- \* 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 SWIPC o INCH FLOPPY SYSTEM

\* U = BOOT A SWIPC 5 INCH FLOPPY SYSTEM

\*\* D = BOOT A SWIPC o INCH FLOPPY SYSTEM

\*\* D = BOOT A SWIPC o 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	SVCVO	RMB	2	SUPERVISOR CALL VECTOR ORGIN
DFCE	SVCVL	RMB	2	SUPERVISOR CALL VECTOR LIMIT

DFD0 DFE0 DFE2 DFE3	E004 E018 E014	LRARAM CPORT ECHO BPTBL ACIAS Comreg Drvreg		16 2 1 24 \$E004 \$E018 \$E014	COMMAND REGISTER DRIVE REGISTER
	E01A E01B	Secreg Datreg		\$E01A \$E01B	SECTOR REGISTER DATA REGISTER
	FOID	Datteg	БQU	SECTE	DATA REGISTER
	F000 F002 F010 F014 F015 F016 F020 F022 F024 F040	ADDREG CNTREG CCREG PRIREG AAAREG BBBREG COMREG SECREG DRVREG CCCREG	EQU EQU EQU EQU	\$F000 \$F002 \$F010 \$F014 \$F015 \$F016 \$F020 \$F022 \$F024 \$F040 \$FFF0	ADDRESS REGISTER COUNT REGISTER CHANNEL CONTROL REGISTER DMA PRIORITY REGISTER ??? ??? 1791 COMMAND REGISTER SECTOR REGISTER DRIVE SELECT LATCH ??? DAT RAM CHIP
E000			ODG	d m o o o	
F800 F800 F814			ORG FDB	\$F800 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 F810 FDAD			FDB	PCRLF	
F810 FDAD F812 FB81			FDB FDB	PSTRNG LRA	
LOTZ LDOT			I DD	TIVA	
			100		

- \* MONITOR
- \* VECTOR ADDRESS STRING IS.....
- \* \$F8A1-\$F8A1-\$F8A1-\$F8A1-\$FAB0-\$FFFF-\$FFFF

F814 8E F817 108E		MONITOR	LDY	#RAMVEC #STACK	POINT TO I	VECTOR ADDR. STRING
F81B C6	10		LDB	#\$10	BYTES TO I	
F81D A6	80	LOOPA	LDA	, X+	GET VECTOR	R BYTE
F81F A7	A0		STA	, Y+	PUT VECTOR	RS IN RAM / \$DFC0-\$DFCF
F821 5A			DECB		SUBTRACT :	FROM NUMBER OF BYTES TO MOVE
F822 26	F9		BNE	LOOPA	CONTINUE U	JNTIL ALL VECTORS MOVED
		* CONTE	NTS	FROM	TO	FUNCTION
		* \$F8A	L	\$FE40	\$DFC0	USER-V
		* \$F8A	L	\$FE42	\$DFC2	SWI3-V
		* \$F8A1	L	\$FE44	\$DFC4	SWI2-V
		* \$F8A1	L	\$FE46	\$DFC6	FIRQ-V
		* \$F8A	L	\$FE48	\$DFC8	IRQ-V

		* \$FAB( * \$FFF) * \$FFF)	F	\$FE4A \$FE4C \$FE4E	\$DFCA \$DFCC \$DFCE	SWI-V SVC-VO SVC-VL
F824 8E F827 BF	E004 DFE0		LDX STX	#ACIAS CPORT	GET CONTROL STORE ADDR	L PORT ADDR. . IN RAM
F82A 17	027A		LBSR	XBKPNT	CLEAR OUTS	TANDING BREAKPOINTS
F82D C6	0C		LDB	#12	CLEAR 12 BY	YTES ON STACK
F82F 6F	E2	CLRSTK	CLR	,-S		
F831 5A F832 26	ED		DECB BNE	CLRSTK		
F834 30	FB 8C DD		LEAX		CR SET DC T(	O SBUG-E ENTRY
F837 AF	6A		STX	10,S	ON STACK	5 5500 E ENTRI
F839 86	D0		LDA	#\$D0		DITION CODES ON STACK
F83B A7	E4		STA	,S		
F83D 1F	43		TFR	S,U		
F83F 17	05BE		LBSR	ACINIZ		CONTROL PORT
F842 8E	FE5F 0575		LDX	#MSG1		SBUG 1.8' MESSAGE
F845 17 F848 8E	DFD0		LBSR LDX	PDATA #LRARAM	PRINT MSG	RA RAM STORAGE AREA
F84B 4F	DIDO		CLRA	πыкмкми	START TOTAL	
F84C C6	0D		LDB	#13		LL ACTIVE RAM MEMORY
F84E 6D	85	FNDREL	TST	B,X	TEST FOR RA	AM AT NEXT LOC.
F850 27	03		BEQ	RELPAS	_	GO TO NEXT LOC.
F852 8B	04		ADDA	#4	ELSE ADD 41	
F854 19			DAA			FOR DECIMAL
F855 5A F856 2A	F6	RELPAS	DECB BPL	ENDDET	PRINT TOTAL	M LOCS. TO TEST
F858 17	0526		LBSR	FNDREL OUT2H		BYTE AS ASCII
F85B 8E	FE74		LDX	#MSG2		SG 'K' CR/LF + 3 NULS
F85E 17	055C		LBSR	PDATA	PRINT MSG	
		**** N	EXTCMD	****		
F861 8E	FE7B	NEXTCMD	LDX	#MSG3	POINT TO M	SG ">"
F864 17	0546		LBSR	PSTRNG	PRINT MSG	
F867 17	0565		LBSR	INCH		AR. FROM TERMINAL
F86A 84	7F		ANDA	#\$7F		TY FROM CHAR.
F86C 81	0D		CMPA	#\$0D		IAGE RETURN ?
F86E 27 F870 1F	F1 89		BEQ TFR	NEXTCMD A,B		GET ANOTHER CHAR. IN "B" ACCUM.
F870 IF	20		CMPA	#\$20		ROL OR DATA CHAR ?
F874 2C	09		BGE	PRTCMD		R IS DATA, PRNT IT
F876 86	5E		LDA	# ' ^		CHAR CMD SO
F878 17	0573		LBSR	OUTCH	PRINT "^"	
F87B 1F	98		TFR	B,A		RL CMD CHAR
F87D 8B	40		ADDA	#\$40		TO ASCII LETTER
F87F 17	056C	PRTCMD	LBSR	OUTCH	PRNT CMD CI	HAR
F882 17 F885 C1	0567 60		LBSR CMPB	OUT1S #\$60	PRNT SPACE	
F887 2F	02		BLE	msou NXTCH0		
F889 C0	20		SUBB	#\$20		
				•		

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

F88E E1 80 F890 27 OF F892 30 02 F894 8C FE F897 26 F5 F899 8E FE	44F 47D 1E JMPCMD	LDX CMPB BEQ LEAX CMPX BNE LDX LBSR BRA JSR BRA O OR COM	#JMPTAB ,X+ JMPCMD 2,X #TABEND NXTCHR #MSG4 PDATA NEXTCMD [,X] NEXTCMD	POINT TO JUMP TABLE DOES COMMAND MATCH TABLE ENTRY? BRANCH IF MATCH FOUND POINT TO NEXT ENTRY IN TABLE REACHED END OF TABLE YET? IF NOT END, CHECK NEXT ENTRY POINT TO MSG "WHAT?" PRINT MSG IF NO MATCH, PRMPT FOR NEW CMD JUMP TO COMMAND ROUTINE PROMPT FOR NEW COMMAND
F8A5 1F 34 F8A7 3B	GO RTI	TFR RTI	U,S	
	* "R" D	ISPLAY F	REGISTERS	
F8AB 17 04 F8AE 17 04 F8B1 17 04 F8B4 17 04 F8BA 17 04 F8BA 17 04 F8BD 8E FE F8C0 17 04 F8C3 17 04 F8C6 17 04 F8C9 17 04	119 221 229 331 883 EA 333 33A 441	LDX LBSR LBSR LBSR LBSR LBSR LBSR LDX LBSR LBSR LBSR LBSR LBSR	#MSG5 PSTRNG PRTSP PRTUS PRTUP PRTIX PRTIY #MSG5 PSTRNG PRTPC PRTA PRTB PRTCC	POINT TO MSG " - " PRINT MSG \$FCBF \$FCCA \$FCD5 \$FCE0 \$FCEB POINT TO MSG " - " PRINT MSG \$FCF5 \$FCFF \$FD09 \$FD13
	* ALTER	"PC" PF	ROGRAM COUN	NTER
F8D2 17 05		LBSR LBSR LBSR BVS STX RTS	PRTPC OUT1S IN1ADR ALTPCD 10,U	\$FCF5 PRINT MSG " PC = " OUTPUT SPACE GET NEW CONTENTS FOR "PC" EXIT IF INVALID HEX POKE IN NEW CONTENTS
	* ALTER	"U" USE	ER STACK PO	DINTER
	ED ALTRU	LBSR LBSR	PRTUS OUT1S	\$FCCA PRINT MSG " US = " OUTPUT SPACE

BUG18	MP-09	MONITOR			9-5-00	TSC ASSEMBLER	PAGE	5
F8E3 F8E6 F8E8 F8EA	29 AF	0449 02 48	ALTUD	STX	IN1ADR ALTUD 8,U			
			* ALTER	"Y" INI	DEX REGIST	ER		
F8EB F8EE F8F1 F8F4	17	0400 04FB 043B 02	ALTRY	LBSR LBSR	PRTIY OUT1S IN1ADR ALTYD	PRINT MSG " IY OUTPUT SPACE	= "	
F8F6 F8F8		46	ALTYD		6,U	\$F8F0		
			* ALTER	"X" INI	DEX REGIST	ER		
F8F9 F8FC		03E7 04ED	ALTRX	LBSR LBSR	PRTIX OUT1S	\$FCE0 PRINT MSO	G " IX =	II

*	ALTER	"X"	INDEX	REGISTER
---	-------	-----	-------	----------

SBUG18 MP-09 MONITOR

	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	<b>4</b> ,U	
F906	39		ALTXD	RTS		

### \* ALTER "DP" DIRECT PAGE REGISTER

F907 1	7 03CE	ALTRDP	LBSR	PRTDP	\$FCD5 PRINT MSG " DP = "
F90A 1	7 04DF	1	LBSR	OUT1S	OUTPUT SPACE
F90D 1	7 0430	)	LBSR	BYTE	INPUT BYTE (2 HEX CHAR)
F910 2	9 02		BVS	ALTDPD	
F912 A	7 43		STA	3,U	
F914 3	9	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 F929 17 F92C 29 F92E A7 F930 39	04C3 0414 02 41	ALTAD	LBSR LBSR BVS STA RTS	OUT1S BYTE ALTAD 1,U	OUTPUT SPACE INPUT BYTE (2 HEX CHAR)
		* ALTER	2 "CC" R	REGISTER	
F931 17 F934 17 F937 17 F93A 29 F93C 8A F93E A7 F940 39	03E3 04B5 0406 04 80 C4	ALTCCD	LBSR LBSR BVS ORA STA RTS	PRTCC OUT1S BYTE ALTCCD #\$80 ,U	
E041 17	0250				AND CHANGE ****
F941 17 F944 29 F946 1F	03EB 2D 12	MEMCHG	LBSR BVS TFR	IN1ADR CHRTN X,Y	INPUT ADDRESS IF NOT HEX, RETURN SAVE ADDR IN "Y"
F948 8E F94B 17 F94E 1F F950 17 F953 17 F956 A6 F958 17 F95E 17 F95E 17 F961 28 F963 81 F965 27 F967 81 F969 27 F968 81 F960 27 F968 81 F960 27 F967 81 F960 27 F967 81 F967 339	FE83 045F 21 0426 0496 A4 0426 048E 03DF 11 08 E1 18 DD 5E 17	MEMC2	LDX LBSR TFR LBSR LBSR LDA LBSR LBSR CBSR CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ	#MSG5 PSTRNG Y,X OUT4H OUT1S ,Y OUT2H OUT1S BYTE CHANGE #8 MEMC2 #\$18 MEMC2	POINT TO MSG " - " PRINT MSG FETCH ADDRESS PRINT ADDR IN HEX OUTPUT SPACE GET CONTENTS OF CURRENT ADDR. OUTPUT CONTENTS IN ASCII OUTPUT SPACE LOOP WAITING FOR OPERATOR INPUT IF VALID HEX GO CHANGE MEM. LOC. IS IT A BACKSPACE (CNTRL H)? PROMPT OPERATOR AGAIN IS IT A CANCEL (CNTRL X)? PROMPT OPERATOR AGAIN IS IT AN UP ARROW? DISPLAY PREVIOUS BYTE IS IT A CR? DISPLAY NEXT BYTE EXIT ROUTINE
F974 A7 F976 A1 F978 27 F97A 17 F97D 86 F97F 17 F982 31 F984 20 F986 31 F988 20	A4 A4 08 046F 3F 046C 21 C2 3F BE	CHANGE FORWRD BACK	STA CMPA BEQ LBSR LDA LBSR LEAY BRA LEAY BRA	,Y ,Y FORWRD OUT1S #'? OUTCH 1,Y MEMC2 -1,Y	CHANGE BYTE IN MEMORY DID MEMORY BYTE CHANGE? \$F972 OUTPUT SPACE LOAD QUESTION MARK PRINT IT POINT TO NEXT HIGHER MEM LOCATION PRINT LOCATION & CONTENTS POINT TO LAST MEM LOCATION 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		E4		LDX	,S	LOAD LOWER ADDR FROM TOP OF STACK
F9C7		03AF		LBSR		PRINT THE ADDRESS LBSR OUT2S PRINT 2
SPACES						
F9CA	C6	10		LDB	#\$10	LOAD COUNT OF 16 BYTES TO DUMP
F9CC			ELOOP		, X+	GET FROM MEMORY HEX BYTE TO PRINT
F9CE		03B0		LBSR	OUT2H	OUTPUT HEX BYTE AS ASCII
F9D1		0418		LBSR	OUT1S	OUTPUT SPACE
F9D4		0110		DECB	00110	\$F9D1 DECREMENT BYTE COUNT
F9D5		F5		BNE	ELOOP	CONTINUE TIL 16 HEX BYTES PRINTED
1 1 1 1 1	20	1 3		DIVI	ппоот	CONTINUE TIE TO HER BITED TRINIED
			* PRINT	16 ASC	II CHARACTI	ERS
					ABLE OR NOT	
					A PERIOD (	
F9D7	17	0410	112011	LBSR	OUT2S	•
F9DA		E1				GET LOW LIMIT FRM STACK - ADJ STACK
F9DC		10		LDB		SET ASCII CHAR TO PRINT = 16
F9DE		80	EDPASC			GET CHARACTER FROM MEMORY
F9E0		20	EDPASC	CMPA	, <u>∧</u> ⊤ #୯၁∩	IF LESS THAN \$20, NON-PRINTABLE?
F9E0 F9E2		04			#\$20	
				BCS	PERIOD	IF SO, PRINT PERIOD INSTEAD
F9E4		7E		CMPA BLS	#\$7E PRASC	IS IT VALID ASCII?
F9E6		02				IF SO PRINT IT
F9E8			PERIOD			LOAD A PERIOD (.)
F9EA		0401	PRASC	LBSR	OUTCH	PRINT ASCII CHARACTER
F9ED				DECB		DECREMENT COUNT
F9EE		EE		BNE	EDPASC	
F9F0	20	BF		BRA	NXTLIN	
			****	רי אביא∩ו	RY TEST **	***
			,	2 MEMOI	XI IESI	
F9F2	6F	E2	MEMTST	OT D	C	CLEAD DYME ON CMACK
				ULIK		CLEAR BYTE ON STACK
F9F4	6F	E.2	пшптот			CLEAR BYTE ON STACK CLEAR ANOTHER BYTE
F9F4 F9F6		E2 032B	HEMIGI	CLR	,-S	CLEAR ANOTHER BYTE
F9F6	17	032B	ивитот	CLR LBSR	,-S IN2ADR	CLEAR ANOTHER BYTE GET BEGIN(Y) & END(X) ADDR. LIMITS
F9F6 F9F9	17 34	032B 30	MEMISI	CLR LBSR PSHS	,-S IN2ADR X,Y	CLEAR ANOTHER BYTE GET BEGIN(Y) & END(X) ADDR. LIMITS SAVE ADDRESSES ON STACK
F9F6 F9F9 F9FB	17 34 29	032B 30 7B	HERISI	CLR LBSR PSHS BVS	,-S IN2ADR X,Y ADJSK6	CLEAR ANOTHER BYTE GET BEGIN(Y) & END(X) ADDR. LIMITS SAVE ADDRESSES ON STACK EXIT IF NOT VALID HEX
F9F6 F9F9 F9FB F9FD	17 34 29 AC	032B 30 7B 62	HERISI	CLR LBSR PSHS BVS CMPX	,-S IN2ADR X,Y ADJSK6 2,S	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.
F9F6 F9F9 F9FB F9FD F9FF	17 34 29 AC 25	032B 30 7B 62 77	MMIGI	CLR LBSR PSHS BVS CMPX BCS	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.
F9F6 F9F9 F9FB F9FD F9FF FA01	17 34 29 AC 25 17	032B 30 7B 62 77 03E8		CLR LBSR PSHS BVS CMPX BCS LBSR	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE
F9F6 F9F9 F9FD F9FF FA01 FA04	17 34 29 AC 25 17	032B 30 7B 62 77 03E8 20	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06	17 34 29 AC 25 17 1F E3	032B 30 7B 62 77 03E8 20 64		CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08	17 34 29 AC 25 17 1F E3 34	032B 30 7B 62 77 03E8 20 64		CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.
F9F6 F9F9 F9FD F9FF FA01 FA04 FA06 FA08	17 34 29 AC 25 17 1F E3 34 AB	032B 30 7B 62 77 03E8 20 64 04 E0		CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR
F9F6 F9F9 F9FD F9FF FA01 FA06 FA08 FA0A FA0C	17 34 29 AC 25 17 1F E3 34 AB A7	032B 30 7B 62 77 03E8 20 64 04 E0 A0		CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR
F9F6 F9F9 F9FD F9FF FA01 FA04 FA06 FA08 FA0A FA0C	17 34 29 AC 25 17 1F E3 34 AB A7 10AC	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4		CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0A FA0C FA0E	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1		CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0C FA0E FA11 FA13	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0C FA0E FA11 FA13	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20		CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDR IN 'D'-ACC.
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0E FA11 FA13 FA16 FA18	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20 64	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR ADDD	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS  PUT BEGIN ADDR IN 'D'-ACC.  ADD PASS COUNT TO ADDRESS
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0E FA11 FA13 FA16 FA18 FA18	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F E3 34	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDR IN 'D'-ACC.
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0A FA11 FA13 FA11 FA13 FA16 FA18 FA1A	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F E3 34 EB	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20 64	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR ADDD	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D 4,S	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS  PUT BEGIN ADDR IN 'D'-ACC.  ADD PASS COUNT TO ADDRESS  ADD MS BYTE TO LS BYTE OF ADDRESS
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0E FA11 FA13 FA16 FA18 FA18	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F E3 34 EB	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20 64 02	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR ADDD PSHS	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D 4,S A	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS  PUT BEGIN ADDR IN 'D'-ACC.  ADD PASS COUNT TO ADDRESS
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0A FA11 FA13 FA11 FA13 FA16 FA18 FA1A	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F E3 34 EB EB	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20 64 02 E0	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR ADDD PSHS ADDD	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D 4,S A ,S+	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS  PUT BEGIN ADDR IN 'D'-ACC.  ADD PASS COUNT TO ADDRESS  ADD MS BYTE TO LS BYTE OF ADDRESS
F9F6 F9F9 F9FB F9FF FA01 FA04 FA06 FA08 FA0A FA11 FA13 FA16 FA11 FA13 FA16 FA18 FA1A	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F E3 34 EB EB	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20 64 02 E0 A0	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR ADDD PSHS ADDD PSHS ADDD PSHS ADDD PSHS ADDD PSHS ADDB EORB	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D 4,S A ,S+ ,Y+	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS  PUT BEGIN ADDR IN 'D'-ACC.  ADD PASS COUNT TO ADDRESS  ADD MS BYTE TO LS BYTE OF ADDRESS  EX-OR THIS DATA WITH DATA IN MEMORY LOC.
F9F6 F9F9 F9FB F9FD F9FF FA01 FA06 FA08 FA0C FA0E FA11 FA13 FA16 FA18 FA16 FA18 FA1C FA1E FA1C	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F E3 34 EB E8 27 8E	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20 64 02 E0 A0 3C	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR ADDD PSHS ADDD EORB BEQ	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D 4,S A ,S+ ,Y+ GUDPAS	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS  PUT BEGIN ADDR IN 'D'-ACC.  ADD PASS COUNT TO ADDRESS  ADD MS BYTE TO LS BYTE OF ADDRESS  EX-OR THIS DATA WITH DATA IN MEMORY LOC.  IF (Z) SET, MEMORY BYTE OK
F9F6 F9F9 F9FB F9FD F9FF FA01 FA06 FA08 FA0C FA0E FA11 FA13 FA16 FA18 FA16 FA18 FA12 FA12 FA12	17 34 29 AC 25 17 1F E3 34 AB A7 10AC 25 10AE 1F E3 34 EB E8 27 8E 17	032B 30 7B 62 77 03E8 20 64 04 E0 A0 E4 F1 62 20 64 02 E0 A0 3C FE83	MEMSET	CLR LBSR PSHS BVS CMPX BCS LBSR TFR ADDD PSHS ADDA STA CMPY BCS LDY TFR ADDD PSHS ADDD ESHS ADDD ESHS ADDD ADDD ESHS ADDD ADDD ESHS ADDD ADDD ESHS ADDB EORB BEQ LDX	,-S IN2ADR X,Y ADJSK6 2,S ADJSK6 OUT1S Y,D 4,S B ,S+ ,Y+ ,S MEMSET 2,S Y,D 4,S A ,S+ ,Y+ GUDPAS #MSG5	CLEAR ANOTHER BYTE  GET BEGIN(Y) & END(X) ADDR. LIMITS  SAVE ADDRESSES ON STACK  EXIT IF NOT VALID HEX  COMPARE BEGIN TO END ADDR.  EXIT IF BEGIN > END ADDR.  OUTPUT SPACE  PUT BEGIN ADDR. IN 'D'-ACCUM.  ADD PASS COUNT TO BEGIN ADDR  ADD LS BYTE TO MS BYTE OF BEGIN ADDR  SAVE THIS DATA BYTE AT BEGIN ADDR  COMPARE END TO BEGIN ADDR  IF BEGIN LOWER, CONTINUE TO SET MEMORY  RELOAD BEGIN ADDRESS  PUT BEGIN ADDR IN 'D'-ACC.  ADD PASS COUNT TO ADDRESS  ADD MS BYTE TO LS BYTE OF ADDRESS  EX-OR THIS DATA WITH DATA IN MEMORY LOC.  IF (Z) SET, MEMORY BYTE OK  POINT TO MSG " - "

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	T <sub>1</sub> RA	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	
FA4B 8E	FE8F		LDX	#MSG7	POINT TO MSG ", BITS IN ERROR
FA4E 17	036C		LBSR	PDATA	PRINT MSG
FA51 1F	98		TFR	D 1	GET ERROR BYTE INTO A-ACC
FA51 1F	FEA6		LDX	B,A #MSG9	POINT TO MSG "76543210"
			TDCD	#MOG9	OUTPUT IN BINARY/ASCII FORMAT
FA56 17 FA59 17	033E		LBSR	BIASCI	
	0383		LBSR	INCHEK	CHECK FOR INPUT FROM KEYBOARD \$FA56
FA5C 26	1A	CIIDD A C	BNE	ADJSK6	
FA5E 10AC		GUDPAS	_	,S	COMPARE END ADDR TO BEGIN ADDR
FA61 25	B3		BCS	TEST1	GEE HEAGGH GIMBOL TE MEMORY DAGG OF
FA63 86	2B		LDA	#'+	GET "PASS" SYMBOL IF MEMORY PASS OK
	0386		LBSR	OUTCH	
FA68 17	0374		LBSR	INCHEK ADJSK6	INPUT FROM KEYBOARD?
FA6B 26	0B		BNE		IF SO, EXIT MEMORY TEST
FA6D 10AE			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 1	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	BPTEST	TEST BP TABLE FOR FREE SPACE
FA8C 35	10		PULS	X	POP BP ADDRESS FROM STACK
FA8E 27	OF		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	J -	EXITBP	RTS	,	

<sup>\*</sup> INDICATE ERROR SETTING BREAKPOINT

FA9F 17 034A FAA2 86 3F FAA4 16 0347	LDA	4'?	OUTPUT SPACE LOAD (?), INDICATE BREAKPOINT ERROR PRINT "?"
	*** "X" CLE	CAR OUTSTANDIN	NG BREAKPOINTS ***
FAA7 108E DFE3 FAAB C6 08 FAAD 8D 18 FAAF 5A FABO 26 FB FAB2 39	XBKPNT LDY LDE XBPLP BSF DEC BNE RTS	B #8 RPLSWI CB XBPLP	POINT TO BREAKPOINT TABLE LOAD BREAKPOINT COUNTER REMOVE USED ENTRY IN BP TABLE \$FAAC DECREMENT BP COUNTER END OF BREAKPOINT TABLE?
	**** SWI E	NTRY POINT *	***
FAB3 1F 43 FAB5 AE 4A FAB7 30 1F FAB9 8D 26 FABB 27 04 FABD AF 4A FABF 8D 06 FAC1 17 FDE4 FAC4 16 FD9A FAC7 AE 21 FAC9 8C DFC0 FACC 24 0A FACE A6 84 FAD0 81 3F FAD2 26 04 FAD4 A6 A4 FAD6 A7 84 FAD8 86 FF FADA A7 A0 FADC A7 A0 FADC A7 A0	LDZ LEA BSF BEC STZ BSF REGPR LBS LBF RPLSWI LDZ CME BCC LDA CME	AX -1,X BPTEST REGPR 10,U RPLSWI REGSTR A NEXTCMD 1,Y X #STACK FFSTBL A X A #\$3F FFSTBL A X A #\$5F A Y A X A #\$FF A Y+ A Y+	FIND BREAKPOINT IN BP TABLE IF FOUND, REPLACE DATA AT BP ADDR SAVE BREAKPOINT ADDR IN STACK GO REPLACE SWI WITH ORIGINAL DATA GO PRINT REGISTERS GET NEXT COMMAND LOAD BP ADDRESS FROM BP TABLE COMPARE TO TOP AVAILABLE USER MEMORY GO RESET TABLE ENTRY TO \$FF'S GET DATA FROM BP ADDRESS IS IT SWI?
FAE0 39	RTS	•	
	** SEARCH E	BREAKPOINT TAE	BLE FOR MATCH **
FAE1 108E DFE3 FAE5 C6 08 FAE7 A6 A0 FAE9 AC A1 FAEB 27 04 FAED 5A FAEE 26 F7 FAF0 39	LDE FNDBP LDA	#8 #8 ,Y+ PX ,Y++ P BPADJ PB FNDBP	LOAD BREAKPOINT COUNTER LOAD DATA BYTE COMPARE ADDRESS, IS IT SAME? IF SO, ADJUST POINTER FOR TABLE ENTRY IF NOT, DECREMENT BREAKPOINT COUNTER
FAF1 31 3D FAF3 39	BPADJ LE <i>F</i>		MOVE POINTER TO BEGIN OF BP ENTRY

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

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

<sup>\*</sup> THE FOLLOWING CODE TESTS THE STATUS OF THE

FB57 5F CLRB

<sup>\*</sup> CHANNEL CONTROL REGISTER. IF "D7" IS NOT

<sup>\*</sup> ZERO THEN IT WILL LOOP WAITING FOR "D7"

<sup>\*</sup> TO GO TO ZERO. IF AFTER 65,536 TRIES IT

<sup>\*</sup> IS STILL A ONE THE BOOT OPERATION WILL

<sup>\*</sup> BE STARTED OVER FROM THE BEGINING.

FB58 FB5A	_	04	DBOOT1	PSHS CLRB	В	\$FB55
FB5B FB5E FB60		F010 0A	DBOOT2	TST BPL DECB	CCREG DBOOT3	
FB61 FB63 FB65	26 35	F8 04		BNE PULS DECB	DBOOT2 B	
FB66 FB68 FB6A FB6C FB6F FB71 FB73	20 35 B6 85 27	F0 8A 04 F020 1C 01	DBOOT3	BNE BRA PULS LDA BITA BEQ RTS	DBOOT1 DBOOT B COMREG #\$1C DBOOT4	
FB74 FB76 FB79 FB7C FB7E FB80	F7 8E AF 1F	DE F024 C000 4A 34	DBOOT4	LDB STB LDX STX TFR RTI	#\$DE DRVREG #\$C000 10,U U,S	\$FB7B

\*\*\*\*\* 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 108	BE DFD0		LDY	#LRARAM	LOAD LRA TABLE BASE ADDRESS
FB8D E6	Aб		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

SRIIG1	Я	MP - 09	MONTTOR

# 9-5-00 TSC ASSEMBLER PAGE 13

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	110		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		•
	62			2 C	CET MC DYTE OF LOCION ADDD
FB9C A6				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 I	S NIBBI	E OF LOGIC	CAL ADDRESS
FBA4 E7	62	1 200 1	STB	2,S	SAVE AS LS 16 BITS OF PHY ADDR IN X REG
FDA4 E/	02	+ 017 057		2,5	DAVE AS LS TO BITS OF PHI ADDR IN A REG
		* ON STA			
FBA6 35	36		PULS	A,B,X,Y	POP REGS. FROM STACK
FBA8 39			RTS		
		* DELAY	T.OOP		
		ביינום	поот		
TD 7 0 2 4	0.4	DT 17	Datta	ъ	CALLE COMMENTES OF THE
FBA9 34	04	DLY	PSHS	В	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
FBB0 35	04		PULS	В	RESTORE CONTENTS OF "B"
FBB2 39	01			Ъ	REDIORE CONTENTS OF D
FBB2 39			RTS		
		J., ****	J" MINII	DISK BOOT	****
			-		****
FBB3 7D	E018	***** "U	-		****
FBB3 7D FBB6 7F			TST		
			-	Comreg	
		MINBOOT	TST CLR	Comreg Drvreg	SELECT DRIVE 0
FBB6 7F	E014	MINBOOT	TST CLR BEFORE	Comreg Drvreg	
FBB6 7F	E014	MINBOOT	TST CLR BEFORE LDB	Comreg Drvreg ISSUING RI	SELECT DRIVE 0
FBB6 7F FBB9 C6 FBBB 8E	E014 03 0000	MINBOOT	TST CLR BEFORE LDB LDX	Comreg Drvreg ISSUING RI #3 #0	SELECT DRIVE 0 ESTORE COMMAND
FBB6 7F	E014	MINBOOT	TST CLR BEFORE LDB LDX	Comreg Drvreg ISSUING RI	SELECT DRIVE 0
FBB6 7F FBB9 C6 FBBB 8E	E014 03 0000	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX	Comreg Drvreg ISSUING RI #3 #0	SELECT DRIVE 0 ESTORE COMMAND
FBB9 C6 FBBB 8E FBBE 30 FBC0 8C	03 0000 01 0000	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX	Comreg Drvreg ISSUING RI #3 #0 1,X #0	SELECT DRIVE 0 ESTORE COMMAND
FBB6 7F  FBB9 C6 FBBB 8E FBBE 30 FBC0 8C FBC3 26	E014 03 0000 01	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE	Comreg Drvreg ISSUING RI #3 #0 1,X	SELECT DRIVE 0 ESTORE COMMAND \$FBBB
FBB6 7F  FBB9 C6  FBBB 8E  FBBC 30  FBC0 8C  FBC3 26  FBC5 5A	E014 03 0000 01 0000 F9	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP	SELECT DRIVE 0 ESTORE COMMAND
FBB6 7F  FBB9 C6 FBBB 8E FBBE 30 FBC0 8C FBC3 26	03 0000 01 0000	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE	Comreg Drvreg ISSUING RI #3 #0 1,X #0	SELECT DRIVE 0 ESTORE COMMAND \$FBBB
FBB6 7F  FBB9 C6  FBB8 8E  FBBC 30  FBC0 8C  FBC3 26  FBC5 5A  FBC6 26	03 0000 01 0000 F9	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE	Comreg Drvreg ISSUING RI #3 #0 1,X #0 LOOP	SELECT DRIVE 0 ESTORE COMMAND \$FBBB \$FBC2
FBB6 7F  FBB9 C6  FBBB 8E  FBBC 30  FBC0 8C  FBC3 26  FBC5 5A	E014 03 0000 01 0000 F9	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP	SELECT DRIVE 0 ESTORE COMMAND \$FBBB
FBB6 7F  FBB9 C6  FBB8 8E  FBBC 30  FBC0 8C  FBC3 26  FBC5 5A  FBC6 26	03 0000 01 0000 F9	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE	Comreg Drvreg ISSUING RI #3 #0 1,X #0 LOOP LOOP	SELECT DRIVE 0 ESTORE COMMAND \$FBBB \$FBC2
FBB6 7F  FBB9 C6  FBBB 8E  FBBC 30  FBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBC8 B7	03 0000 01 0000 F9 F6 0F E018	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg	SELECT DRIVE 0 ESTORE COMMAND \$FBBB \$FBC2 *LOAD HEAD, VERIFY, 20msec/step
FBB6 7F  FBB9 C6  FBBB 8E  FBBC 30  FBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBC8 86  FBCA B7  FBCD 8D	03 0000 01 0000 F9 F6 0F E018 37	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg DELAY	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND
FBB6 7F  FBB9 C6  FBBB 8E  FBBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBCA B7  FBCD 8D  FBCF F6	03 0000 01 0000 F9 F6 0F E018 37 E018	MINBOOT  * DELAY	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  LOOP  #\$0F Comreg DELAY Comreg	SELECT DRIVE 0 ESTORE COMMAND \$FBBB \$FBC2 *LOAD HEAD, VERIFY, 20msec/step
FBB6 7F  FBB9 C6  FBBB 8E  FBBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBCA B7  FBCD 8D  FBCF F6  FBD2 C5	03 0000 01 0000 F9 F6 0F E018 37 E018	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg DELAY Comreg #1	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC
FBB6 7F  FBB9 C6  FBBB 8E  FBBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBCA B7  FBCD 8D  FBCF F6  FBD2 C5  FBD4 26	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BNE	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND
FBB6 7F  FBB9 C6  FBBB 8E  FBBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBCA B7  FBCD 8D  FBCF F6  FBD2 C5	03 0000 01 0000 F9 F6 0F E018 37 E018	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg DELAY Comreg #1	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC
FBB6 7F  FBB9 C6  FBBB 8E  FBBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBCA B7  FBCD 8D  FBCF F6  FBD2 C5  FBD4 26  FBD6 86	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9 01	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BITB BNE LDA	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1 #1	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC
FBB6 7F  FBB9 C6 FBBB 8E FBBE 30 FBC0 8C FBC3 26 FBC5 5A FBC6 26  FBC8 86 FBCA B7 FBCD 8D FBCF F6 FBD2 C5 FBD4 26 FBD6 86 FBD6 86 FBD8 B7	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9 01 E01A	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BITB BNE LDA STA	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1 #1 Secreg	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC  LOOP UNTIL THRU
FBB6 7F  FBB9 C6 FBBB 8E FBBE 30 FBC0 8C FBC3 26 FBC5 5A FBC6 26  FBC8 86 FBCA B7 FBCD 8D FBCF F6 FBD2 C5 FBD4 26 FBD6 86 FBD8 B7 FBDB 8D	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9 01 E01A 29	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BITB BITB BNE LDA STA BSR	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1 #1 Secreg DELAY	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC  LOOP UNTIL THRU  SET SECTOR REGISTER TO ONE
FBB6 7F  FBB9 C6 FBBB 8E FBBE 30 FBC0 8C FBC3 26 FBC5 5A FBC6 26  FBC8 86 FBCA B7 FBCD 8D FBCF F6 FBD2 C5 FBD4 26 FBD4 26 FBD6 86 FBD8 B7 FBDB 8D FBDB 8D	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9 01 E01A 29 8C	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BITB BNE LDA STA BSR LDA STA BSR LDA STA	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1 #1 Secreg DELAY #\$8C	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC  LOOP UNTIL THRU  SET SECTOR REGISTER TO ONE LOAD HEAD, DELAY 10msec,
FBB6 7F  FBB9 C6  FBBB 8E  FBBE 30  FBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBCA B7  FBCD 8D  FBCF F6  FBD2 C5  FBD4 26  FBD6 86  FBD8 B7  FBD8 B7  FBDB 8D  FBDB 8D  FBDB 8D  FBDD 86  FBDD 86  FBDD 87	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9 01 E01A 29 8C E018	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BITB BNE LDA STA BSR LDA STA BSR LDA STA BSR LDA STA	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1 #1 Secreg DELAY #\$8C Comreg	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC  LOOP UNTIL THRU  SET SECTOR REGISTER TO ONE
FBB6 7F  FBB9 C6 FBBB 8E FBBE 30 FBC0 8C FBC3 26 FBC5 5A FBC6 26  FBC8 86 FBCA B7 FBCD 8D FBCF F6 FBD2 C5 FBD4 26 FBD4 26 FBD6 86 FBD8 B7 FBDB 8D FBDB 8D	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9 01 E01A 29 8C	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BITB BNE LDA STA BSR LDA STA BSR LDA STA	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1 #1 Secreg DELAY #\$8C	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC  LOOP UNTIL THRU  SET SECTOR REGISTER TO ONE LOAD HEAD, DELAY 10msec,
FBB6 7F  FBB9 C6  FBBB 8E  FBBE 30  FBC0 8C  FBC3 26  FBC5 5A  FBC6 26  FBC8 86  FBCA B7  FBCD 8D  FBCF F6  FBD2 C5  FBD4 26  FBD6 86  FBD8 B7  FBD8 B7  FBDB 8D  FBDB 8D  FBDB 8D  FBDD 86  FBDD 86  FBDD 87	E014  03 0000 01 0000 F9  F6  0F E018 37 E018 01 F9 01 E01A 29 8C E018	MINBOOT  * DELAY  LOOP	TST CLR BEFORE LDB LDX LEAX CMPX BNE DECB BNE LDA STA BSR LDB BITB BITB BNE LDA STA BSR LDA STA BSR LDA STA BSR LDA STA	Comreg Drvreg  ISSUING RI #3 #0 1,X #0 LOOP  LOOP  LOOP  #\$0F Comreg DELAY Comreg #1 LOOP1 #1 Secreg DELAY #\$8C Comreg	SELECT DRIVE 0 ESTORE COMMAND  \$FBBB  \$FBC2  *LOAD HEAD, VERIFY, 20msec/step ISSUE RESTORE COMMAND  \$FBCC  LOOP UNTIL THRU  SET SECTOR REGISTER TO ONE LOAD HEAD, DELAY 10msec,

SRIIG18	MD_09	MONTTOR
סודוח	パローいっ	7 1910 1131 1 1 1 1 1 1 1 1 1

				_	
FBE7 20	09		BRA	LOOP3	
FBE9 C5	02	LOOP2	BITB	#2	\$FBE6 DRO?
FBEB 27	05	LOOPZ	BEQ	LOOP3	Promo Dry:
FBED B6	E01B		LDA	Datreg	
FBF0 A7	80		STA	,X+	
1210 117			2111	,	
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			
		22211			
FC06 C6	20	DELAY	LDB	#\$20	
FC08 5A		LOOP5	DECB		
FC09 26	FD		BNE	LOOP5	
FC0B 39			RTS		
		***** "	T    T \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	MIKBUG TA	DE ****
			L LOAD	MIKBUG IA	VAE
FC0C 86	11	LOAD	LDA	#\$11	LOAD 'DC1' CASS. READ ON CODE
FC0C 86 FC0E 17	11 01DD	LOAD	LDA LBSR	#\$11 OUTCH	LOAD 'DC1' CASS. READ ON CODE OUTPUT IT TO TERMINAL PORT
		LOAD		•	
FC0E 17	01DD	LOAD	LBSR	OUTCH	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO
FC0E 17 FC11 7F	01DD DFE2		LBSR CLR	OUTCH ECHO	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26	01DD DFE2 01AD 53 F9	LOAD1	LBSR CLR LBSR	OUTCH ECHO ECHON #'S LOAD1	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17	01DD DFE2 01AD 53 F9 01A6	LOAD1	LBSR CLR LBSR CMPA BNE LBSR	OUTCH ECHO ECHON #'S LOAD1 ECHON	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81	01DD DFE2 01AD 53 F9 01A6 39	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR. IS IT A "9", END OF FILE CHAR?
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27	01DD DFE2 01AD 53 F9 01A6 39	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81	01DD DFE2 01AD 53 F9 01A6 39 3D 31	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR?
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26	01DD DFE2 01AD 53 F9 01A6 39 3D 31	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR? IF NOT, LOOK FOR START CHAR.
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR CMPA	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17 FC30 29	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17 FC30 29 FC32 34	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS PSHS	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17 FC30 29 FC32 34 FC34 E6	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21 10 E0	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS PSHS LDB	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X ,S+	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK LOAD MSB OF ADDR AS CHECKSUM BYTE
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17 FC30 29 FC32 34 FC34 E6 FC36 EB	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21 10 E0	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS LBSR BVS LDB ADDB	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X ,S+ ,S+ ,S	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK LOAD MSB OF ADDR AS CHECKSUM BYTE ADD LSB OF ADDR TO CHECKSUM
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC28 29 FC2D 17 FC30 29 FC32 34 FC34 E6 FC36 EB FC38 EB FC3A 6A FC3C 6A	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21 10 E0 E0 E4	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS LBSR BVS LDB ADDB ADDB	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X ,S+ ,S+	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK LOAD MSB OF ADDR AS CHECKSUM ADD BYTE COUNT BYTE TO CHECKSUM
FC0E 17 FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC28 29 FC2D 17 FC30 29 FC31 34 FC34 E6 FC36 EB FC38 EB FC3A 6A FC3C 6A FC3E 34	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21 10 E0 E0 E4 E4 E4 E4	LOAD1	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS LBSR BVS LDB ADDB ADDB DEC	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X ,S+ ,S+ ,S	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK LOAD MSB OF ADDR AS CHECKSUM BYTE ADD LSB OF ADDR TO CHECKSUM ADD BYTE COUNT BYTE TO CHECKSUM \$FC37 DECREMENT BYTE COUNT 2 TO BYPASS
FC0E 17 FC11 7F FC11 7F FC14 17 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17 FC30 29 FC32 34 FC34 E6 FC36 EB FC36 EB FC38 EB FC38 6A FC3C 6A FC3C 34 FC40 17	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21 10 E0 E0 E4 E4 E4 O4 00FD	LOAD1 LOAD2	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS PSHS LDB ADDB ADDB ADDB ADDB ADDB ADDB ADDB	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X ,S+ ,S+ ,S ,S ,S B BYTE	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK LOAD MSB OF ADDR AS CHECKSUM BYTE ADD LSB OF ADDR TO CHECKSUM ADD BYTE COUNT BYTE TO CHECKSUM \$FC37 DECREMENT BYTE COUNT 2 TO BYPASS ADDRESS BYTES. PUSH CHECKSUM ON STACK INPUT DATA BYTE (2 HEX CHAR)
FC0E 17 FC11 7F FC11 7F FC14 17 FC17 81 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17 FC30 29 FC32 34 FC34 E6 FC36 EB FC36 EB FC37 6A	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21 10 E0 E0 E4 E4 E4 04 00FD 04	LOAD1 LOAD2	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS LBSR BVS LBSR DEC DEC PSHS LBSR PSHS	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X ,S+ ,S+ ,S ,S ,S B BYTE B	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK LOAD MSB OF ADDR AS CHECKSUM BYTE ADD LSB OF ADDR TO CHECKSUM ADD BYTE COUNT BYTE TO CHECKSUM \$FC37 DECREMENT BYTE COUNT 2 TO BYPASS ADDRESS BYTES. PUSH CHECKSUM ON STACK INPUT DATA BYTE (2 HEX CHAR) POP CHECKSUM FROM STACK
FC0E 17 FC11 7F FC11 7F FC14 17 FC19 26 FC1B 17 FC1E 81 FC20 27 FC22 81 FC24 26 FC26 17 FC29 34 FC2B 29 FC2D 17 FC30 29 FC32 34 FC34 E6 FC36 EB FC36 EB FC38 EB FC38 6A FC3C 6A FC3C 34 FC40 17	01DD DFE2 01AD 53 F9 01A6 39 3D 31 F1 0117 02 26 00FF 21 10 E0 E0 E4 E4 E4 O4 00FD	LOAD1 LOAD2	LBSR CLR LBSR CMPA BNE LBSR CMPA BEQ CMPA BNE LBSR PSHS BVS LBSR BVS PSHS LDB ADDB ADDB ADDB ADDB ADDB ADDB ADDB	OUTCH ECHO ECHON #'S LOAD1 ECHON #'9 LOAD21 #'1 LOAD2 BYTE A LODERR IN1ADR LODERR X ,S+ ,S+ ,S ,S ,S B BYTE	OUTPUT IT TO TERMINAL PORT TURN OFF ECHO FLAG INPUT 8 BIT BYTE WITH NO ECHO IS IT AN "S", START CHARACTER ? IF NOT, DISCARD AND GET NEXT CHAR.  IS IT A "9", END OF FILE CHAR ? IF SO, EXIT LOAD IS IT A "1", FILE LOAD CHAR ? IF NOT, LOOK FOR START CHAR. INPUT BYTE COUNT PUSH COUNT ON STACK (V) C-CODE SET, ILLEGAL HEX INPUT LOAD ADDRESS (V) C-CODE SET, ADDR NOT HEX PUSH ADDR ON STACK LOAD MSB OF ADDR AS CHECKSUM BYTE ADD LSB OF ADDR TO CHECKSUM ADD BYTE COUNT BYTE TO CHECKSUM \$FC37 DECREMENT BYTE COUNT 2 TO BYPASS ADDRESS BYTES. PUSH CHECKSUM ON STACK INPUT DATA BYTE (2 HEX CHAR)

FC49 EB	ΕO		ADDB	,S+	ADD DATA TO CHECKSUM, AUTO INC STACK
FC4B 6A	E4		DEC		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" PUNCI	H MIKBUG T	APE ****
FC67 6F	E2	DIINCH	CT.R	-S	CLEAR RESERVED BYTE ON STACK
FC69 17	00B8	1 011011	LBSR		GET BEGIN AND END ADDRESS
FC6C 34	30		PSHS		SAVE ADDRESSES ON STACK
FC6E 29	4A		BVS		(V) C-CODE SET, EXIT PUNCH
FC70 AC	62		CMPX	2,S	
FC72 25	46		BCS	PUNEXT	
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		PUNCH2			LOAD END ADDR IN D-ACC
FC7F A3	62	1 0110112	SUBD	2,S	
FC81 27	06		BEQ	PUNCH3	
FC83 1083			CMPD	#\$20	LESS THAN 32 BYTES?
FC87 23	02		BLS	PUNCH4	PUNCH THAT MANY BYTES
FC89 C6		PUNCH3		#\$20	LOAD BYTE COUNT OF 32.
FC8B E7		PUNCH4		4,S	STORE ON STACK AS BYTE COUNT
FC8D 8E	FEEB	2 021011 1	LDX		POINT TO MSG "S1"
FC90 17	011A		LBSR		PRINT MSG
FC93 CB	03		ADDB		ADD 3 BYTES TO BYTE COUNT
FC95 1F	98		TFR	**	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.
<del></del>	-		•	<del></del>	

FCBA 86 FCBC 17 FCBF 32 FCC1 39	14 012F 65	PUNEXT	LDA LBSR LEAS RTS	#\$14 OUTCH 5,S	LOAD 'DC4' PUNCH OFF CODE OUTPUT IT READJUST STACK POINTER
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	
FCEO 16	009E	DDMIII	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 FCEB 16	44 000B		LDX	4,U	\$FCE6
FCEE 8E	008B FEC0	PRTIY	LBRA LDX	OUT4H	POINT TO MSG "IY="
FCF1 17	00C9	PRILI	LBSR	#MSG13 PDATA	PRINT MSG
FCF4 AE	46		LDX	6,U	PRINI MSG
FCF6 16	0080		LBRA	OUT4H	
FCF9 8E	FEB4	PRTPC	LDX	#MSG11	POINT TO MSG "PC="
FCFC 17	00BE	IRIIC	LBSR	PDATA	PRINT MSG
FCFF AE	4A		LDX	10,U	TICINI MOO
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
FDOD 8E	FED7	PRTB	LDX	#MSG17	POINT TO MSG "B="
FD10 17	00AA		LBSR	PDATA	PRINT MSG
FD13 A6	42		LDA	2,U	
FD15 20	бA		BRA	OUT2H	OUTPUT HEX BYTE AS ASCII
FD17 8E	FEDC	PRTCC	LDX	#MSG18	POINT TO MSG "CC:"
FD1A 17	0A0		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
- $\mbox{\scriptsize \star}$  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"

-	
	· /

SBUG18	MP-09	MONITOR			9-5-00	TSC ASSEMBLER PAGE 17
	86 17	2D 00BF		LDA LBSR		PRINT " - "
			* OPERA	TOR TO	INPUT ONE	LOOPS WAITING FOR THE VALID HEX ADDRESS. THE THE "X" REGISTER.
FD35 FD37 FD39 FD3B	29 1F 8D 29 34 A7 35	09 3D 10 61		BVS TFR BSR	NOTHEX D,X BYTE NOTHEX X 1,S	INPUT BYTE (2 HEX CHAR) EXIT IF NOT VALID HEX INPUT BYTE (2 HEX CHAR)
			**** II	NPUT BY	re (2 HEX	CHAR.) ****
FD42 FD44 FD45	29 48 48	11 32	BYTE			GET HEX LEFT EXIT IF NOT VALID HEX
FD46 FD47	-	89		ASLA ASLA TER		SHIFT INTO LEFT NIBBLE PUT HEXL IN "B"
FD4A FD4C FD4E	8D 29 34 AB	07 28		BSR BVS PSHS	INHEX NOTHEX B	GET HEX RIGHT EXIT IF NOT VALID HEX PUSH HEXL ON STACK ADD HEXL TO HEXR AND ADJ. STK RETURN WITH HEX L&R IN "A"
FD55	81 25 81 22 80	30		CMPA	#'0	<pre>INPUT ASCII CHAR. IS IT &gt; OR = "0" ? IF LESS IT AIN'T HEX IS IT &lt; OR = "9" ? IF &gt; MAYBE IT'S ALPHA ASCII ADJ. NUMERIC</pre>
FD60 FD62 FD64 FD66 FD68 FD6A	25 81 22 80	41 12 46 03 37	INHEXA	CMPA BCS CMPA BHI SUBA RTS	#'A NOTHEX #'F INHEXL #\$37	IS IT > OR = "A"  IF LESS IT AIN'T HEX  IS IT < OR = "F" ?  IF > IT AIN'T HEX  ASCII ADJ. ALPHA
FD6B FD6D FD6F FD71 FD73	25 81 22	61 07 66 03 57	INHEXL	CMPA BCS CMPA BHI SUBA	#'a NOTHEX #'f NOTHEX #\$57	IS IT > OR = "a" IF LESS IT AIN'T HEX IS IT < "f" IF > IT AIN'T HEX ADJUST TO LOWER CASE

FD75 39			RTS		
FD76 1A FD78 39	02	NOTHEX	ORCC RTS	#2	SET (V) FLAG IN C-CODES REGISTER
FD79 34 FD7B 35 FD7D 8D FD7F 35	10 02 02 02 02 FD81	OUT4H OUTHL OUT2H	PSHS PULS BSR PULS EQU PSHS	X A OUTHL A *	PUSH X-REG. ON THE STACK POP MS BYTE OF X-REG INTO A-ACC. OUTPUT HEX LEFT POP LS BYTE OF X-REG INTO A-ACC. SAVE IT BACK ON STACK
FD81 34 FD83 44 FD84 44 FD85 44 FD86 44	02	OUIZH	LSRA LSRA LSRA LSRA	A	CONVERT UPPER HEX NIBBLE TO ASCII
FD87 8D FD89 35 FD8B 84 FD8D 8B	04 02 0F 30	OUTHR XASCII	BSR PULS ANDA ADDA	XASCII A #\$0F #\$30	PRINT HEX NIBBLE AS ASCII CONVERT LOWER HEX NIBBLE TO ASCII STRIP LEFT NIBBLE ASCII ADJ
FD8F 81 FD91 2F FD93 8B FD95 20	39 02 07 57	OUTC	CMPA BLE ADDA BRA	#\$39 OUTC #7 OUTCH	IS IT < OR = "9" ? IF LESS, OUTPUT IT IF > MAKE ASCII LETTER OUTPUT CHAR
		* OUTPU * BINAR * IS DO * LETTE * THE A * ARE O	TS A BY Y FORMA NE BY S RS FOR SCII EN BTAINED	HANCEMENT FROM THE	NCED ANCEMENT G ASCII N THE BYTE. LETTERS
FD97 34 FD99 C6 FD9B A6 FD9D 68	02 08 80 E4	BIASCI OUTBA	PSHS LDB LDA ASL	A #8 ,X+ ,S	SAVE "A" ON STACK PRESET LOOP# TO BITS PER BYTE GET LETTER FROM STRING TEST BYTE FOR "1" IN B7
FD9F 25 FDA1 86 FDA3 8D FDA5 8D FDA7 5A FDA8 26 FDAA 35 FDAC 39	02 2D 49 45 F1 02	PRTBA	BCS LDA BSR BSR DECB BNE PULS RTS	PRTBA #'- OUTCH OUT1S OUTBA A	IF ONE PRINT LETTER IF ZERO PRINT "-" PRINT IT PRINT SPACE SUB 1 FROM #BITS YET TO PRINT
		* PRINT	STRING	PRECEEDED	BY A CR & LF.
FDAD 8D FDAF 20	02 0C	PSTRNG	BSR BRA	PCRLF PDATA	PRINT CR/LF PRINT STRING POINTED TO BY IX

<sup>\*</sup> PCRLF

FDB1 34 FDB3 8E FDB6 8D FDB8 35 FDBA 39 FDBB 8D	10 FE75 05 10	PCRLF	PSHS LDX BSR PULS RTS BSR	X #MSG2+1 PDATA X OUTCH	SAVE IX POINT TO MSG CR/LF + 3 NULS PRINT MSG RESTORE IX
		* PDATA			
FDBD A6 FDBF 81 FDC1 26 FDC3 39	80 04 F8	PDATA	LDA CMPA BNE RTS	,X+ #4 PRINT	GET 1st CHAR. TO PRINT IS IT EOT? IF NOT EOT PRINT IT
FDC4 7D FDC7 27	DFE2 06	ECHON * INCHE	TST BEQ	ECHO INCH	IS ECHO REQUIRED ? ECHO NOT REQ. IF CLEAR

- \* ---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 C	CHAR	FROM	I TERM	IINAL
FDCB	84	7F		ANDA	#\$7F	STRIE	PAR	ITY	FROM	CHAR.
FDCD	20	1F		BRA	OUTCH	ECHO	CHAR	TO	TERMI	NAL

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

- \* OUTCH
- \* OUTPUT CHARACTER TO TERMINAL.
- \* THE CHAR. TO BE OUTPUT IS
- \* PASSED IN THE A REGISTER.
- \* ALL REGISTERS ARE PRESERVED.

FDEE 34 FDF0 BE FDF3 A6 FDF5 85 FDF7 27 FDF9 35 FDFB A7 FDFD 35 FDFF 39	12 DFE0 84 02 FA 02 01	OUTCH	PSHS LDX LDA BITA BEQ PULS STA PULS RTS	A,X CPORT ,X #2 FETSTA A 1,X	SAVE A ACCUM AND IX GET ADDR. OF TERMINAL FETCH PORT STATUS TEST TDRE, OK TO XMIT? IF NOT LOOP UNTIL RDY GET CHAR. FOR XMIT XMIT CHAR. RESTORE IX
FE00 BE FE03 86 FE05 A7 FE07 86 FE09 A7 FE0B 6D FE0D 86 FE0F B7 FE12 39	DFE0 03 84 11 84 01 FF DFE2	ACINIZ	LDX LDA STA LDA STA TST LDA STA RTS	CPORT #3 ,X #\$11 ,X 1,X #\$FF ECHO	POINT TO CONTROL PORT ADDRESS RESET ACIA PORT CODE STORE IN CONTROL REGISTER SET 8 DATA, 2 STOP AN 0 PARITY STORE IN CONTROL REGISTER ANYTHING IN DATA REGISTER? TURN ON ECHO FLAG

#### \* MONITOR KEYBOARD COMMAND JUMP TABLE

		FE13	JMPTAB	EQU	*				
FE13	01			FCB	1	11	^A	II .	\$F91D
FE14	F923			FDB	ALTRA				
FE16	02			FCB	2	"	^B	II .	\$F90F
FE17	F915			FDB	ALTRB				
FE19	03			FCB	3	"	^C	II .	\$F92B
FE1A	F931			FDB	ALTRCC				
FE1C	04			FCB	4	11	^D	"	\$F901
FE1D	F907			FDB	ALTRDP				

SBUG18 MP-09 M	ONTIOR
----------------	--------

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	<del></del>	FCC	'B'		
	FA7B	FDB	BRKPNT	*\$FA78	
FE2E		FCC	'D'		
	FAF4	FDB	DBOOT	*\$FAF1	
FE31	= -	FCC	'E'		
	F996	FDB	MEMDUMP	*\$F990	
FE34		FCC	' G '		
	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	\$FFFF	SVC-VO
FE5D	FFFF		FDB	\$FFFF	SVC-VL

# \* PRINTABLE MESSAGE STRINGS

FE5F FE63		00	0D 00	MSG1	FCB	\$0,\$0,\$0,\$D,\$A,\$0,\$0,\$0 * 0, CR/LF, 0
FE67					FCC	'S-BUG 1.8 - '
FE6B						
FE6F						
FE73					FCB	4
FE74	0D	0A	0.0	MSG2	FCB	'K,\$D,\$A,\$0,\$0,\$0,4 K, * CR/LF + 3 NULS
FE78				11002	102	21,42,411,40,40,1111, 011,21 1010
FE7B		0 -		MSG3	FCC	'>'
FE7C					FCB	4
FE7D	48	41	54	MSG4	FCC	- 'WHAT?'
FE81						
FE82					FCB	4
FE83	2D	20		MSG5	FCC	1 _ 1
FE86					FCB	4
FE87	2.0	50	41	MSG6	FCC	', PASS '
FE8B				11500	100	, 11100
FE8E	55	20			FCB	4
FE8F	20	42	49	MSG7	FCC	', BITS IN ERROR: '
FE93				11507	100	, DIIS IN BRICOR
FE97						
FE9B						
FE9F	11	22	JA			
FEA0					FCB	4
FEA1	מצ	3 F.	20	MSG8	FCC	' => '
FEA5	שנ	20	20	11000	FCB	4
FEA6	36	35	34	MSG9	FCC	'76543210'
FEAA				141000	rcc	70313210
FEAE				MSG10	FCC	' SP='
FEB2	20	55	30	PIDGIG	100	51 -
FEB3					FCB	4
FEB4	20	50	43	MSG11	FCC	' PC='
FEB8	20	50	13	HOULI	100	
FEB9					FCB	4
FEBA	20	55	53	MSG12	FCC	' US='
FEBE	20	55	33	MOGIZ	rcc	05-
FEBF					FCB	4
FEC0	20	49	59	MSG13	FCC	' IY='
FEC4	20	17	5,5	110013	100	
FEC5					FCB	4
FEC6	20	49	5.8	MSG14	FCC	' IX='
FECA	20	17	30	INDOT I	100	111
FECB					FCB	4
FECC	2.0	44	50	MSG15	FCC	' DP='
FED0	20		30	115015	100	
FED1					FCB	4
FED2	20	41	3D	MSG16	FCC	' A='
FED6					FCB	4
FED7	20	42	3D	MSG17	FCC	' B='
FEDB	_ •				FCB	4
FEDC	20	43	43	MSG18	FCC	' CC: '
FEE0						
5	 -					

CRITC1 8	MD = 00	MONITOR	
SDUGTO	MP-US	MONTIOK	

9-5-00 TSC ASSEMBLER PAGE 23

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

FEEE FF FF FF FF FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF
FEF2 FF FF FF FF
FEFA FF FF FF
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF
FCB \$FF,\$FF,\$FF,\$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 OF 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
- $^{\star}$  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
FFOE	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 OUTPUTING 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	В7	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 -- -- --
- \* ....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
- \* OF OE OD OC OB OA O9 O8 O6 O5 OO OO O4 O3 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 -- --

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 OC		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	
FF72 88 OF		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
	FINTAB		#\$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 FF82 FF84	E6 26	0C A6 05	FINDC	BNE	#\$0C A,Y FOUNDC	PRESET NUMBER HEX "C" GET ENTRY FROM LRARAM TABLE 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	Aб	FOUNDC	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 FF90 TABLE.		21		CLRA TFR	Y,X	START AT ZERO START POINTER "X" START OF "LRARAM"
FF92 FF94 FF96 FF98 FF9A	27 6F E7	A6 04 A6 80	COMPRS	LDB BEQ CLR STB INCA	A,Y PNTNXT A,Y ,X+	GET ENTRY FROM "LRARAM" TABLE IF IT'S ZERO SKIP ELSE ERASE FROM TABLE AND ENTER ABOVE LAST ENTRY- BUMP GET OFFSET TO NEXT ENTRY
FF9B FF9D	-	0C F3		CMPA BLT	#\$0C COMPRS	LAST ENTRY YET ?
				RS FROM	THE LRARAI	NSFER THE TRANSLATION M TABLE TO IC11 ON
FF9F FFA2 FFA4 FFA6 FFA9 FFAB FFAC FFAC	C6 A6 A7 5A 26 53 F7	FFF0 10 A0 80 F9 DFE2 F862	XFERTF	LDX LDB LDA STA DECB BNE COMB STB LBRA	#IC11 #\$10 ,Y+ ,X+ FETCH ECHO MONITOR	POINT TO DAT RAM IC11 GET NO. OF BYTES TO MOVE GET BYTE AND POINT TO NEXT POKE XLATION FACTOR IN IC11 SUB 1 FROM BYTES TO MOVE CONTINUE UNTIL 16 MOVED SET "B" NON-ZERO TURN ON ECHO FLAG INITIALIZATION IS COMPLETE
FFB2 FFB6 FFBA FFBE FFC2	6E 6E 6E	9F DFC0 9F DFC4 9F DFC6 9F DFC8 9F DFCA	V2 V3	JMP JMP JMP JMP JMP	[STACK] [SWI2] [FIRQ] [IRQ] [SWI]	
FFC6 FFC8 FFCC FFCE FFCF FFD0 FFD1 FFD4 FFD7 FFD9 FFD8 FFDE FFD8	AE E6 AF 4F 58 49 BE 8C 27 30 BC 22 34	43 4A 80 4A DFCC FFFF 0F 8B DFCE 08 10 C4	SWI3E	TFR LDX LDB STX CLRA ASLB ROLA LDX CMPX BEQ LEAX CMPX BHI PSHS LDD	S,U 10,U ,X+ 10,U SVCVO #\$FFFF SWI3Z D,X SVCVL SWI3Z X	*\$FFC8

aprra1	$\circ$	14D 00	MONTEMOR
$SBU(\pm 1)$	×	MP-119	MONTTOR

9-5-00 TSC ASSEMBLER PAGE 27

FFE4	AE	44		LDX	<b>4</b> ,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	ALTBD	F922	ALTCCD	F940	ALTDPD	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	BPTEST	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		EDUMP	F9BD	ELOOP	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
FOUNDC	FF8B	GETSTA	FDD4	GO	F8A5	GUDPAS	FA5E	IC11	FFF0
IN1ADR	FD2F	IN2ADR	FD24	INCH	FDCF	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	DFD0	MDUMP1	F99B	MEMC2	F948	MEMCHG	F941
MEMDUM	F996	MEMSET	FA04	MEMTST	F9F2	MINBOO	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	RELPAS	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						