINGLE RECORD
FBE2	8D	22		BSR	DELAY	
FBE4	8E	C000		LDX	#\$C000	

```

FBE7 20 09          BRA    LOOP3

FBE9 C5 02          LOOP2  BITB  #2          $FBE6 DRQ?
FBEB 27 05          BEQ    LOOP3
FBED B6 E01B        LDA    Datreg
FBF0 A7 80          STA    ,X+

FBF2 F6 E018        LOOP3  LDB    Comreg      FETCH STATUS
FBF5 C5 01          BITB  #1          BUSY?
FBF7 26 F0          BNE    LOOP2
FBF9 C5 2C          BITB  #$2C        CRC ERROR OR LOST DATA?
FBFB 27 01          BEQ    LOOP4
FBFD 39            RTS
FBFE 8E C000        LOOP4  LDX    #$C000      $FBFB
FC01 AF 4A          STX    10,U
FC03 1F 34          TFR    U,S
FC05 3B            RTI

```

* DELAY

```

FC06 C6 20          DELAY  LDB    #$20
FC08 5A            LOOP5  DECB
FC09 26 FD          BNE    LOOP5
FC0B 39            RTS

```

***** "L" LOAD MIKBUG TAPE *****

```

FC0C 86 11          LOAD   LDA    #$11        LOAD 'DC1' CASS. READ ON CODE
FC0E 17 01DD        LBSR   OUTCH      OUTPUT IT TO TERMINAL PORT
FC11 7F DFE2        CLR    ECHO       TURN OFF ECHO FLAG
FC14 17 01AD        LOAD1  LBSR   ECHON     INPUT 8 BIT BYTE WITH NO ECHO
FC17 81 53          LOAD2  CMPA   #'S       IS IT AN "S", START CHARACTER ?
FC19 26 F9          BNE    LOAD1      IF NOT, DISCARD AND GET NEXT CHAR.
FC1B 17 01A6        LBSR   ECHON
FC1E 81 39          CMPA   #'9        IS IT A "9" , END OF FILE CHAR ?
FC20 27 3D          BEQ    LOAD21     IF SO, EXIT LOAD
FC22 81 31          CMPA   #'1        IS IT A "1" , FILE LOAD CHAR ?
FC24 26 F1          BNE    LOAD2      IF NOT, LOOK FOR START CHAR.
FC26 17 0117        LBSR   BYTE      INPUT BYTE COUNT
FC29 34 02          PSHS   A          PUSH COUNT ON STACK
FC2B 29 26          BVS    LODERR     (V) C-CODE SET, ILLEGAL HEX
FC2D 17 00FF        LBSR   IN1ADR     INPUT LOAD ADDRESS
FC30 29 21          BVS    LODERR     (V) C-CODE SET, ADDR NOT HEX
FC32 34 10          PSHS   X          PUSH ADDR ON STACK
FC34 E6 E0          LDB    ,S+        LOAD MSB OF ADDR AS CHECKSUM BYTE
FC36 EB E0          ADDB   ,S+        ADD LSB OF ADDR TO CHECKSUM
FC38 EB E4          ADDB   ,S        ADD BYTE COUNT BYTE TO CHECKSUM
FC3A 6A E4          DEC    ,S        $FC37 DECREMENT BYTE COUNT 2 TO BYPASS
FC3C 6A E4          DEC    ,S        ADDRESS BYTES.
FC3E 34 04          LOAD10 PSHS   B      PUSH CHECKSUM ON STACK
FC40 17 00FD        LBSR   BYTE      INPUT DATA BYTE (2 HEX CHAR)
FC43 35 04          PULS   B          POP CHECKSUM FROM STACK
FC45 29 0C          BVS    LODERR     (V) SET, DATA BYTE NOT HEX
FC47 34 02          PSHS   A          PUSH DATA BYTE ON STACK

```

FC49	EB	E0		ADDB	,S+	ADD DATA TO CHECKSUM, AUTO INC STACK
FC4B	6A	E4		DEC	,S	DECREMENT BYTE COUNT 1
FC4D	27	05		BEQ	LOAD16	IF BYTE COUNT ZERO, TEST CHECKSUM
FC4F	A7	80		STA	,X+	SAVE DATA BYTE IN MEMORY
FC51	20	EB		BRA	LOAD10	GET NEXT DATA BYTE
FC53	5F		LODERR	CLRB		ERROR CONDITION, ZERO CHECKSUM
FC54	35	02	LOAD16	PULS	A	ADJUST STACK (REMOVE BYTE COUNT)
FC56	C1	FF		CMPB	#\$FF	CHECKSUM OK?
FC58	27	B2		BEQ	LOAD	IF SO, LOAD NEXT LINE
FC5A	86	3F		LDA	#'?	LOAD (?) ERROR INDICATOR
FC5C	17	018F		LBSR	OUTCH	OUTPUT IT TO TERMINAL
FC5F	73	DFE2	LOAD21	COM	ECHO	TURN ECHO ON
FC62	86	13		LDA	#\$13	\$FC5F LOAD 'DC3' CASS. READ OFF CODE
FC64	16	0187		LBRA	OUTCH	OUTPUT IT

***** "P" PUNCH MIKBUG TAPE *****

FC67	6F	E2	PUNCH	CLR	, -S	CLEAR RESERVED BYTE ON STACK
FC69	17	00B8		LBSR	IN2ADR	GET BEGIN AND END ADDRESS
FC6C	34	30		PSHS	X,Y	SAVE ADDRESSES ON STACK
FC6E	29	4A		BVS	PUNEXT	(V) C-CODE SET, EXIT PUNCH
FC70	AC	62		CMPX	2,S	COMPARE BEGIN TO END ADDR
FC72	25	46		BCS	PUNEXT	IF BEGIN GREATER THAN END, EXIT PUNCH
FC74	30	01		LEAX	1,X	INCREMENT END ADDRESS
FC76	AF	E4		STX	,S	STORE END ADDR ON STACK
FC78	86	12		LDA	#\$12	LOAD 'DC2' PUNCH ON CODE
FC7A	17	0171		LBSR	OUTCH	OUTPUT IT TO TERMINAL
FC7D	EC	E4	PUNCH2	LDD	,S	LOAD END ADDR IN D-ACC
FC7F	A3	62		SUBD	2,S	SUBTRACT BEGIN FROM END
FC81	27	06		BEQ	PUNCH3	SAME, PUNCH 32 BYTES DEFAULT
FC83	1083	0020		CMPD	#\$20	LESS THAN 32 BYTES?
FC87	23	02		BLS	PUNCH4	PUNCH THAT MANY BYTES
FC89	C6	20	PUNCH3	LDB	#\$20	LOAD BYTE COUNT OF 32.
FC8B	E7	64	PUNCH4	STB	4,S	STORE ON STACK AS BYTE COUNT
FC8D	8E	FEED		LDX	msgid20	POINT TO MSG "S1"
FC90	17	011A		LBSR	PSTRNG	PRINT MSG
FC93	CB	03		ADDB	#3	ADD 3 BYTES TO BYTE COUNT
FC95	1F	98		TFR	B,A	GET BYTE COUNT IN A-ACC TO PUNCH
FC97	17	00E7		LBSR	OUT2H	OUTPUT BYTE COUNT
FC9A	AE	62		LDX	2,S	LOAD BEGIN ADDRESS
FC9C	17	00DA		LBSR	OUT4H	PUNCH ADDRESS
FC9F	EB	62		ADDB	2,S	ADD ADDR MSB TO CHECKSUM
FCA1	EB	63		ADDB	3,S	ADD ADDR LSB TO CHECKSUM
FCA3	EB	84	PUNCHL	ADDB	,X	ADD DATA BYTE TO CHECKSUM
FCA5	A6	80		LDA	,X+	LOAD DATA BYTE TO PUNCH
FCA7	17	00D7		LBSR	OUT2H	OUTPUT DATA BYTE
FCAA	6A	64		DEC	4,S	DECREMENT BYTE COUNT
FCAC	26	F5		BNE	PUNCHL	NOT DONE, PUNCH NEXT BYTE
FCAE	53			COMB		1's COMPLIMENT CHECKSUM BYTE
FCAF	1F	98		TFR	B,A	GET IT IN A-ACC TO PUNCH
FCB1	17	00CD		LBSR	OUT2H	OUTPUT CHECKSUM BYTE
FCB4	AF	62		STX	2,S	SAVE X-REG IN STACK AS NEW PUNCH ADDR
FCB6	AC	E4		CMPX	,S	COMPARE IT TO END ADDR
FCB8	26	C3		BNE	PUNCH2	\$FCB5 PUNCH NOT DONE, CONT.

FCBA 86	14	PUNEXT	LDA	#\$14	LOAD 'DC4' PUNCH OFF CODE
FCBC 17	012F		LBSR	OUTCH	OUTPUT IT
FCBF 32	65		LEAS	5,S	READJUST STACK POINTER
FCC1 39			RTS		
FCC2 8E	FEAE	PRTSP	LDX	#MSG10	POINT TO MSG "SP="
FCC5 17	00F5		LBSR	PDATA	PRINT MSG
FCC8 1F	31		TFR	U,X	
FCCA 16	00AC		LBRA	OUT4H	
FCCD 8E	FEBA	PRTUS	LDX	#MSG12	POINT TO MSG "US="
FCD0 17	00EA		LBSR	PDATA	PRINT MSG
FCD3 AE	48		LDX	8,U	
FCD5 16	00A1		LBRA	OUT4H	
FCD8 8E	FECC	PRTDP	LDX	#MSG15	POINT TO MSG "DP="
FCDB 17	00DF		LBSR	PDATA	PRINT MSG
FCDE A6	43		LDA	3,U	
FCE0 16	009E		LBRA	OUT2H	OUTPUT HEX BYTE AS ASCII
FCE3 8E	FEC6	PRTIX	LDX	#MSG14	POINT TO MSG "IX="
FCE6 17	00D4		LBSR	PDATA	PRINT MSG
FCE9 AE	44		LDX	4,U	\$FCE6
FCEB 16	008B		LBRA	OUT4H	
FCEE 8E	FEC0	PRTIY	LDX	#MSG13	POINT TO MSG "IY="
FCF1 17	00C9		LBSR	PDATA	PRINT MSG
FCF4 AE	46		LDX	6,U	
FCF6 16	0080		LBRA	OUT4H	
FCF9 8E	FEB4	PRTPC	LDX	#MSG11	POINT TO MSG "PC="
FCFC 17	00BE		LBSR	PDATA	PRINT MSG
FCFF AE	4A		LDX	10,U	
FD01 20	76		BRA	OUT4H	
FD03 8E	FED2	PRTA	LDX	#MSG16	POINT TO MSG "A="
FD06 17	00B4		LBSR	PDATA	PRINT MSG
FD09 A6	41		LDA	1,U	
FD0B 20	74		BRA	OUT2H	OUTPUT HEX BYTE AS ASCII
FD0D 8E	FED7	PRTB	LDX	#MSG17	POINT TO MSG "B="
FD10 17	00AA		LBSR	PDATA	PRINT MSG
FD13 A6	42		LDA	2,U	
FD15 20	6A		BRA	OUT2H	OUTPUT HEX BYTE AS ASCII
FD17 8E	FEDC	PRTCC	LDX	#MSG18	POINT TO MSG "CC:"
FD1A 17	00A0		LBSR	PDATA	PRINT MSG
FD1D A6	C4		LDA	,U	
FD1F 8E	FEE3		LDX	#MSG19	POINT TO MSG "EFHINZVC"
FD22 20	73		BRA	BIASCI	OUTPUT IN BINARY/ASCII FORMAT

* THE FOLLOWING ROUTINE LOOPS WAITING FOR THE
 * OPERATOR TO INPUT TWO VALID HEX ADDRESSES.
 * THE FIRST ADDRESS INPUT IS RETURNED IN "IY".
 * THE SECOND IS RETURNED IN "IX". THE "V" BIT
 * IN THE C-CODE REG. IS SET IF AN INVALID HEX
 * ADDRESS IS INPUT.

FD24 8D	09	IN2ADR	BSR	IN1ADR	GET FIRST ADDRESS
FD26 29	4E		BVS	NOTHEX	EXIT IF NOT VALID HEX
FD28 1F	12		TFR	X,Y	SAVE FIRST ADDR. IN "IY"

```

FD2A 86 2D LDA #' -
FD2C 17 00BF LBSR OUTCH PRINT " - "

```

* THE FOLLOWING ROUTINE LOOPS WAITING FOR THE
 * OPERATOR TO INPUT ONE VALID HEX ADDRESS. THE
 * ADDRESS IS RETURNED IN THE "X" REGISTER.

```

FD2F 8D 0F IN1ADR BSR BYTE INPUT BYTE (2 HEX CHAR)
FD31 29 43 BVS NOTHEX EXIT IF NOT VALID HEX
FD33 1F 01 TFR D,X
FD35 8D 09 BSR BYTE INPUT BYTE (2 HEX CHAR)
FD37 29 3D BVS NOTHEX
FD39 34 10 PSHS X
FD3B A7 61 STA 1,S
FD3D 35 10 PULS X
FD3F 39 RTS

```

***** INPUT BYTE (2 HEX CHAR.) *****

```

FD40 8D 11 BYTE BSR INHEX GET HEX LEFT
FD42 29 32 BVS NOTHEX EXIT IF NOT VALID HEX
FD44 48 ASLA
FD45 48 ASLA
FD46 48 ASLA SHIFT INTO LEFT NIBBLE
FD47 48 ASLA
FD48 1F 89 TFR A,B PUT HEXL IN "B"
FD4A 8D 07 BSR INHEX GET HEX RIGHT
FD4C 29 28 BVS NOTHEX EXIT IF NOT VALID HEX
FD4E 34 04 PSHS B PUSH HEXL ON STACK
FD50 AB E0 ADDA ,S+ ADD HEXL TO HEXR AND ADJ. STK
FD52 39 RTS RETURN WITH HEX L&R IN "A"

```

```

FD53 8D 6F INHEX BSR ECHON INPUT ASCII CHAR.
FD55 81 30 CMPA #'0 IS IT > OR = "0" ?
FD57 25 1D BCS NOTHEX IF LESS IT AIN'T HEX
FD59 81 39 CMPA #'9 IS IT < OR = "9" ?
FD5B 22 03 BHI INHEXA IF > MAYBE IT'S ALPHA
FD5D 80 30 SUBA #$30 ASCII ADJ. NUMERIC
FD5F 39 RTS

```

```

FD60 81 41 INHEXA CMPA #'A IS IT > OR = "A"
FD62 25 12 BCS NOTHEX IF LESS IT AIN'T HEX
FD64 81 46 CMPA #'F IS IT < OR = "F" ?
FD66 22 03 BHI INHEXL IF > IT AIN'T HEX
FD68 80 37 SUBA #$37 ASCII ADJ. ALPHA
FD6A 39 RTS

```

```

FD6B 81 61 INHEXL CMPA #'a IS IT > OR = "a"
FD6D 25 07 BCS NOTHEX IF LESS IT AIN'T HEX
FD6F 81 66 CMPA #'f IS IT < "f"
FD71 22 03 BHI NOTHEX IF > IT AIN'T HEX
FD73 80 57 SUBA #$57 ADJUST TO LOWER CASE

```

```

FD75 39                      RTS

FD76 1A  02      NOTHEX  ORCC  #2      SET (V) FLAG IN C-CODES REGISTER
FD78 39                      RTS

FD79 34  10      OUT4H   PSHS   X      PUSH X-REG. ON THE STACK
FD7B 35  02      PULS   A      POP MS BYTE OF X-REG INTO A-ACC.
FD7D 8D  02      BSR    OUTHL     OUTPUT HEX LEFT
FD7F 35  02      PULS   A      POP LS BYTE OF X-REG INTO A-ACC.
FD81 34  02      FD81  OUTHL EQU   *
FD81 34  02      OUT2H   PSHS   A      SAVE IT BACK ON STACK
FD83 44          LSRA          CONVERT UPPER HEX NIBBLE TO ASCII
FD84 44          LSRA
FD85 44          LSRA
FD86 44          LSRA
FD87 8D  04      BSR    XASCII     PRINT HEX NIBBLE AS ASCII
FD89 35  02      OUTHR   PULS   A      CONVERT LOWER HEX NIBBLE TO ASCII
FD8B 84  0F      ANDA   #$0F      STRIP LEFT NIBBLE
FD8D 8B  30      XASCII  ADDA   #$30     ASCII ADJ
FD8F 81  39      CMPA   #$39     IS IT < OR = "9" ?
FD91 2F  02      BLE    OUTC      IF LESS, OUTPUT IT
FD93 8B  07      ADDA   #7       IF > MAKE ASCII LETTER
FD95 20  57      OUTC   BRA     OUTCH    OUTPUT CHAR

```

```

* BINARY / ASCII --- THIS ROUTINE
* OUTPUTS A BYTE IN ENHANCED
* BINARY FORMAT. THE ENHANCEMENT
* IS DONE BY SUBSTITUTING ASCII
* LETTERS FOR THE ONES IN THE BYTE.
* THE ASCII ENHANCEMENT LETTERS
* ARE OBTAINED FROM THE STRING
* POINTED TO BY THE INDEX REG. "X".

```

```

FD97 34  02      BIASCI  PSHS   A      SAVE "A" ON STACK
FD99 C6  08      LDB     #8       PRESET LOOP# TO BITS PER BYTE
FD9B A6  80      OUTBA   LDA     ,X+    GET LETTER FROM STRING
FD9D 68  E4      ASL     ,S       TEST BYTE FOR "1" IN B7
FD9F 25  02      BCS     PRTBA     IF ONE PRINT LETTER
FDA1 86  2D      LDA     #' -      IF ZERO PRINT "-"
FDA3 8D  49      PRTBA   BSR     OUTCH   PRINT IT
FDA5 8D  45      BSR     OUT1S     PRINT SPACE
FDA7 5A          DECB          SUB 1 FROM #BITS YET TO PRINT
FDA8 26  F1      BNE     OUTBA
FDAA 35  02      PULS   A
FDAC 39                      RTS

```

```

* PRINT STRING PRECEDED BY A CR & LF.

```

```

FDAD 8D  02      PSTRNG  BSR     PCRLF    PRINT CR/LF
FDAF 20  0C      BRA     PDATA    PRINT STRING POINTED TO BY IX

```

```

* PCRLF

```

```

FDB1 34 10      PCRLF  PSHS  X      SAVE IX
FDB3 8E FE75      LDX   #MSG2+1  POINT TO MSG CR/LF + 3 NULS
FDB6 8D 05      BSR   PDATA  PRINT MSG
FDB8 35 10      PULS  X      RESTORE IX
FDBA 39      RTS
FDBB 8D 31      PRINT  BSR   OUTCH

```

* PDATA

```

FDBD A6 80      PDATA  LDA   ,X+      GET 1st CHAR. TO PRINT
FDBF 81 04      CMPA  #4      IS IT EOT?
FDC1 26 F8      BNE   PRINT  IF NOT EOT PRINT IT
FDC3 39      RTS

```

```

FDC4 7D DFE2    ECHON  TST   ECHO      IS ECHO REQUIRED ?
FDC7 27 06      BEQ   INCH   ECHO NOT REQ. IF CLEAR

```

* INCHE

* ---GETS CHARACTER FROM TERMINAL AND
 * ECHOS SAME. THE CHARACTER IS RETURNED
 * IN THE "A" ACCUMULATOR WITH THE PARITY
 * BIT MASKED OFF. ALL OTHER REGISTERS
 * ARE PRESERVED.

```

FDC9 8D 04      INCHE  BSR   INCH      GET CHAR FROM TERMINAL
FDCB 84 7F      ANDA  #$7F      STRIP PARITY FROM CHAR.
FDCD 20 1F      BRA   OUTCH  ECHO CHAR TO TERMINAL

```

* INCH

* GET CHARACTER FROM TERMINAL. RETURN
 * CHARACTER IN "A" ACCUMULATOR AND PRESERVE
 * ALL OTHER REGISTERS. THE INPUT CHARACTER
 * IS 8 BITS AND IS NOT ECHOED.

```

FDCF 34 10      INCH  PSHS  X      SAVE IX
FDD1 BE DFE0      LDX   CPORT  POINT TO TERMINAL PORT
FDD4 A6 84      GETSTA LDA  ,X      FETCH PORT STATUS
FDD6 85 01      BITA  #1      TEST READY BIT, RDRF ?
FDD8 27 FA      BEQ   GETSTA  IF NOT RDY, THEN TRY AGAIN
FDDA A6 01      LDA   1,X      FETCH CHAR
FDDC 35 10      PULS  X      RESTORE IX
FDDE 39      RTS

```

* INCHEK

* CHECK FOR A CHARACTER AVAILABLE FROM
 * THE TERMINAL. THE SERIAL PORT IS CHECKED
 * FOR READ READY. ALL REGISTERS ARE
 * PRESERVED, AND THE "Z" BIT WILL BE

* CLEAR IF A CHARACTER CAN BE READ.

FDDF	34	02	INCHEK	PSHS	A	SAVE A ACCUM.
FDE1	A6	9F DFE0		LDA	[CPORT]	FETCH PORT STATUS
FDE5	85	01		BITA	#1	TEST READY BIT, RDRF ?
FDE7	35	02		PULS	A	RESTORE A ACCUM.
FDE9	39			RTS		
FDEA	8D	00	OUT2S	BSR	OUT1S	OUTPUT 2 SPACES
FDEC	86	20	OUT1S	LDA	#\$20	OUTPUT 1 SPACE

* OUTCH

* OUTPUT CHARACTER TO TERMINAL.

* THE CHAR. TO BE OUTPUT IS

* PASSED IN THE A REGISTER.

* ALL REGISTERS ARE PRESERVED.

FDEE	34	12	OUTCH	PSHS	A,X	SAVE A ACCUM AND IX
FDF0	BE	DFE0		LDX	CPORT	GET ADDR. OF TERMINAL
FDF3	A6	84	FETSTA	LDA	,X	FETCH PORT STATUS
FDF5	85	02		BITA	#2	TEST TDRE, OK TO XMIT ?
FDF7	27	FA		BEQ	FETSTA	IF NOT LOOP UNTIL RDY
FDF9	35	02		PULS	A	GET CHAR. FOR XMIT
FDFB	A7	01		STA	1,X	XMIT CHAR.
FDFD	35	10		PULS	X	RESTORE IX
FDFE	39			RTS		

FE00	BE	DFE0	ACINIZ	LDX	CPORT	POINT TO CONTROL PORT ADDRESS
FE03	86	03		LDA	#3	RESET ACIA PORT CODE
FE05	A7	84		STA	,X	STORE IN CONTROL REGISTER
FE07	86	11		LDA	#\$11	SET 8 DATA, 2 STOP AN 0 PARITY
FE09	A7	84		STA	,X	STORE IN CONTROL REGISTER
FE0B	6D	01		TST	1,X	ANYTHING IN DATA REGISTER?
FE0D	86	FF		LDA	#\$FF	TURN ON ECHO FLAG
FE0F	B7	DFE2		STA	ECHO	
FE12	39			RTS		

* MONITOR KEYBOARD COMMAND JUMP TABLE

	FE13	JMPTAB	EQU	*		
FE13	01		FCB	1	" ^A "	\$F91D
FE14	F923		FDB	ALTRA		
FE16	02		FCB	2	" ^B "	\$F90F
FE17	F915		FDB	ALTRB		
FE19	03		FCB	3	" ^C "	\$F92B
FE1A	F931		FDB	ALTRCC		
FE1C	04		FCB	4	" ^D "	\$F901
FE1D	F907		FDB	ALTRDP		

FE1F 10	FCB	\$10	" ^P "	\$F8C9
FE20 F8CF	FDB	ALTRPC		
FE22 15	FCB	\$15	" ^U "	\$F8D7
FE23 F8DD	FDB	ALTRU		
FE25 18	FCB	\$18	" ^X "	\$F8F3
FE26 F8F9	FDB	ALTRX		
FE28 19	FCB	\$19	" ^Y "	\$F8E5
FE29 F8EB	FDB	ALTRY		

FE2B 42	FCC	'B'		
FE2C FA7B	FDB	BRKPNT	*\$FA78	
FE2E 44	FCC	'D'		
FE2F FAF4	FDB	DB00T	*\$FAF1	
FE31 45	FCC	'E'		
FE32 F996	FDB	MEMDUMP	*\$F990	
FE34 47	FCC	'G'		
FE35 F8A5	FDB	GO	*\$F89F	
FE37 4C	FCC	'L'		
FE38 FC0C	FDB	LOAD	*\$FC09	
FE3A 4D	FCC	'M'		
FE3B F941	FDB	MEMCHG	*\$F93B	
FE3D 50	FCC	'P'		
FE3E FC67	FDB	PUNCH	*\$FC64	
FE40 51	FCC	'Q'		
FE41 F9F2	FDB	MEMTST	*\$F9EF	
FE43 52	FCC	'R'		
FE44 F8A8	FDB	REGSTR	*\$F8A2	
FE46 53	FCC	'S'		
FE47 F98A	FDB	DISSTK	*\$F984	
FE49 55	FCC	'U'		
FE4A FBB3	FDB	MINBOOT	*\$FBB0	
FE4C 58	FCC	'X'		
FE4D FAA7	FDB	XBKPNT	*\$FAA4	

FE4F TABEND EQU *

* ** 6809 VECTOR ADDRESSES **

* FOLLOWING ARE THE ADDRESSES OF THE VECTOR ROUTINES
 * FOR THE 6809 PROCESSOR. DURING INITIALIZATION THEY
 * ARE RELOCATED TO RAM FROM \$DFC0 TO \$DFCF. THEY ARE
 * RELOCATED TO RAM SO THAT THE USER MAY REVECTOR TO
 * HIS OWN ROUTINES IF HE SO DESIRES.

FE4F FAB3	RAMVEC	FDB	SWIE	USER-V
FE51 F8A7		FDB	RTI	SWI3-V
FE53 F8A7		FDB	RTI	SWI2-V
FE55 F8A7		FDB	RTI	FIRQ-V
FE57 F8A7		FDB	RTI	IRQ-V
FE59 FAB3		FDB	SWIE	SWI-V
FE5B FFFF		FDB	FFFFF	SVC-V0
FE5D FFFF		FDB	FFFFF	SVC-VL

* PRINTABLE MESSAGE STRINGS

FE5F 00 00 00 0D	MSG1	FCB	\$0,\$0,\$0,\$D,\$A,\$0,\$0,\$0 * 0, CR/LF, 0
FE63 0A 00 00 00			
FE67 53 2D 42 55		FCC	'S-BUG 1.8 - '
FE6B 47 20 31 2E			
FE6F 38 20 2D 20			
FE73 04		FCB	4
FE74 4B 0D 0A 00	MSG2	FCB	'K,\$D,\$A,\$0,\$0,\$0,4 K, * CR/LF + 3 NULS
FE78 00 00 04			
FE7B 3E	MSG3	FCC	'>'
FE7C 04		FCB	4
FE7D 57 48 41 54	MSG4	FCC	'WHAT?'
FE81 3F			
FE82 04		FCB	4
FE83 20 2D 20	MSG5	FCC	' - '
FE86 04		FCB	4
FE87 2C 20 50 41	MSG6	FCC	', PASS '
FE8B 53 53 20			
FE8E 04		FCB	4
FE8F 2C 20 42 49	MSG7	FCC	', BITS IN ERROR: '
FE93 54 53 20 49			
FE97 4E 20 45 52			
FE9B 52 4F 52 3A			
FE9F 20			
FEA0 04		FCB	4
FEA1 20 3D 3E 20	MSG8	FCC	' => '
FEA5 04		FCB	4
FEA6 37 36 35 34	MSG9	FCC	'76543210'
FEAA 33 32 31 30			
FEAE 20 20 53 50	MSG10	FCC	' SP= '
FEB2 3D			
FEB3 04		FCB	4
FEB4 20 20 50 43	MSG11	FCC	' PC= '
FEB8 3D			
FEB9 04		FCB	4
FEBA 20 20 55 53	MSG12	FCC	' US= '
FEBE 3D			
FEBF 04		FCB	4
FEC0 20 20 49 59	MSG13	FCC	' IY= '
FEC4 3D			
FEC5 04		FCB	4
FEC6 20 20 49 58	MSG14	FCC	' IX= '
FECA 3D			
FECB 04		FCB	4
FECB 20 20 44 50	MSG15	FCC	' DP= '
FED0 3D			
FED1 04		FCB	4
FED2 20 20 41 3D	MSG16	FCC	' A= '
FED6 04		FCB	4
FED7 20 20 42 3D	MSG17	FCC	' B= '
FEDB 04		FCB	4
FEDC 20 20 43 43	MSG18	FCC	' CC: '
FEE0 3A 20			

```

FEE2 04          FCB      4
FEE3 45 46 48 49 MSG19  FCC      'EFHINZVC'
FEE7 4E 5A 56 43
FEEB 53 31      MSG20  FCC      'S1'
FEED 04          FCB      4

```

* MESSAGE EXPANSION AREA

```

EEEE FF FF FF FF          FCB      $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
FEF2 FF FF FF FF
FEF6 FF FF FF FF          FCB      $FF,$FF,$FF,$FF,$FF,$FF,$FF
FEFA FF FF FF

```

* POWER UP/ RESET/ NMI ENTRY POINT

```

FF00          ORG      $FF00

```

```

FF00 8E      FFF0      START  LDX      #IC11      POINT TO DAT RAM IC11
FF03 86      0F          LDA      #$F          GET COMPLIMENT OF ZERO

```

* INITIALIZE DAT RAM --- LOADS \$F-\$0 IN LOCATIONS \$0-\$F
 * OF DAT RAM, THUS STORING COMPLEMENT OF MSB OF ADDRESS
 * IN THE DAT RAM. THE COMPLEMENT IS REQUIRED BECAUSE THE
 * OUTPUT OF IC11, A 74S189, IS THE INVERSE OF THE DATA
 * STORED IN IT.

```

FF05 A7      80      DATLP  STA      ,X+      STORE & POINT TO NEXT RAM LOCATION
FF07 4A          DECA      GET COMP. VALUE FOR NEXT LOCATION
FF08 26      FB      BNE      DATLP      ALL 16 LOCATIONS INITIALIZED ?

```

* NOTE: IX NOW CONTAINS \$0000, DAT RAM IS NO LONGER
 * ADDRESSED, AND LOGICAL ADDRESSES NOW EQUAL
 * PHYSICAL ADDRESSES.

```

FF0A 86      F0          LDA      #$F0
FF0C A7      84          STA      ,X      STORE $F0 AT $FFFF
FF0E 8E      D0A0        LDX      #$D0A0  ASSUME RAM TO BE AT $D000-$DFFF
FF11 108E 55AA          LDY      #TSTPAT  LOAD TEST DATA PATTERN INTO "Y"
FF15 EE      84      TSTRAM LDU      ,X      SAVE DATA FROM TEST LOCATION
FF17 10AF 84          STY      ,X      STORE TEST PATTERN AT $D0A0
FF1A 10AC 84          CMPY     ,X      IS THERE RAM AT THIS LOCATION ?
FF1D 27      0B          BEQ      CNVADR   IF MATCH THERE'S RAM, SO SKIP
FF1F 30      89 F000      LEAX     -$1000,X ELSE POINT 4K LOWER
FF23 8C      F0A0        CMPX     #$F0A0  DECREMENTED PAST ZERO YET ?
FF26 26      ED          BNE      TSTRAM   IF NOT CONTINUE TESTING FOR RAM
FF28 20      D6          BRA      START   ELSE START ALL OVER AGAIN

```

* THE FOLLOWING CODE STORES THE COMPLEMENT OF
 * THE MS CHARACTER OF THE FOUR CHARACTER HEX
 * ADDRESS OF THE FIRST 4K BLOCK OF RAM LOCATED

* BY THE ROUTINE "TSTRAM" INTO THE DAT RAM. IT
 * IS STORED IN RAM IN THE LOCATION THAT IS
 * ADDRESSED WHEN THE PROCESSOR ADDRESS IS \$D---,
 * THUS IF THE FIRST 4K BLOCK OF RAM IS FOUND
 * WHEN TESTING LOCATION \$70A0, MEANING THERE
 * IS NO RAM PHYSICALLY ADDRESSED IN THE RANGE
 * \$8000-\$DFFF, THEN THE COMPLEMENT OF THE
 * "7" IN THE \$70A0 WILL BE STORED IN
 * THE DAT RAM. THUS WHEN THE PROCESSOR OUTPUTS
 * AN ADDRESS OF \$D---, THE DAT RAM WILL RESPOND
 * BY RECOMPLEMENTING THE "7" AND OUTPUTTING THE
 * 7 ONTO THE A12-A15 ADDRESS LINES. THUS THE
 * RAM THAT IS PHYSICALLY ADDRESSED AT \$7---
 * WILL RESPOND AND APPEAR TO THE 6809 THAT IT
 * IS AT \$D--- SINCE THAT IS THE ADDRESS THE
 * 6809 WILL BE OUTPUTTING WHEN THAT 4K BLOCK
 * OF RAM RESPONDS.

FF2A EF	84	CNVADR	STU	,X	RESTORE DATA AT TEST LOCATION
FF2C 1F	10		TFR	X,D	PUT ADDR. OF PRESENT 4K BLOCK IN D
FF2E 43			COMA		COMPLEMENT MSB OF THAT ADDRESS
FF2F 44			LSRA		PUT MS 4 BITS OF ADDRESS IN
FF30 44			LSRA		LOCATION D0-D3 TO ALLOW STORING
FF31 44			LSRA		IT IN THE DYNAMIC ADDRESS
FF32 44			LSRA		TRANSLATION RAM.
FF33 B7	FFFD		STA	\$FFFD	STORE XLATION FACTOR IN DAT "D"
FF36 10CE	DFC0		LDS	#STACK	INITIALIZE STACK POINTER

* THE FOLLOWING CHECKS TO FIND THE REAL PHYSICAL ADDRESSES
 * OF ALL 4K BLKS OF RAM IN THE SYSTEM. WHEN EACH 4K BLK
 * OF RAM IS LOCATED, THE COMPLEMENT OF IT'S REAL ADDRESS
 * IS THEN STORED IN A "LOGICAL" TO "REAL" ADDRESS XLATION
 * TABLE THAT IS BUILT FROM \$DFD0 TO \$DFDF. FOR EXAMPLE IF
 * THE SYSTEM HAS RAM THAT IS PHYSICALLY LOCATED (WIRED TO
 * RESPOND) AT THE HEX LOCATIONS \$0--- THRU \$F-----.

* 0 1 2 3 4 5 6 7 8 9 A B C D E F
 * 4K 4K 4K 4K 4K 4K 4K 4K -- 4K 4K 4K 4K -- -- --

*FOR A TOTAL OF 48K OF RAM, THEN THE TRANSLATION TABLE
 * CREATED FROM \$DFD0 TO \$DFDF WILL CONSIST OF THE FOLLOWING....

* 0 1 2 3 4 5 6 7 8 9 A B C D E F
 * 0F 0E 0D 0C 0B 0A 09 08 06 05 00 00 04 03 F1 F0

* HERE WE SEE THE LOGICAL ADDRESSES OF MEMORY FROM \$0000-\$7FFF
 * HAVE NOT BEEN SELECTED FOR RELOCATION SO THAT THEIR PHYSICAL
 * ADDRESS WILL = THEIR LOGICAL ADDRESS; HOWEVER, THE 4K BLOCK
 * PHYSICALLY AT \$9000 WILL HAVE ITS ADDRESS TRANSLATED SO THAT
 * IT WILL LOGICALLY RESPOND AT \$8000. LIKEWISE \$A,\$B, AND \$C000

* WILL BE TRANSLATED TO RESPOND TO \$9000,\$C000, AND \$D000
 * RESPECTIVELY. THE USER SYSTEM WILL LOGICALLY APPEAR TO HAVE
 * MEMORY ADDRESSED AS FOLLOWS....

* 0 1 2 3 4 5 6 7 8 9 A B C D E F
 * 4K 4K 4K 4K 4K 4K 4K 4K 4K 4K -- -- 4K 4K -- --

FF3A	108E	DFD0		LDY	#LRARAM	POINT TO LOGICAL/REAL ADDR. TABLE
FF3E	A7	2D		STA	13,Y	STORE \$D--- XLATION FACTOR AT \$DFDD
FF40	6F	2E		CLR	14,Y	CLEAR \$DFDE
FF42	86	F0		LDA	#\$F0	DESTINED FOR IC8 AN MEM EXPANSION ?
FF44	A7	2F		STA	15,Y	STORE AT \$DFDF
FF46	86	0C		LDA	#\$0C	PRESET NUMBER OF BYTES TO CLEAR
FF48	6F	A6	CLRLRT	CLR	A,Y	CLEAR \$DFDC THRU \$DFD0
FF4A	4A			DECA		SUB. 1 FROM BYTES LEFT TO CLEAR
FF4B	2A	FB		BPL	CLRLRT	CONTINUE IF NOT DONE CLEARING
FF4D	30	89 F000	FNDRAM	LEAX	-\$1000,X	POINT TO NEXT LOWER 4K OF RAM
FF51	8C	F0A0		CMPX	#\$F0A0	TEST FOR DECREMENT PAST ZERO
FF54	27	22		BEQ	FINTAB	SKIP IF FINISHED
FF56	EE	84		LDU	,X	SAVE DATA AT CURRENT TEST LOCATION
FF58	108E	55AA		LDY	#TSTPAT	LOAD TEST DATA PATTERN INTO Y REG.
FF5C	10AF	84		STY	,X	STORE TEST PATT. INTO RAM TEST LOC.
FF5F	10AC	84		CMPY	,X	VERIFY RAM AT TEST LOCATION
FF62	26	E9		BNE	FNDRAM	IF NO RAM GO LOOK 4K LOWER
FF64	EF	84		STU	,X	ELSE RESTORE DATA TO TEST LOCATION
FF66	108E	DFD0		LDY	#LRARAM	POINT TO LOGICAL/REAL ADDR. TABLE
FF6A	1F	10		TFR	X,D	PUT ADDR. OF PRESENT 4K BLOCK IN D
FF6C	44			LSRA		PUT MS 4 BITS OF ADDR. IN LOC. D0-D3
FF6D	44			LSRA		TO ALLOW STORING IT IN THE DAT RAM.
FF6E	44			LSRA		
FF6F	44			LSRA		
FF70	1F	89		TFR	A,B	SAVE OFFSET INTO LRARAM TABLE
FF72	88	0F		EORA	#\$0F	INVERT MSB OF ADDR. OF CURRENT 4K BLK
FF74	A7	A5		STA	B,Y	SAVE TRANSLATION FACTOR IN LRARAM TABLE
FF76	20	D5		BRA	FNDRAM	GO TRANSLATE ADDR. OF NEXT 4K BLK
FF78	86	F1	FINTAB	LDA	#\$F1	DESTINED FOR IC8 AND MEM EXPANSION ?
FF7A	108E	DFD0		LDY	#LRARAM	POINT TO LRARAM TABLE
FF7E	A7	2E		STA	14,Y	STORE \$F1 AT \$DFCE

* THE FOLLOWING CHECKS TO SEE IF THERE IS A 4K BLK OF
 * RAM LOCATED AT \$C000-\$CFFF. IF NONE THERE IT LOCATES
 * THE NEXT LOWER 4K BLK AN XLATES ITS ADDR SO IT
 * LOGICALLY RESPONDS TO THE ADDRESS \$C---.

FF80	86	0C		LDA	#\$0C	PRESET NUMBER HEX "C"
FF82	E6	A6	FINDC	LDB	A,Y	GET ENTRY FROM LRARAM TABLE
FF84	26	05		BNE	FOUNDNC	BRANCH IF RAM THIS PHYSICAL ADDR.
FF86	4A			DECA		ELSE POINT 4K LOWER
FF87	2A	F9		BPL	FINDC	GO TRY AGAIN
FF89	20	14		BRA	XFERTF	
FF8B	6F	A6	FOUNDNC	CLR	A,Y	CLR XLATION FACTOR OF 4K BLOCK FOUND
FF8D	E7	2C		STB	\$C,Y	GIVE IT XLATION FACTOR MOVING IT TO \$C---

* THE FOLLOWING CODE ADJUSTS THE TRANSLATION
 * FACTORS SUCH THAT ALL REMAINING RAM WILL
 * RESPOND TO A CONTIGUOUS BLOCK OF LOGICAL
 * ADDRESSES FROM \$0000 AND UP....

FF8F	4F		CLRA		START AT ZERO
FF90	1F	21	TFR	Y,X	START POINTER "X" START OF "LRARAM"

TABLE.

FF92	E6	A6	COMPRS	LDB	A,Y	GET ENTRY FROM "LRARAM" TABLE
FF94	27	04		BEQ	PNTNXT	IF IT'S ZERO SKIP
FF96	6F	A6		CLR	A,Y	ELSE ERASE FROM TABLE
FF98	E7	80		STB	,X+	AND ENTER ABOVE LAST ENTRY- BUMP
FF9A	4C		PNTNXT	INCA		GET OFFSET TO NEXT ENTRY
FF9B	81	0C		CMPA	#\$0C	LAST ENTRY YET ?
FF9D	2D	F3		BLT	COMPRS	

* THE FOLLOWING CODE TRANSFER THE TRANSLATION
 * FACTORS FROM THE LRARAM TABLE TO IC11 ON
 * THE MP-09 CPU CARD.

FF9F	8E	FFF0	XFERTF	LDX	#IC11	POINT TO DAT RAM IC11
FFA2	C6	10		LDB	#\$10	GET NO. OF BYTES TO MOVE
FFA4	A6	A0	FETCH	LDA	,Y+	GET BYTE AND POINT TO NEXT
FFA6	A7	80		STA	,X+	POKE XLATION FACTOR IN IC11
FFA8	5A			DECB		SUB 1 FROM BYTES TO MOVE
FFA9	26	F9		BNE	FETCH	CONTINUE UNTIL 16 MOVED
FFAB	53			COMB		SET "B" NON-ZERO
FFAC	F7	DFE2		STB	ECHO	TURN ON ECHO FLAG
FFAF	16	F862		LBRA	MONITOR	INITIALIZATION IS COMPLETE

FFB2	6E	9F DFC0	V1	JMP	[STACK]
FFB6	6E	9F DFC4	V2	JMP	[SWI2]
FFBA	6E	9F DFC6	V3	JMP	[FIRQ]
FFBE	6E	9F DFC8	V4	JMP	[IRQ]
FFC2	6E	9F DFCA	V5	JMP	[SWI]

* SWI3 ENTRY POINT

FFC6	1F	43	SWI3E	TFR	S,U	
FFC8	AE	4A		LDX	10,U	*\$FFC8
FFCA	E6	80		LDB	,X+	
FFCC	AF	4A		STX	10,U	
FFCE	4F			CLRA		
FFCF	58			ASLB		
FFD0	49			ROLA		
FFD1	BE	DFCC		LDX	SVCV0	
FFD4	8C	FFFF		CMPX	#\$FFFF	
FFD7	27	0F		BEQ	SWI3Z	
FFD9	30	8B		LEAX	D,X	
FFDB	BC	DFCE		CMPX	SVCVL	
FFDE	22	08		BHI	SWI3Z	
FFE0	34	10		PSHS	X	
FFE2	EC	C4		LDD	,U	

FFE4	AE	44		LDX	4,U
FFE6	6E	F1		JMP	[,S++]
FFE8	37	1F	SWI3Z	PULU	A,B,X,CC,DP
FFEA	EE	42		LDU	2,U
FFEC	6E	9F	DFC2	JMP	[SWI3]

* 6809 VECTORS

FFF0	FFB2	FDB	V1	USER-V
FFF2	FFC6	FDB	SWI3E	SWI3-V
FFF4	FFB6	FDB	V2	SWI2-V
FFF6	FFBA	FDB	V3	FIRQ-V
FFF8	FFBE	FDB	V4	IRQ-V
FFFA	FFC2	FDB	V5	SWI-V
FFFC	FFB2	FDB	V1	NMI-V
FFFE	FF00	FDB	START	RESTART-V
		END		

SYMBOL TABLE:

AAAREG F015	ACIAS E004	ACINIZ FE00	ADDREG F000	ADJSK6 FA78
AJDUMP F9A2	ALTAD F930	ALTB D F922	ALTCCD F940	ALDTPD F914
ALTPCD F8DC	ALTRA F923	ALTRB F915	ALTRCC F931	ALTRDP F907
ALTRPC F8CF	ALTRU F8DD	ALTRX F8F9	ALTRY F8EB	ALTUD F8EA
ALTXD F906	ALTYD F8F8	BACK F986	BBBREG F016	BIASCI FD97
BPADJ FAF1	BPERR FA9F	BPTBL DFE3	BPTTEST FAE1	BRKPNT FA7B
BYTE FD40	CCCREG F040	CCREG F010	CHANGE F974	CHRTN F973
CLRLRT FF48	CLRSTK F82F	CNTREG F002	CNVADR FF2A	COMPRS FF92
COMREG F020	CPORT DFE0	Comreg E018	DATLP FF05	DBOOT FAF4
DBOOT0 FB12	DBOOT1 FB58	DBOOT2 FB5B	DBOOT3 FB6A	DBOOT4 FB74
DELAY FC06	DISKWT FB1F	DISSTK F98A	DLY FBA9	DRVREG F024
Datreg E01B	Drvreg E014	ECHO DFE2	ECHON FDC4	EDPASC F9DE
EDPRTN F9A1	EDUMP F9BD	ELLOOP F9CC	EXITBP FA9E	FETCH FFA4
FETSTA FDF3	FFSTBL FAD8	FINDC FF82	FINTAB FF78	FIRQ DFC6
FLEX 0005	FNDBP FAE7	FNDRAM FF4D	FNDREL F84E	FORWRD F982
FOUND C FF8B	GETSTA FDD4	GO F8A5	GU DPAS FA5E	IC11 FFF0
IN1ADR FD2F	IN2ADR FD24	INCH FDC F	INCHE FDC9	INCHEK FDDF
INHEX FD53	INHEXA FD60	INHEXL FD6B	IRQ DFC8	JMPCMD F8A1
JMPTAB FE13	LOAD FC0C	LOAD1 FC14	LOAD10 FC3E	LOAD16 FC54
LOAD2 FC17	LOAD21 FC5F	LODERR FC53	LOOP FBBE	LOOP1 FBCF
LOOP2 FBE9	LOOP3 FBF2	LOOP4 FBFE	LOOP5 FC08	LOOPA F81D
LRA FB81	LRARAM FDF0	MDUMP1 F99B	MEMC2 F948	MEMCHG F941
MEMDUM F996	MEMSET FA04	MEMTST F9F2	MINB00 FBB3	MONITO F814
MSG1 FE5F	MSG10 FEAE	MSG11 FEB4	MSG12 FEBA	MSG13 FEC0
MSG14 FEC6	MSG15 FECC	MSG16 FED2	MSG17 FED7	MSG18 FEDC
MSG19 FEE3	MSG2 FE74	MSG20 FEEB	MSG3 FE7B	MSG4 FE7D
MSG5 FE83	MSG6 FE87	MSG7 FE8F	MSG8 FEA1	MSG9 FEA6
NEXTCM F861	NOTHEX FD76	NXTCH0 F88B	NXTCHR F88E	NXTLIN F9B1
OUT1S FDEC	OUT2H FD81	OUT2S FDEA	OUT4H FD79	OUTBA FD9B
OUTC FD95	OUTCH FDEE	OUTHL FD81	OUTHR FD89	PCRLF FDB1
PDATA FDBD	PERIOD F9E8	PNTNXT FF9A	PRASC F9EA	PRINT FDBB
PRIREG F014	PRTA FD03	PRTB FD0D	PRTBA FDA3	PRTCC FD17
PRTCMD F87F	PRTDP FCD8	PRTIX FCE3	PRTIY FCEE	PRTPC FCF9
PRTSP FCC2	PRTUS FCCD	PSTRNG FDAD	PUNCH FC67	PUNCH2 FC7D
PUNCH3 FC89	PUNCH4 FC8B	PUNCHL FCA3	PUNEXT FCBA	RAMVEC FE4F
REGPR FAC1	REGSTR F8A8	REL PAS F855	RPLSWI FAC7	RTI F8A7
SECREG F022	SKPDMP F9BA	STACK DFC0	START FF00	SUB1 FBAD
SVCVL DFCE	SVCVO DFCC	SWI DFCA	SWI2 DFC4	SWI3 DFC2
SWI3E FFC6	SWI3Z FFE8	SWIE FAB3	Secreg E01A	TABEND FE4F
TEST1 FA16	TSTPAT 55AA	TSTRAM FF15	V1 FFB2	V2 FFB6
V3 FFBA	V4 FFBE	V5 FFC2	XASCII FD8D	XBKPNT FAA7
XBPLP FAAD	XFERTF FF9F			