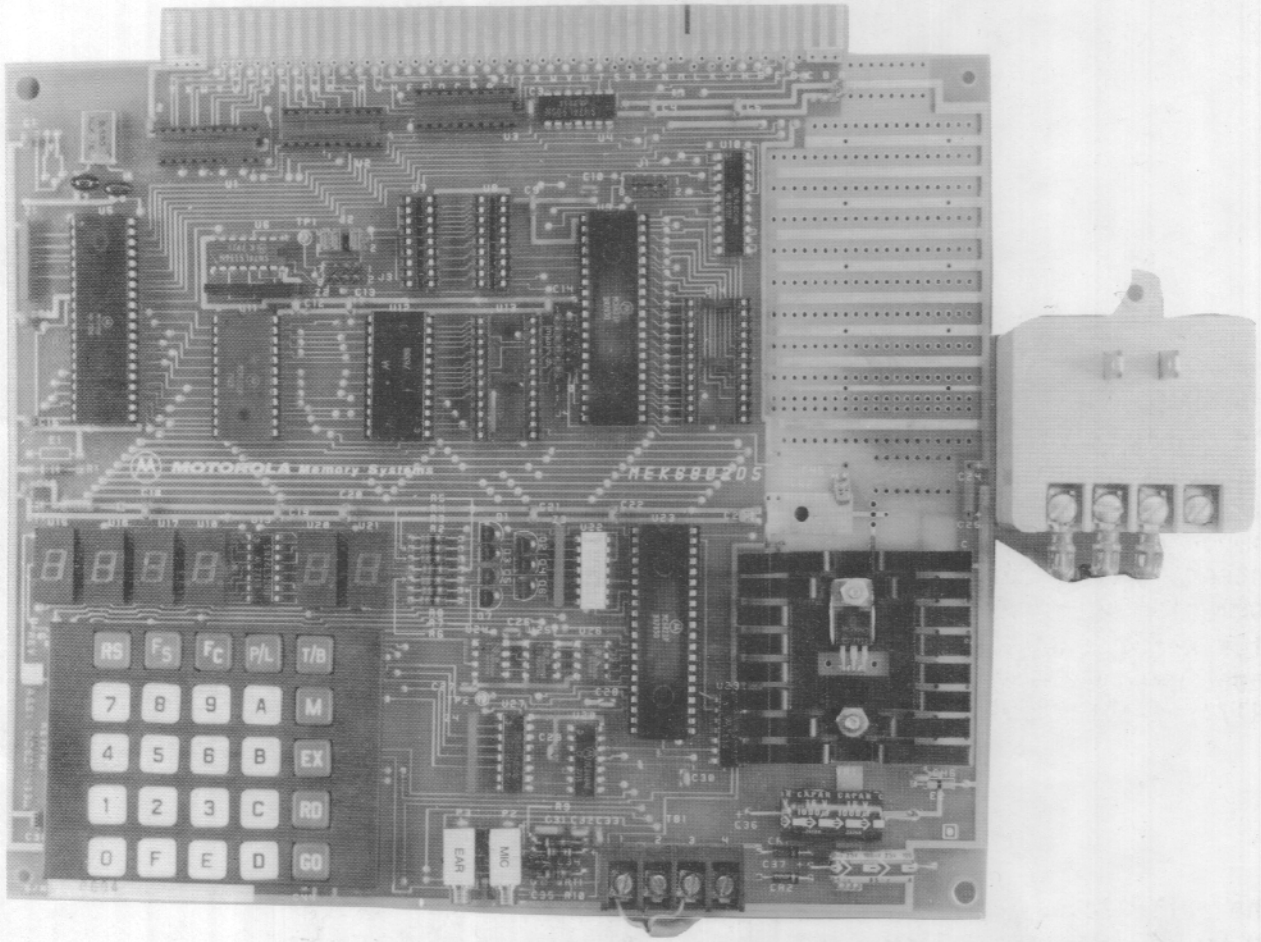
MOTOROLA INC.

MEK6802D5

MICROCOMPUTER EVALUATION BOARD USER'S MANUAL

Memory Systems



MEK6802D5 MICROCOMPUTER EVALUATION BOARD

MEK6802D5

MICROCOMPUTER EVALUATION BOARD USER'S MANUAL

PREFACE

The MEK6802D5 Microcomputer Evaluation Board provides the necessary hardware and firmware for a computer system based on the Motorola MC6802 Microprocessor. The system forms an evaluation tool to faciliate the application of Motorola microprocessors and associated components. The system is supplied with all the components required to operate, including a 115 Vac plug-in transformer and integral power supply and can be used in various configurations. With the information supplied in this manual, the user can prototype dedicated systems plus write and evaluate software programs in machine language. Provisions are made for extensive system expansion using a video terminal or keyboard and CRT/TV display unit.

Second Edition

Printed in USA

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

MEK6802D5 SPECIFICATIONS

System Components:	MC6802 Microprocessor D5BUG Firmware (2K) Six 7-segment displays 25-key scanned keypad 300 Baud cassette interface Uncommitted user PIA Wire Wrap area On-board power supply
Supply Voltage:	120 Vac 60 Hz to external transformer, 18 Vac input to board, +5 Vdc regulated output on board.
Operating Temperature:	0 °C to 55 °C (32 °F to 131 °F)
Relative Humidity:	To 80% without condensation
Dimensions:	A two-sided printed circuit board measuring 9.75 in (247.65 mm) wide by 8.75 in (222.25 mm) high by 0.062 in (1.57 mm) thick.
Options/Expansion:	Two (2) MCM2114 1K x 4 RAMs One (1) ROM Bus buffers MEK6800AB Motherboard MEK68R2 CRT display board ASCII Keyboard CRT/TV Monitor MEK68MM RAM board MEK68RR RAM/ROM board MEK68IO Input/Output Interface MEK68EP EPROM Programmer
Software Features:	Memory Change/Display Advance to next ADDR Back-up to previous ADDR Calculate offsets Register Change/Display Breakpoint Editor Cassette punch/load/verify Trace single instruction Go To/Continue user program User function capability Escape from all functions Save user registers

CHAPTER 1

GENERAL INFORMATION

1.1 Description

The MEK6802D5 Microcomputer Evaluation Board is a basic, low-cost system to evaluate the capability of the MC6800 Microcomputer Family of Components. It is useful as a tool to learn programming techniques, and to develop custom applications of microprocessors.

The system consists of an MC6802 microprocessor, D5BUG firmware in an MC68A316E ROM, (6) 7segment multiplexed displays, a 25-key scanned keypad, 300 Baud K.C. standard cassette interface and an on-board power supply capable of driving the minimum system.

A wire wrap area and an uncommitted user PIA are provided to allow the user to create desired options. In addition, provisions are made to add (2) MCM2114 1K x 4 RAMs, a 24-pin ROM or EPROM, and Bus buffers to an 86-pin connector.

All components, including the AC transformer required for the minimum system are supplied with the MEK6802D5.

The functions of the basic and optional components of the system are described in the following sections.

1.2 Components

1.2.1 MC6802 - MPU (U5)

The microprocessor generates the system clock, executes the programs residing in D5BUG, and also executes user programs. There are 128 bytes of RAM (reserved for the user in the D5) included in the MC6802 (addresses $ØØØØ to $007F).

1.2.2 MC68A316E - D5BUG (U12)

This factory programmed ROM (Read Only Memory), also known as a Monitor, contains the operating system firmware programs which control the D5 system.

1.2.3 MC6821 - PIA (U23)

This PIA (Peripheral Interface Adapter) is the system PIA and interfaces the keypad and display to the MPU. This unit can transmit data (in

parallel form) in either direction and provides a flow path for data in or out of the system. It also interfaces the cassette interface circuitry and trace-one instruction timer to the MPU (U5).

1.2.4 MC6821 - PIA (U9)

This PIA interfaces the D5 to external devices or systems for the user to use in producing custom designs. It provides two 8-bit bi-directional data ports and four control lines connected to a 24-pin Dip socket located adjacent to U9.

1.2.5 MC6810 - Static RAM (U11)

This is a static RAM (Random Access Memory) used as a scratch pad for the MPU (U5). It provides 128 bytes of Flag, Data, and Stack area for the D5BUG operating system (24 bytes are allowed for use as a user's stack area).

1.2.6 LED Display

The display system consists of LEDs, U15, U16, U17, U18, U20, U21; anode drivers Q1 through Q7; and cathode drivers U24 through U26. The system PIA (U23) uses seven lines to drive the anode drivers and six lines to drive the cathode drivers.

1.2.7 Keyboard

Depressing a key generates an interrupt to the processor. The system firmware causes the processor to search for the key closure and act upon it.

1.3 Optional Devices

1.3.1 MC2114 RAM Memories (U7, U8)

There are two 18 pin sockets for installing MCM2114 (1K x 4 bit) static RAM memories.

1.3.2 Bus Drivers (U1, U2, U3, U10)

These units interface the processor to an 86 pin connector to expand the MEK6802D5 system.

CHAPTER 2

INSTALLATION

2.1 Preparation For Use

2.1.1 Handling

> CAUTION
>
> Turn power off before removing or installing components
>
> SPECIAL HANDLING REQUIRED
>
> Store board in special conductive bag. Do not touch circuitry while handling. Static discharge may damage components.

Generally, it is required that the board and repairing personnel all be at the same (ground) potential. If the unit is on a bench or is not installed, a common ground connection between installing and/or repairing personnel and the unit should be made.

2.2 Use

Place the D5 board on its stand offs or otherwise insulated from a bench or table top and plug the transformer into a wall outlet of the correct AC voltage. Refer to the operation section of this manual for more information.

2.3 Precautions

Because MOS devices have extremely high input resistance, they are susceptible to damage when exposed to high static electrical charges.

To avoid possible damage to the devices during handling, testing, or actual operation, use the following procedures:

a) The leads of devices should be in contact with a conductive material, except when being tested or in actual operation, to avoid build-up of static charge.

2.3 Precautions (cont'd)

b) Soldering-iron tips, metal parts of fixtures and tools, and handling facilities should be grounded.

c) Devices should not be inserted into or removed from circuits with the power on because transient voltages may cause permanent damage.

d) Signals should not be applied to the inputs while the device power supply is off.

e) All unused input leads must be connected to either V_{SS} (ground) or V_{DD} (device supply), whichever is appropriate for the logic circuit involved.

2.3.1 Handling of Unmounted Chips

In handling of unmounted chips, care should be taken to avoid differences in voltage potential. A conductive carrier, or a carrier having a conductive overlay should be used.

Some CMOS chips use a double diode plus resistor input protection circuit. This input protection circuit will clamp the input voltage to the chip to within the V_{DD} and V_{SS} supply voltages. This will protect the CMOS chip from over-voltages of 100 volts for 0.1 millisecond. The static charges encountered in normal handling may be as high as 10,000 volts.

2.4 Pin Assignments

Refer to Table 2.1 for the Bus Pinout Assignments for the MEK6802D5.

2.5 System Bus Signal Description

System Bus Signals:

(1) 16 Address lines (AØ-A15) - For selection of one byte or location from 64K bytes or locations.

(2) 8 Data lines (DØ-D7) - Bidirectional lines that carry data information.

(3) 9 Control lines:

(a) E - ENABLE. This signal is the system clock. A standard 3.579545 MHz crystal is used by the processor to generate the 894.8 KHz system clock.

TABLE 2.1. SYSTEM BUS PINOUTS

Pin Number	Description	Pin Number	Description
1	+5 VDC	A	+5 VDC
2	+5 VDC	B	+5 VDC
3	+5 VDC	C	+5 VDC
4	$\overline{\text{HALT}}$	D	$\overline{\text{IRQ}}$
5	$\overline{\text{RESET}}$	E	$\overline{\text{NMI}}$
6	R/$\overline{\text{W}}$	F	VMA
7		H	
8	+12 V GND (ref)	J	E (Ø2)
9	+12 V GND (ref)	K	+12 V GND (ref)
10		L	
11	-12 V (ref)	M	-12 V (ref)
12		N	
13		P	BA
14		R	MR
15		S	
16	+12 V (ref)	T	+12 V (ref)
17		U	
18		V	
19		W	
20		X	
21		Y	
22		Z	
23		$\overline{\text{A}}$	
24		$\overline{\text{B}}$	
25	-5 V (ref)	$\overline{\text{C}}$	
26		$\overline{\text{D}}$	
27		$\overline{\text{E}}$	
28		$\overline{\text{F}}$	
29	D1	$\overline{\text{H}}$	D3
30	D5	$\overline{\text{J}}$	D7
31	DØ	$\overline{\text{K}}$	D2
32	D4	$\overline{\text{L}}$	D6
33	A15	$\overline{\text{M}}$	A14
34	A12	$\overline{\text{N}}$	A13
35	A11	$\overline{\text{P}}$	A10
36	A8	$\overline{\text{R}}$	A9
37	A7	$\overline{\text{S}}$	A6
38	A4	$\overline{\text{T}}$	A5
39	A3	$\overline{\text{U}}$	A2
40	AØ	$\overline{\text{V}}$	A1
41	GND	$\overline{\text{W}}$	GND
42	GND	$\overline{\text{X}}$	GND
43	GND	$\overline{\text{Y}}$	GND

2.5 System Bus Signal Description (cont'd)

(3) 9 Control lines (cont'd)

(b) R/$\overline{W}$ - Read/Write. This is the read-write control line. Its logic state determines the direction of data (into or out of) a selected chip. When high, the data direction is toward the processor (read), and when low, the data direction is away from the processor (write).

(c) VMA - Valid Memory Address. When low this signal indicates that the address on the bus is invalid.

(d) MR - Memory Ready. This control signal can cause the E signal to be stretched. When MR is high, E will be in normal operation. When MR is low, E may be stretched an integral multiple of half periods, thus allowing interface to slow devices.

(e) $\overline{\text{RESET}}$. This line is used to reset and start the MPU.

(f) BA - Bus Available. When this signal is active, the MC6802 is stopped and the address bus is available for external devices (but not tri-state).

(g) $\overline{\text{HALT}}$. When this signal is active, all activity of the MC6802 will be halted.

(h) $\overline{\text{IRQ}}$ - Interrupt Request. This signal requests than an interrupt sequence be generated within the MC6802. If the interrupt mask bit in the condition code register is set, the $\overline{\text{IRQ}}$ input is disabled.

(i) $\overline{\text{NMI}}$ - Non-Maskable Interrupt. This signal is similar to $\overline{\text{IRQ}}$ except that the interrupt mask bit in the condition code register has no effect on $\overline{\text{NMI}}$.

2.6 System Interfaces

2.6.1 Power

For the minimum system, an on-board power supply is provided which is driven by an 18 Vac center tapped transformer. Refer to Figure 2.1 (see page 2-5) in connecting the 18 Vac transformer.

If it is necessary to use an external supply for more power, just remove jumper E2 and connect the external supply as shown in Figure 2.2 (see page 2-5).

2.6.1 Power (cont'd)

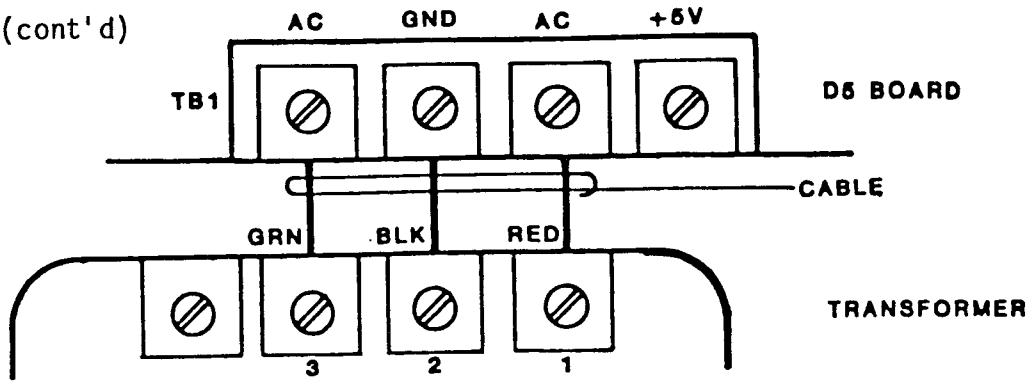


FIGURE 2.1. 18 VAC TRANSFORMER CONNECTION

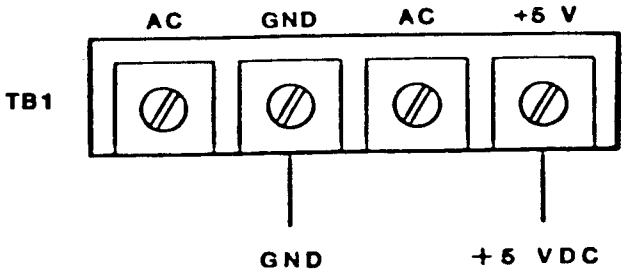


FIGURE 2.2. EXTERNAL SUPPLY CONNECTION

2.6.2 User PIA

A 40-pin socket has been provided to allow an additional uncommitted MC6821P in the D5. The PIA (U9) is supplied with the kit and the interface signals are available at the 24-pin socket (U14).

TABLE 2.2. USER PIA CONNECTOR PINOUTS

Pin Number	Description	Pin Number	Description
1	PA6	13	GND
2	PA7	14	N.C.
3	PBØ	15	PA5
4	PB1	16	PA4
5	PB2	17	PA3
6	PB3	18	PA2
7	PB4	19	PA1
8	PB5	20	PAØ
9	PB6	21	CA2
10	PB7	22	CA1
11	CB1	23	N.C.
12	CB2	24	+5 V

2.6.2 User PIA (cont'd)

The IRQA and IRQB outputs of the user PIA may be connected to either $\overline{NMI}$ or $\overline{IRQ}$. Refer to the following figure while making connections.

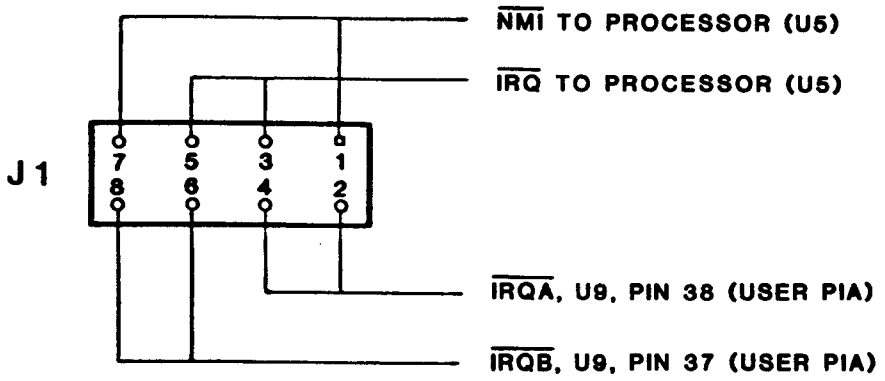


FIGURE 2.3. USER PIA INTERRUPT CONNECTIONS

The User PIA is decoded to appear at $E480.

PORT A	$E480
DATA DIRECTION A	$E480
CONTROL REG. A	$E481
PORT B	$E482
DATA DIRECTION B	$E482
CONTROL REG. B	$E483

2.6.3 Cassette

A 300 Baud Kansas City standard compatible cassette interface is provided for program storage. Most low-cost cassette recorders are compatible with the D5. The output from the cassette is connected to the "EAR" input at P3 of the D5. Generally the higher the cassette output level the better, but the D5 may accept as low as 2.0 Vpp.

To record programs connect the "MIC" output from P2 of the D5 to the "MIC" input to the cassette recorder. The D5 sends a 50 mVpp signal of 1200 Hz and 2400 Hz for zeroes and ones respectively.

2.7 Options

Several options are available for the user through installation or removal of various jumpers or components. In general, addition of options will require additional power which the on-board power supply may or may not be capable of supplying. The limiting factor in the on-board supply is the 7805 regulator heat dissipation characteristics. Some added power capability can be achieved by going to a TO-3 type regulator which can dissipate more heat. If the on-board regulator cannot supply enough power for the added options, then E2 may be removed to disconnect this regulator from the system allowing power to be supplied at the edge connector, or through the four (4) terminal barrier block TB1.

2.7.1 User RAM

Provisions have been made to allow addition of 1K x 8 of User RAM. This RAM is decoded to appear at addresses $E000 - $E3FF. To add this option install (2) MCM2114 1K x 4 RAMs at U7 and U8.

2.7.2 User ROM

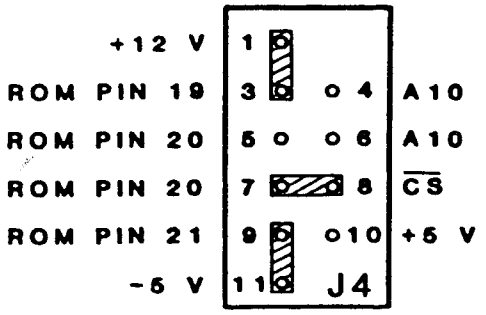
A 24-pin socket has been provided to allow the user to add an additional ROM with its own firmware. Decoding is provided to allow this ROM to appear at either of two areas in the memory map, $E800 to $EFFF or $F800-$FFFF. If the user wishes the ROM to contain the system restart and interrupt vectors, select the ROM at $F800-$FFFF and disable the monitor ROM from mirroring there. The user then has full access to all operating system routines but has priority during restart and interrupts. In addition to selecting where this ROM will appear, the user can connect the socket to accept various ROM or EPROM types. The following table outlines the proper connections for the most common EPROM types.

TABLE 2.3. EPROM CONNECTIONS

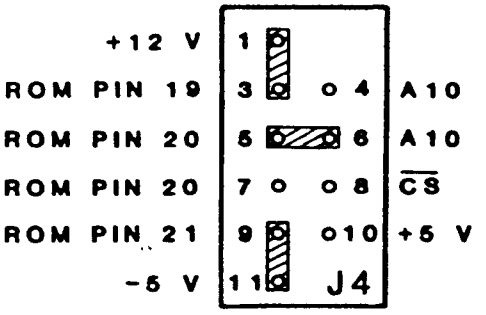
Types	Organization	Pin 21	Pin 20	Pin 19	Pin 18
MCM2708	1K x 8	-5 V	$\overline{CS}$	+12 V	GND
TMS2716	2K x 8	-5 V	A10	+12 V	$\overline{CS}$
MCM2716	2K x 8	+5 V	$\overline{CS}$	A10	GND

Refer to the following figures for configuring the ROM sockets as required. Figure 6.1, page 6-4 shows the location of these sockets.

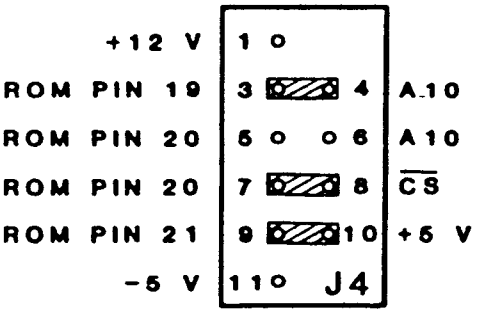
2.7.2 User ROM (cont'd)



a) Jumpers for 1K x 8, 3-supply EPROM.



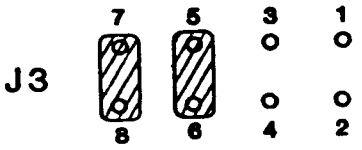
b) Jumpers for 2K x 8, 3-supply EPROM.



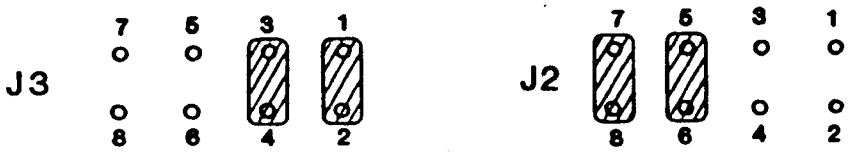
c) Jumpers for 2K x 8 single supply EPROM.

FIGURE 2.4. CONFIGURING THE ROM SOCKET

2.7.2 User ROM (cont'd)



a) Option 1, select at $E800 to $EFFF.



b) Option 2, select at $F800 to $FFFF.

FIGURE 2.5. USER ROM OPTIONS

To use option 2 the system monitor mirror must be disabled by removing the jumper between pins 1-2 of J2.

2.8 Bus Expansion

To expand the D5 through the 86-pin connector, install (2) 74LS244 octal buffers at U1 and U2 and (1) 74LS245 octal bidirectional buffer at U3. Also install a jumper at E1 to disable the MC6802's internal RAM.

CHAPTER 3

OPERATION

3.1 Common Terms and Notations

Microprocessors are digital devices which understand 1 (ones) and Ø (zeroes). In the MC6802 there is a 16-bit address bus and an 8-bit data bus which communicate digitally (with binary 1's and 0's). It becomes cumbersome to specify 16-bit addresses and 8-bit data in binary 1's and 0's so a shorthand code called hexadecimal is used. Each hexadecimal digit (hex digit for short) represents (4) binary bits.

In order to distinguish hex digits from ordinary decimal numbers, precede the hex number with a dollar sign ($). Sometimes it is very obvious that a number is a hex number and the "$" symbol may be left off as in program listings. In a binary number, the leftmost bit is called the Most Signigicant Bit (MSB) and the rightmost bit is called the Least Significant Bit (LSB).

Any group of 8-bits is called a "byte" and a group of 4-bits may be called a "nibble". A "word" is a misleading term because many think it means 8-bits, but "word" really refers to the size of the data word.

In most microprocessors the data bus just happens to be 8-bits wide but there are exceptions such as the MC68000 which has a 16-bit "word" size.

Binary Number	Hex Representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Note: From 10 through 15 the letters A through F are used for a single symbol to represent the 4-bit binary number.

3.1 Common Terms and Notations (cont'd)

Since 7-segment displays are used on the D5, the hex values A through F are difficult to display. The letters B and D become lower case letters.

3.2 Display Indications

Step-by-step examples will show the resulting display. Notice there are 6 displays. In the examples a small letter x indicates "don't care"; that is it may not be possible to predict the actual value which will appear but it indicates that the display will not be dark.

3.3 Keypad Entries and Displays

Located at the end of this chapter are detailed examples which will help explain this section.

3.3.1 RESET

To reset the entire system, depress the "RS" button and the resulting display will be a dash in the left hand position.

3.3.2 ESCAPE

Normally it is possible to abort a user program by depressing the "EX" key; however, since this is dependent upon proper set-up of the keypad PIA, it is possible for the user's program to disable the ESCAPE function.

Assuming a user program and the ESCAPE function active at the time the "EX" key is depressed, the resulting display would reflect the current user Program Counter status and the system would be in the register display editor.

If the system was within the DEBUG operating system when "EX" was depressed the resulting display would be a dash in the left hand position.

3.3.3 Miscellaneous

While entering numeric information, the first digit entry will clear the entire register location being modified and the new digit will be rotated into the least significant 4 bits of the register. Subsequent entries will be shifted in from the right and there is no limit to the number of digits which may be entered. When the next command key is depressed, the value being displayed is the value which will be used.

3.3.3 Miscellaneous (cont'd)

When trying to GO to a user program, a check is performed to see that the user's system stack can support the process of exiting to the program. The method used to transfer control to the user program requires that the user's register contents be put on the system stack and, if no RAM exists at the "S" stack pointer location, then control cannot be passed to the program. The display will be -SP- ??.

3.3.4 Memory Display/Change

To display or change the contents of a memory location, first enter the hexadecimal address to be changed. Leading zeroes are not required and digits may be entered until the display is correct. When this address has been entered, depress the "M" key.

The first four display digits show the address of the memory location being operated on. The fifth and sixth displays show the actual contents of the displayed address and if they do not agree with the data entered, it is an indication that the location is not properly accepting data.

To advance to the next higher memory location, depress "GO" and to back-up to the next previous location, depress the "M" key.

3.3.5 Offset Calculation

To calculate a relative offset, depress "FS" from the memory display change editor. The memory location being displayed at the time the "FS" key was depressed is assumed to be the location of the offset byte. The display at this point will be an A in the right most position,

prompting an entry of the destination address. After this address has been entered, depress "GO". The offset will be calculated and checked to see if it is within the range of an 8-bit offset. The resulting display will be two numbers in the right most two displays or the letters BAD indicating the offset was out of range. The numbers are an 8-bit offset value.

Note this offset on the listing before continuing.

Depress "GO". The calculated offset is stored and control returned to the memory display/change editor. The memory location being displayed will be that immediately following the offset which was just stored.

3.3.5 Offset Calculation (cont'd)

If, for some reason, the calculated value should not be stored prior to continuing, depress the "FC" key which will return control to the memory display/change routine. The displayed memory location will be the same one from which the offset calculation routine was called.

If the calcuation was "BAD" depressing "M" will return to the memory change editor.

3.3.6 Register Display/Alter

To get to register display, begin at the prompt point and depress "RD". The first register displayed will be the user's program counter.

3.3.7 Summary of Register Display Formats

pc	pc	pc	pc
		a	a
		b	b
x	x	x	x
sp	sp	sp	sp
		c	c

P	C
	A
	b
I	d
S	P
C	C

User program counter

User A - register

User B - register

User X index register

User stack pointer

User condition codes

To advance to the next sequential register depress the "GO" key and to back up to the next previous register depress the "M" key. The registers "wrap-around" so depressing "GO" while displaying CC, the next display will be PC.

To change the contents of a psuedo register, just step to that display and enter the desired value.

3.3.8 Trace Single Step

To trace a single MC6802 instruction, just depress "T/B" from within the register display editor. After executing a single user instruction, the operating system will regain control in the register display editor and all registers will reflect the action of the executed instruction. Depress "T/B" as many times as desired to execute that many instructions.

3.3.9 Go To User Program

To Go to a user program, start at the prompt point (a dash in the left most display). Enter the address desired and then depress "GO".

3.3.10 Continue

This is like GO to user program except instead of specifying the GO address, it is assumed to be the current value at the user program counter pseudo register. To invoke this function, depress "GO" from the "-" prompt point. It should not be necessary to enter any numbers prior to "GO" to invoke the "CONTINUE" function.

3.3.11 Breakpoints

To enter the breakpoint editor, depress "FS" then "T/B" from the prompt point.

To get out of the breakpoint editor, depress the "EX" key. To advance to another breakpoint specification, depress "GO". The advance function wraps around so successive presses of "GO" will eventually display all breakpoints. Up to 5 breakpoints may be active at any time.

To de-activate an existing breakpoint, advance to where it is being displayed and depress "FC". The last digit of the display (# of breakpoints) will decrement and another breakpoint will be displayed. To remove all breakpoints, depress "FC" as many times as needed until the "# of breakpoints" digit is zero.

To install a new breakpoint, enter its proposed hexadecimal address, then depress "FS". Breakpoints will only be installed where there is a RAM. If the "# of breakpoints" digit does not change, it is an indication that the breakpoint could not be installed either because five breakpoints already exist or because the breakpoint code ($3F) could not be stored at the proposed address.

3.3.12 Punch

To punch a cassette tape, begin from the prompt (-) point. Depress "P/L" and the display will be 0000 bb, prompting for a beginning address.

For example, depress 1, 2, 3. The display will be 0123 bb.

Depress GO to enter this address. The display will now be 0000 EE, as a prompt to enter the last address of the data to be punched.

For example, depress 4, 5, 6, 7. The display will be 4567 EE.

3.3.12 Punch (cont'd)

Turn on the tape recorder in the RECORD mode and depress GO. A 30 second leader of $FF characters will preceed the data file.

When the punch process is complete, the prompt (-) returns and the D5 board is ready for the next command.

3.3.13 Load/Verify

The load command is "FS" then "P/L" while the verify command is "FS" - "RD".

When the load or verify is completed without errors, the prompt returns and the D5 is ready for the next command.

If an error occurred during load or verify, the message "Fail ??" appears on the display. The user may escape at this point and enter register display to find the error location. The address in error is indicated by the value in the X-register. Also the data in the A-register is the last byte read from the tape.

3.3.14 User Functions

User functions may be performed by use of the function set key and the user function pointer (FNCPNT).

User functions are invoked by typing "FS" and a hex digit.

User functions are set up by providing a table of addresses of user supplied functions, and by storing the address of the table at FNCPNT

Up to 16 functions may be added to the functions already on the D5.

Display and Programming Examples

These examples are given to illustrate programming techniques and applications using the MEK6802D5.

Example to display 'USE d5'

1. Determine the necessary 7-segment display codes required for the letters and numbers to be displayed. This information is found in the detailed description of the "PUT" routine in Chapter 7.

Symbol		Binary		Hex
U	=	0011 1110	=	$3E
S	=	0110 1101	=	$6D
E	=	0111 1001	=	$79
space	=	0000 0000	=	$00
d	=	0101 1110	=	$5E
5	=	0110 1101	=	$6D

2. Each symbol is listed with its binary equivalent and hex code keypad entry.

3. The program listing entries are explained in the D5BUG listing (Chapter 7). The complete program listing for this example is as follows (next page):

```
                          USED5

0001                              NAM     USED5
0002                              OPT     LLEN=80,CREF
0003A 0000                        ORG     $0
0004                        * D5BUG ROUTINE
0005           E41D  A DISBUF EQU     $E41D
0006           F0A2  A DIDDLE EQU     $F0A2
0007           E419  A MNPTR  EQU     $E419
0008           F0BB  A PUT    EQU     $F0BB
0009                        *
0010                        *
0011A 0000 86 3E     A BEG     LDAA    #$3E      "U"
0012A 0002 B7 E41D   A         STAA    DISBUF    STORE TO FIRST DISPLAY
0013A 0005 86 6D     A         LDAA    #$6D      "S"
0014A 0007 B7 E41E   A         STAA    DISBUF+1
0015A 000A 86 79     A         LDAA    #$79      "E"
0016A 000C B7 E41F   A         STAA    DISBUF+2
0017A 000F 86 00     A         LDAA    #$00      BLANK
0018A 0011 B7 E420   A         STAA    DISBUF+3
0019A 0014 86 5E     A         LDAA    #$5E      "D"
0020A 0016 B7 E421   A         STAA    DISBUF+4
0021A 0019 86 6D     A         LDAA    #$6D      "5"
0022A 001B B7 E422   A         STAA    DISBUF+5  STORE TO LAST DISPLAY
0023A 001E CE F0A2   A         LDX     #DIDDLE   ADDR OF DIDDLE ROUTINE
0024A 0021 FF E419   A         STX     MNPTR     ESTABLISH AS ACTIVE SUB OF "PUT"
0025A 0024 7E F0BB   A         JMP     PUT       CALL DISPLAY ROUTINE
0026                              END
     ERRORS 00000--00000


  0000 BEG      0011*
  F0A2 DIDDLE   0006* 0023
  E41D DISBUF   0005* 0012  0014  0016  0018  0020  0022
  E419 MNPTR    0007* 0024
  F0BB PUT      0008* 0025
```

After entering this program into memory using Memory Change, press "Ø" and "GO" to begin execution.

Example to display 'HELP --'

1. Determine the 7-segment codes required, as in the previous example.

Symbol		Binary			Hex
H	=	0111	0110	=	$76
E	=	0111	1001	=	$79
L	=	0011	1000	=	$38
P	=	0111	0011	=	$73
-	=	0100	0000	=	$40
-	=	0100	0000	=	$40

2. Each symbol is listed with its binary and hex codes equivalent.

3. The program listing entries are explained in the D5BUG listing (Chapter 7). The complete program listing for this example is as follows (next page):

HELP

```
0001                                NAM    HELP
0002                                OPT    LLEN=80,CREF
0003A 0000                          ORG    $0
0004                          * D5BUG ROUTINES
0005          FOA2  A DIDDLE EQU    $FOA2
0006          E41D  A DISBUF EQU    $E41D
0007          E419  A MNPTR  EQU    $E419
0008          FOBB  A PUT    EQU    $FOBB
0009                          *
0010                          *
0011A 0000 CE 7679  A BEG    LDX    #$7679   "HE"
0012A 0003 FF E41D  A        STX    DISBUF   STORE TO FIRST 2 DISPLAYS
0013A 0006 CE 3873  A        LDX    #$3873   "LP"
0014A 0009 FF E41F  A        STX    DISBUF+2
0015A 000C CE 4040  A        LDX    #$4040   "--"
0016A 000F FF E421  A        STX    DISBUF+4 STORE TO LAST 2 DISPLAYS
0017A 0012 CE FOA2  A        LDX    #DIDDLE  ADDR OF DIDDLE ROUTINE
0018A 0015 FF E419  A        STX    MNPTR    ESTABLISH AS ACTIVE SUB OF PUT
0019A 0018 7E FOBB  A        JMP    PUT      CALL DISPLAY ROUTINE
0020                          END
```

```
0000 BEG     0011*
FOA2 DIDDLE  0005* 0017
E41D DISBUF  0006* 0012  0014  0016
E419 MNPTR   0007* 0018
FOBB PUT     0008* 0019
```

After entering this program into memory using Memory Change, press "Ø" and "GO" to begin execution.

To help in better understanding the PUT/MNPTR routine operation, try the following:

DIDDLE has been used so far as the subroutine which is executed once each millisecond from PUT. Now substitute a long delay using the DLYX routine as the subroutine which is specified by MNPTR.

```
0020  CE  FFFF   SLOW   LDX   #$FFFF  For 1/2 second delay

0023  BD  F179          JSR   DLYX    Delay

0026  39               RTS            Then return to PUT
```

To install SLOW as the active subprogram of PUT, just change the third last step of the "HELP --" program:

was:

```
0012  CE  FOA2   LDX  #DIDDLE  Addr of DIDDLE routine
```

to:

```
0012  CE  0020   LDX  #SLOW   Addr of SLOW routine
```

When the program is running between successive digits (which are on for about 1 millisecond) there is an approximately half second delay.

Adjust the amount of delay by changing FFFF at 0021,2 and observe the effect.

This example is a clock program to display time in a 24-hour format. The program is as follows (next page):

TIME

```
0001                                    NAM     TIME
0002                                    OPT     LLEN=80,CREF
0003A 0000                              ORG     $0
0004                            * D5BUG ROUTINE
0005            F0BB   A PUT    EQU     $F0BB
0006            E419   A MNPTR  EQU     $E419
0007            E42C   A HEXBUF EQU     $E42C
0008            F120   A DYSCOD EQU     $F120
0009A 0000      0001   A RHR    RMB     1       HOURS
0010A 0001      0001   A RMIN   RMB     1       MINUTES
0011A 0002      0001   A RSEC   RMB     1       SECONDS
0012A 0003      0002   A TICK   RMB     2       COUNTERPASSES; ABOUT 1ms/PASS
0013                            *
0014A 0005 CE 000E   A START  LDX     #TIME   ADDR OF TIME ROUTINE
0015A 0008 FF E419   A        STX     MNPTR   ESTABLISH AS ACTIVE SUB
0016A 000B 7E F0BB   A        JMP     PUT     GET STARTED
0017                            *
0018A 000E DE 03     A TIME   LDX     TICK    # OF PASSES
0019A 0010 08                   INX             INCREMENT # PASSES
0020A 0011 DF 03     A        STX     TICK    UPDATE
0021A 0013 8C 0217   A        CPX     #535    PASSES/SEC
0022A 0016 26 4C 0064         BNE     TBOT    TO BOTTOM OF PROG
0023A 0018 CE 0000   A        LDX     #$0000
0024A 001B DF 03     A        STX     TICK    RESET TICK COUNT
0025A 001D 7C 0002   A        INC     RSEC    INCREMENT SECONDS
0026A 0020 96 02     A        LDAA    RSEC
0027A 0022 8A F5     A        ORAA    #$F5    CHECK FOR BCD OVERFLOW
0028A 0024 43                   COMA
0029A 0025 26 3D 0064         BNE     TBOT    IF NOT $xA SECONDS
0030A 0027 96 02     A        LDAA    RSEC
0031A 0029 8B 06     A        ADDA    #6      INCREMENT TENS DIGIT
0032A 002B 97 02     A        STAA    RSEC    UPDATE
0033A 002D 81 60     A        CMPA    #$60    60 SEC YET ?
0034A 002F 26 33 0064         BNE     TBOT    IF NOT; EXIT
0035A 0031 7F 0002   A        CLR     RSEC    RESET SCONDS COUNT
0036A 0034 7C 0001   A        INC     RMIN    NEW MINUTE
0037A 0037 96 01     A        LDAA    RMIN
0038A 0039 8A F5     A        ORAA    #$F5    CHECK FOR BCD OVERFLOW
0039A 003B 43                   COMA
0040A 003C 26 26 0064         BNE     TBOT    IF NOT; EXIT
0041A 003E 96 01     A        LDAA    RMIN
0042A 0040 8B 06     A        ADDA    #6      INCREMENT TENS DIGIT
0043A 0042 97 01     A        STAA    RMIN    UPDATE
0044A 0044 81 60     A        CMPA    #$60    60 MIN YET ?
0045A 0046 26 1C 0064         BNE     TBOT    IF NOT; EXIT
0046A 0048 7F 0001   A        CLR     RMIN    RESET MINUTES COUNT
0047A 004B 7C 0000   A        INC     RHR     NEW HOUR
0048A 004E 96 00     A        LDAA    RHR
0049A 0050 8A F5     A        ORAA    #$F5    CHECK FOR BCD OVERFLOW
0050A 0052 43                   COMA
0051A 0053 26 06 005B         BNE     CHKHR   CHECK FOR END OF A DAY
0052A 0055 96 00     A        LDAA    RHR
0053A 0057 8B 06     A        ADDA    #6      INCREMENT TENS DIGIT
0054A 0059 97 00     A        STAA    RHR     UPDATE HOURS
0055A 005B 96 00     A CHKHR  LDAA    RHR
0056A 005D 81 24     A        CMPA    #$24    END OF A DAY ?
0057A 005F 26 03 0064         BNE     TBOT
0058A 0061 7F 0000   A        CLR     RHR     RESET HOURS
```

```
0059A 0064 DE 00     A TBOT   LDX     RHR       HOURS/MINUTES
0060A 0066 FF E42C   A        STX     HEXBUF    TO HEX BUFFER
0061A 0069 96 02     A        LDAA    RSEC      SECONDS
0062A 006B B7 E42E   A        STAA    HEXBUF+2  TO LAST DIGITS
0063A 006E 7E F120   A        JMP     DYSCOD    CONVERT TO 7-SEG
0064                      * YOU WILL RETURN TO PUT AT CONCLUSION OF DYSCOD
0065                      END
     ERRORS 00000--00000


  005B CHKHR    0051  0055*
  F120 DYSCOD   0008* 0063
  E42C HEXBUF   0007* 0060  0062
  E419 MNPTR    0006* 0015
  F0BB PUT      0005* 0016
  0000 RHR      0009* 0047  0048  0052  0054  0055  0058  0059
  0001 RMIN     0010* 0036  0037  0041  0043  0046
  0002 RSEC     0011* 0025  0026  0030  0032  0035  0061
  0005 START    0014*
  0064 TBOT     0022  0029  0034  0040  0045  0057  0059*
  0003 TICK     0012* 0018  0020  0024
  000E TIME     0014  0018*
```

After entering this program into memory using Memory Change, press "5" and "GO" to begin execution.

Comments on 24 hour clock program "TIME"

The technique used to measure time is called "software timing" and relies on the system clock accuracy and the sum of the execution times on all of the instructions being performed. This is a cumbersome technique to do right.

Determine all the different paths through the program in order to accurately predict the program's total execution time.

This is precisely the technique used in some digital watches, but in that case there is a fine adjustment to tune the clock frequency to better than .001% which comes out to about 5 minutes per year. In a typical microprocessor system, the clock speed is not adjustable and accuracy is about 0.1% which comes out to more than a minute a day.

As for calculating actual program execution time, look at the program to see how many different paths there are <u>just</u> from the label "TIME" to "TBOT":

1. Not end of tick.
2. End of tick, not end of minute, not a tenth second.
3. End of tick, not end of minute, second tens digit incr.
4. End of tick, end of minute, not end of hour, not a tenth minute.
5. End tick, end minute, not end hour, minute tens digit incr.
6. End tick, end minute, end hour, not end day, not a tenth hour.
7. End tick, end minute, end hour, end day.

Each of the above paths through the TIME program takes a different length of time because a different sequence of instrucitons is followed. That's just part of the task; to completely calculate the execution time, do the same thing to every routine being executed. Then to find the total, determine exactly how many times each path is followed and add up all those times.

CHAPTER 4

THEORY OF OPERATION

4.1 General

The MEK6802D5 allows the user to become familiar with the MC6800 family of microprocessors. Refer to the block diagram, Figure 4.1, and the component function location diagram, Figure 4.2.

The three basic components of the D5 are (1) U5, the MC6802 microprocessor, (2) U12, the MC68A1316E ROM, and (3) U23, the MC6821 PIA.

The MC6802 is the heart of the D5 system. The MC6802 is an eight (8)-bit microprocessor that can execute 72 different instructions (exact instruction set of the MC6800). Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

The MC6802 has a set of registers and accumulators plus an internal clock oscillator and driver.

For a more detailed explanation and description of the MC6802 Microprocessor refer to the data sheet located at the end of this manual.

The MEK6802D5 utilizes a 3.58 MHz crystal (Y1) to control this circuitry (this results in a system operating speed of 895 KHz). Power on/reset is accomplished by components R1 and C17.

Basic to the operation of the D5 system is the Data bus (eight lines labeled BDØ through BD7) and the Address bus (sixteen lines labeled BAØ through BA15). These buses interconnect to the various sections.

The address bus is further decoded by the Address Decode Logic (U6) into select lines. J2 and J3 must be set properly to accomplish this. These jumpers are specified in the installation section of this manual for the various board arrangements possible.

The ROM section is the permanent data storage. Unlike some types of memories, this ROM retains data whether the power is on or not.

The ROM section consists of the D5BUG Monitor (U12) and the optional User ROM (U13). J4 is used to configure the user ROM socket to accept various types of ROMs or EPROMs. J2 is used to position U13 (user ROM) in memory space. J3 is used to position U12 (D5BUG) in memory space.

The RAM section consists of a 128 byte static Random Access Memory. This memory will retain its stored data as long as power is supplied to it and can be written into and read out of as desired.

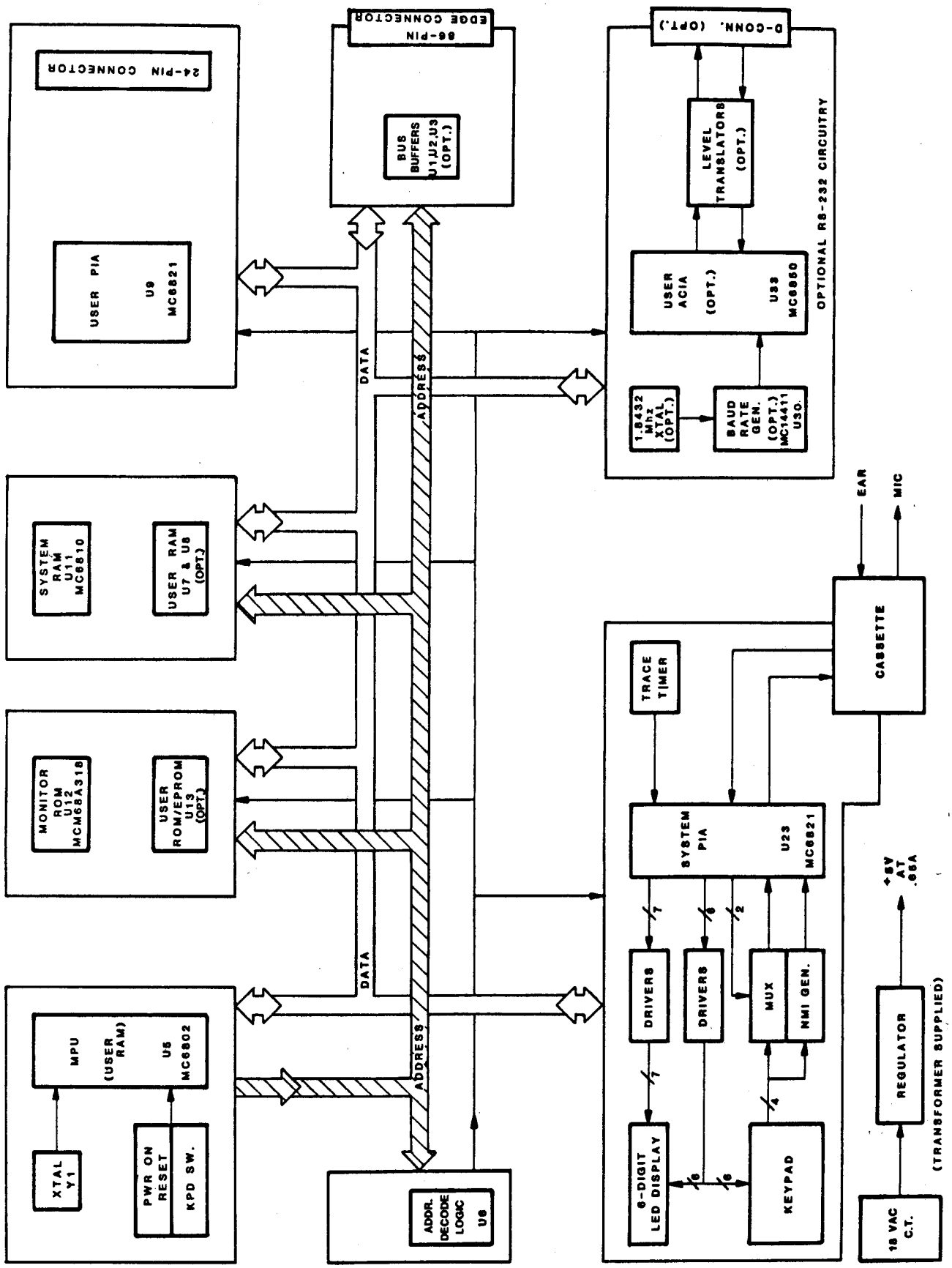


FIGURE 4.1. BLOCK DIAGRAM

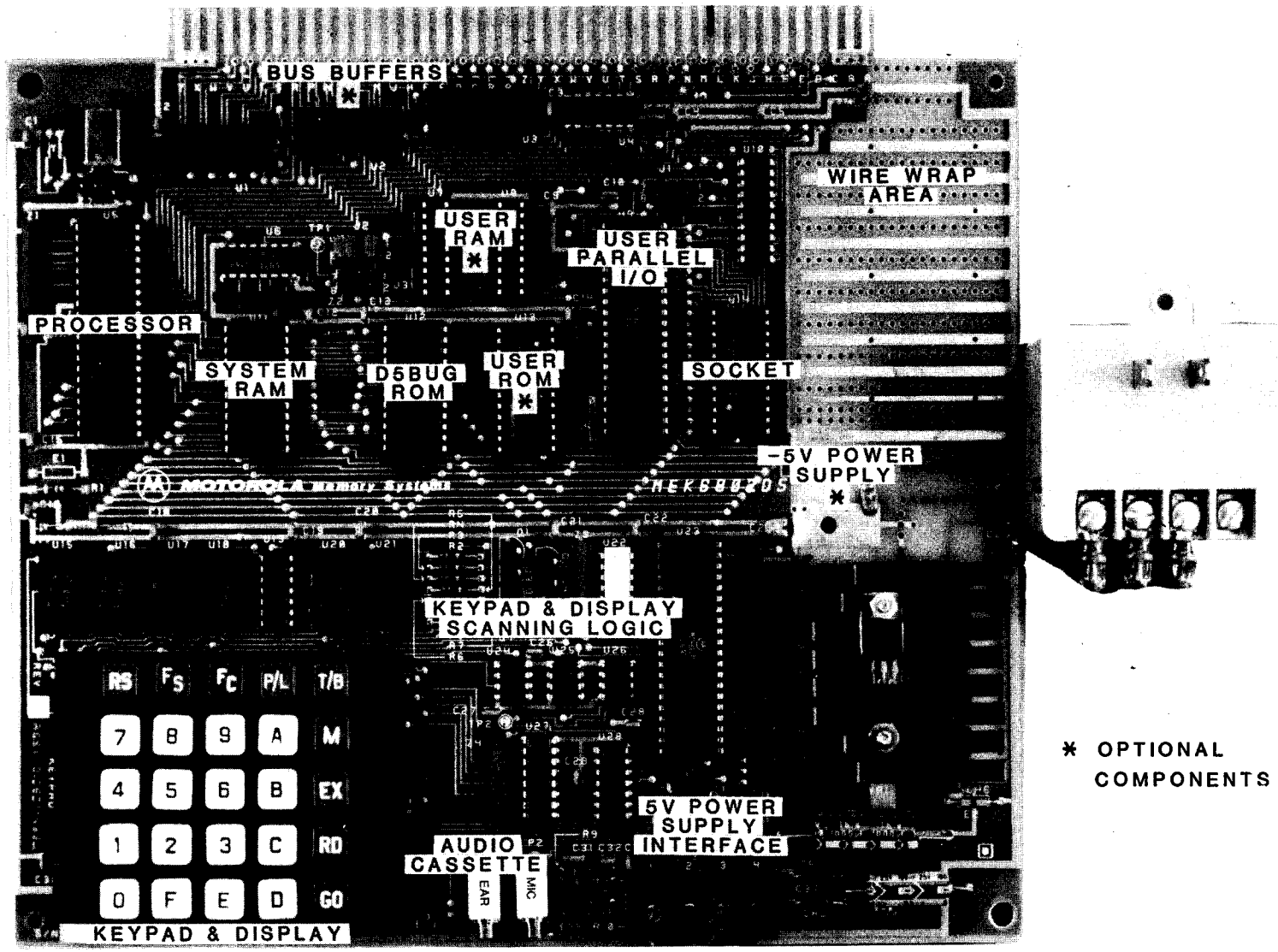


FIGURE 4.2. COMPONENT FUNCTION LOCATION DIAGRAM

The D5 uses this RAM as a Scratch Pad, which is a temporary storage for program material being processed. An example of this could be a program that uses a set of data a large number of times during the running of the whole program. This device would allow that data to be referred to as needed, rather than being repeated over again in the main program.

Another optional feature is the User RAM (U7 and U8). This provides the user with an additional (1K) bytes of memory. This feature is valuable when long programs must be stored.

These memories are volatile, and power must be supplied to them to retain their stored data.

The system PIA has two 8-line I/O ports and four control I/O lines for a total of 20 interface lines. The four control lines provide for a variety of hand shake applications. The two I/O ports are controlled by software to configure these 16 lines as inputs or outputs in any combination.

The User PIA (U9), allows the processor to talk to the outside world. Th 16 I/O lines and 4 control lines are connected to a 24-pin socket (U14).

The optional Bus Buffers (U1, U2, U3) are for expanding the D5 through an 86-pin edge connector. This is accomplished by plugging an IC in the supplied sockets. Note that U10, also a bus driver, is part of the basic D5 since the processor's R/W output cannot drive all the inputs in the basic system (or the bus). All the control signals, data, and addresses are available on this bus for system expansion.

The system PIA, (U23) uses its A side to drive the anode segments of the LED displays and the B side to drive the LED cathodes. The processor put data into the register of the PIA to display on one of the LED's, then places data in the PIA registers to drive the next display element, etc., at a rapid enough rate that the display appears steady.

The keypad is connected to the cathode drivers of the LED's so that when a key is depressed, an interrupt signal is generated. This interrupt signal causes the processor to call up a routine from D5BUG, which then enables it to search for the key that was closed. The PIA is used to enter new data or display data in the system.

The D5 can be supplied to interface to the widely used RS-232 bus. An Asynchronous Communications Interface Adapter (ACIA) converts the parallel data on the data bus to a serial form suitable for transmission

4.1 General (cont'd)

over a communications system. This ACIA will both transmit and receive data and allows the D5 to be used as a remotely controlled processor. An MC14411 Bit Rate Generator supplies the special frequency that the ACIA requires in making the conversion from parallel to serial data.

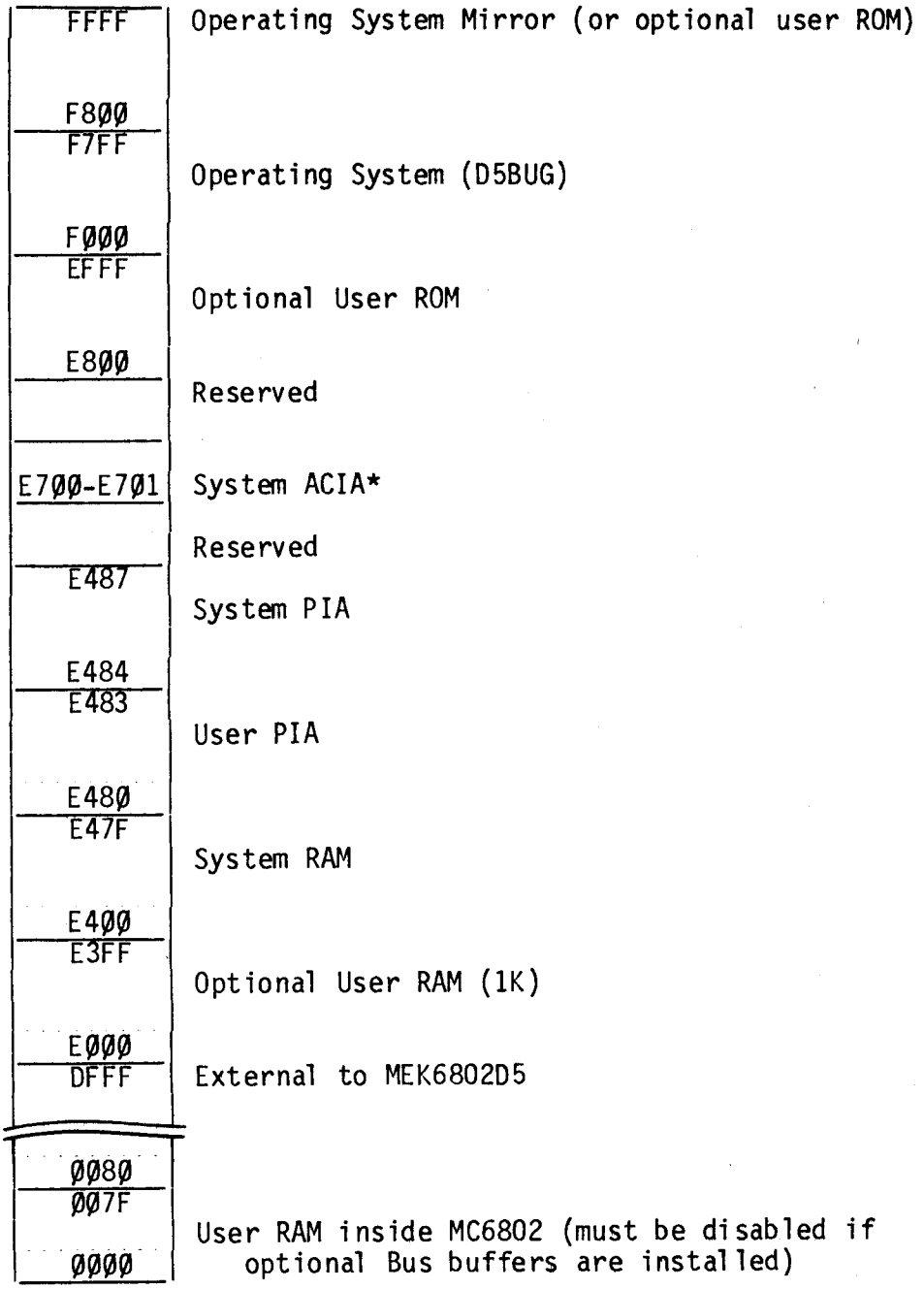
An installed jumper determines the frequency of data formatted in serial form. A socket allows cabling to the RS-232 line.

4.2 Addressing

The MEK6802D5 accesses 64K of memory space by use of 16 address lines AØ through A15. The addresses $EØØØ to $FFFF access parts on the MEK6802D5 and external data buffers are not enabled for such accesses. The area from $ØØØØ to $ØØ7F is RAM within the MC6802 and must be disabled by jumper option if the optional bus buffers are installed.

When the optional edge connector bus buffers are installed, all accesses to addresses $ØØØØ to $DFFF are considered to be off-board.

The system memory map is shown in Figure 4.3, the Address Decode Map in Table 4.1 and the System Timing Diagram in Figure 4.4.



*ACIA is not supported by D5BUG software.

FIGURE 4.3. MEMORY MAP

TABLE 4.1. ADDRESS DECODE MAP

Device	VMA	R/W	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	U No.
MC68A316E	1	X	1	1	1	1	*	.	.	.	.	.	.	.	.	.	.	.	U12
MCM6810	1	.	1	1	1	0	0	1	0	0	0	.	.	.	.	.	.	.	U11
MC6821 (User)	1	.	1	1	1	0	0	1	X	X	1	X	X	X	X	0	.	.	U9
MC6821 (Syst)	1	.	1	1	1	0	0	1	X	X	1	X	X	X	X	1	.	.	U23
MCM2114 (Opt.)	1	.	1	1	1	0	0	0	.	.	.	.	.	.	.	.	.	.	U7,U8
User ROM (Opt. 1)	1	.	1	1	1	0	1	.	.	.	.	.	.	.	.	.	.	.	U13
User ROM (Opt. 2)	1	.	1	1	1	**	1	.	.	.	.	.	.	.	.	.	.	.	U13
MC6802***	1	.	0	0	0	0	0	0	0	0	.	.	.	.	.	.	.	.	U5
MCM6850 (Opt.)	1	.	1	1	1	0	0	1	1	1	1	0	X	X	X	X	X	X	U33
74LS245 (Opt.)	1	.	$0000 - $DFFF (Must install E1)																U3

0 - Logic zero 1 - Logic one . -Both X - Don't Care

* - Normally both but by removing jumpers it becomes 0 only.

** - 1 but monitor mirror must be disabled.

*** - Inherent to the MC6802 by design, may be disabled for system expansion.

TABLE 4.2. AC OPERATING CHARACTERISTICS

Parameter	Symbol	Min	Max	Unit
Cycle	TCYC	1100	1130	ns
Address Setup	TAS	220		ns
Address Hold	TAH	20		ns
Write Data Valid	TDVW		240	ns
Write Data Hold	TDHW	20		ns
Address Delay	TAD		290	ns

TABLE 4.3. AC OPERATING CONDITIONS

Parameter	Symbol	Min	Max	Unit
Access Time[4]	TACC1 TACC2		380 710	ns ns
Data Hold Read[5]	TDHR	0		ns

Test Conditons:

1) Operating temperature, TA = 25°C.

2) Timing signals measured at I/O connector (50% points).

3) I/O loading conditions equivalent to 74LS type devices.

4) Measured from the rising edge of E (Ø2).

5) Measured from the falling edge of E (Ø2).

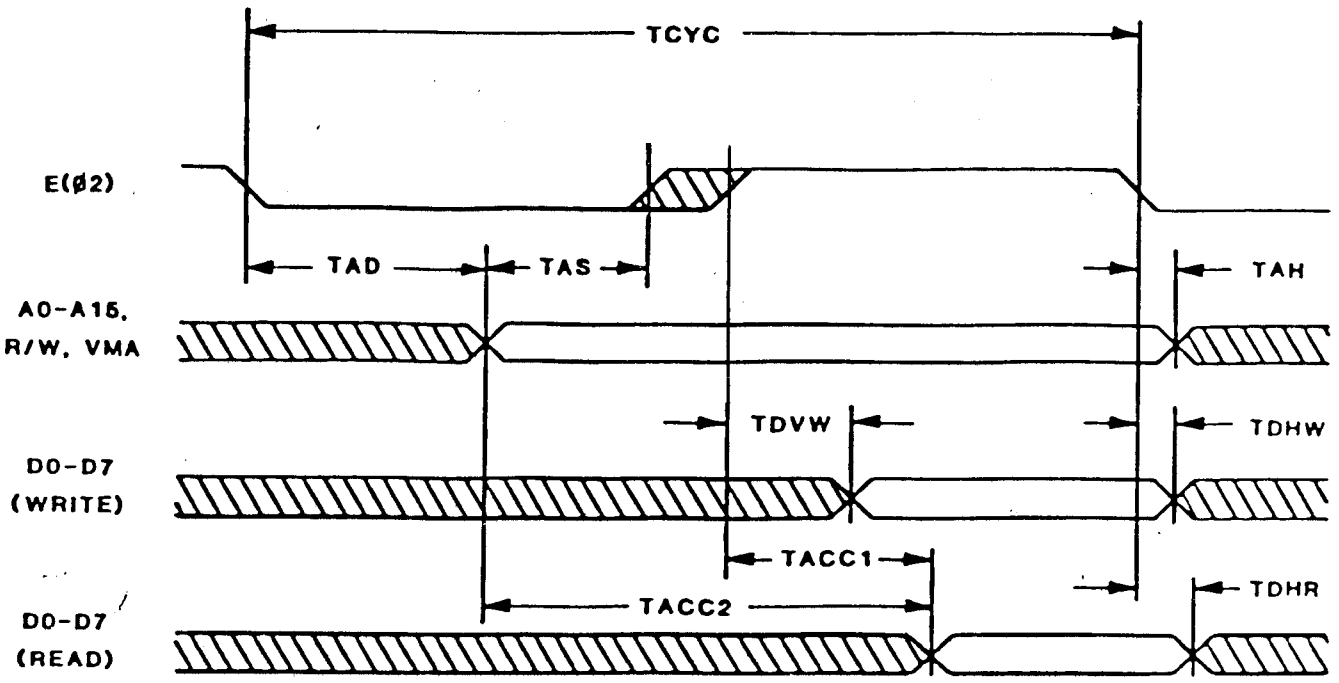


FIGURE 4.4. SYSTEM TIMING DIAGRAM

CHAPTER 5

MAINTENANCE

5.1 Troubleshooting

Troubleshooting techniques can be divided into several steps:

a) Check and analyze operation.

b) Test to isolate section, stage, and part.

c) Replace the defective part.

d) Performance test after repair.

For all troubleshooting, refer to the system schematic, Figure 5.1, and to the component location diagram, Figure 6.1, in Chapter 6.

Begin troubleshooting by checking the power supply voltages used with the D5. Check for secure and correct transformer connections.

The 86-pin edge connector fingers are tin lead plated and should be cleaned for reliable operation.

Use caution when replacing defective parts. Improper handling (refer to Chapter 2) and too much heat can destroy components.

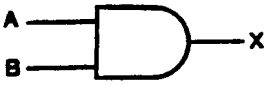
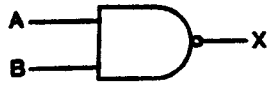
5.2 Standard Logic Symbols

5.2.1 Digital Logic Circuits

All digital equipment, simple or complex, is constructed from a few basic devices. These devices consist of electronic circuitry that provides an output based on the input of one, two, or more variables and are called logic elements (most often referred to as gates).

Digital logic circuits are broken into two basic types: decision-making and memory elements. All decision logic elements monitor binary inputs and produce outputs based on the input states and the operational characteristics of the logic element. Memory elements are used to store binary data. Table 5.1 gives the basic logic elements, their more commonly used forms, and their single binary output. All other digital logic elements and circuits are variations or combinations of these basic elements.

TABLE 5.1. LOGIC SYMBOLS AND TRUTH TABLES

Device	Logic Symbol	Truth Table *		
		A	B	X
INVERTER		L H	--- ---	H L
AND		L H L H	L L H H	L L L H
OR		L H L H	L L H H	L H H H
NAND		L H L H	L L H H	H H H L
NOR		L H L H	L L H H	H L L L
XOR (EXCLUSIVE OR)		L H L H	L L H H	L H H L

* The Truth Table is a table showing all possible combinations of inputs with the respective output for each set of inputs.

H = High voltage (logic 1)
L = Low voltage (logic 0)

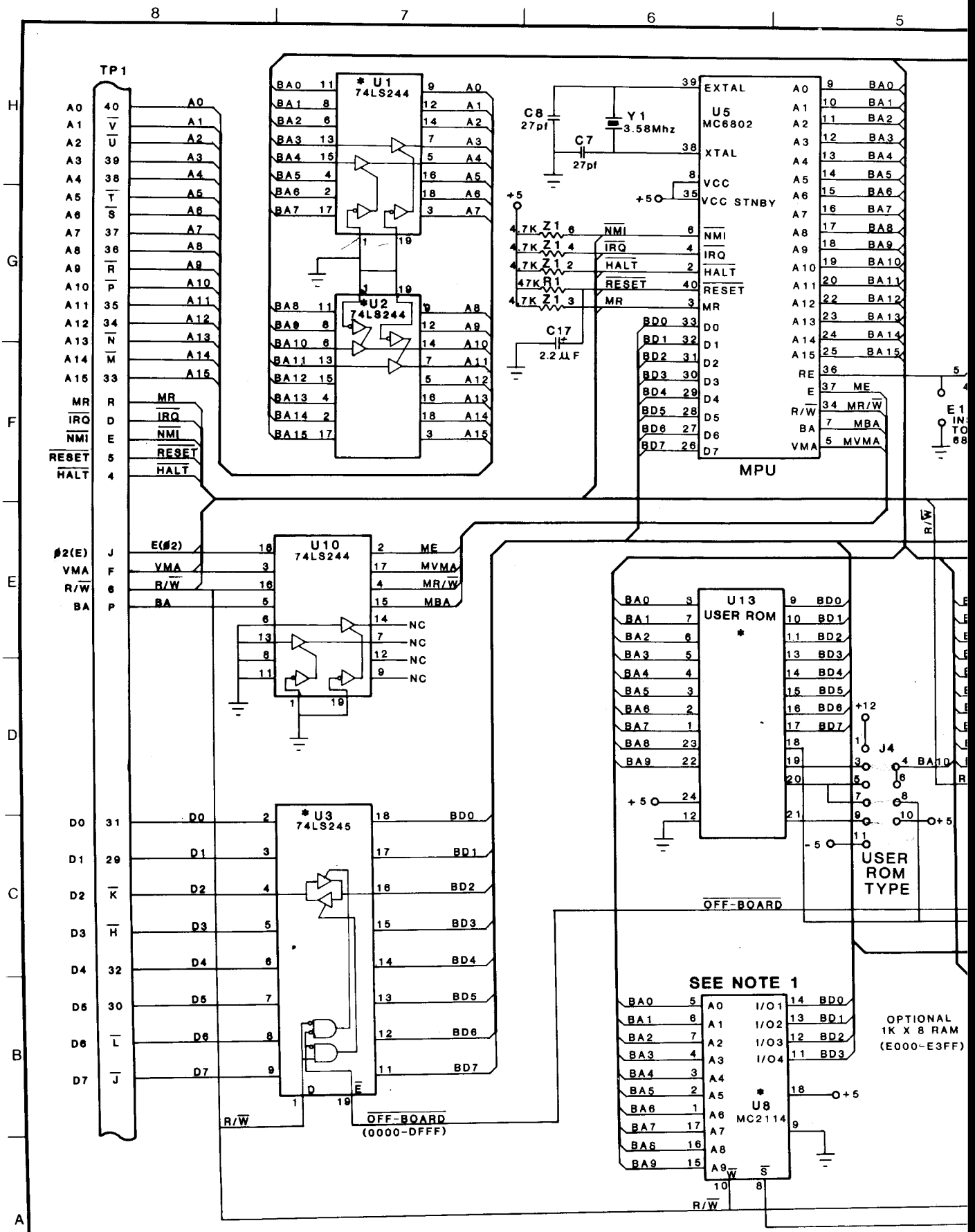
FIGURE 5.1. SCHEMATIC DIAGRAM

Sheet 1: Index Page

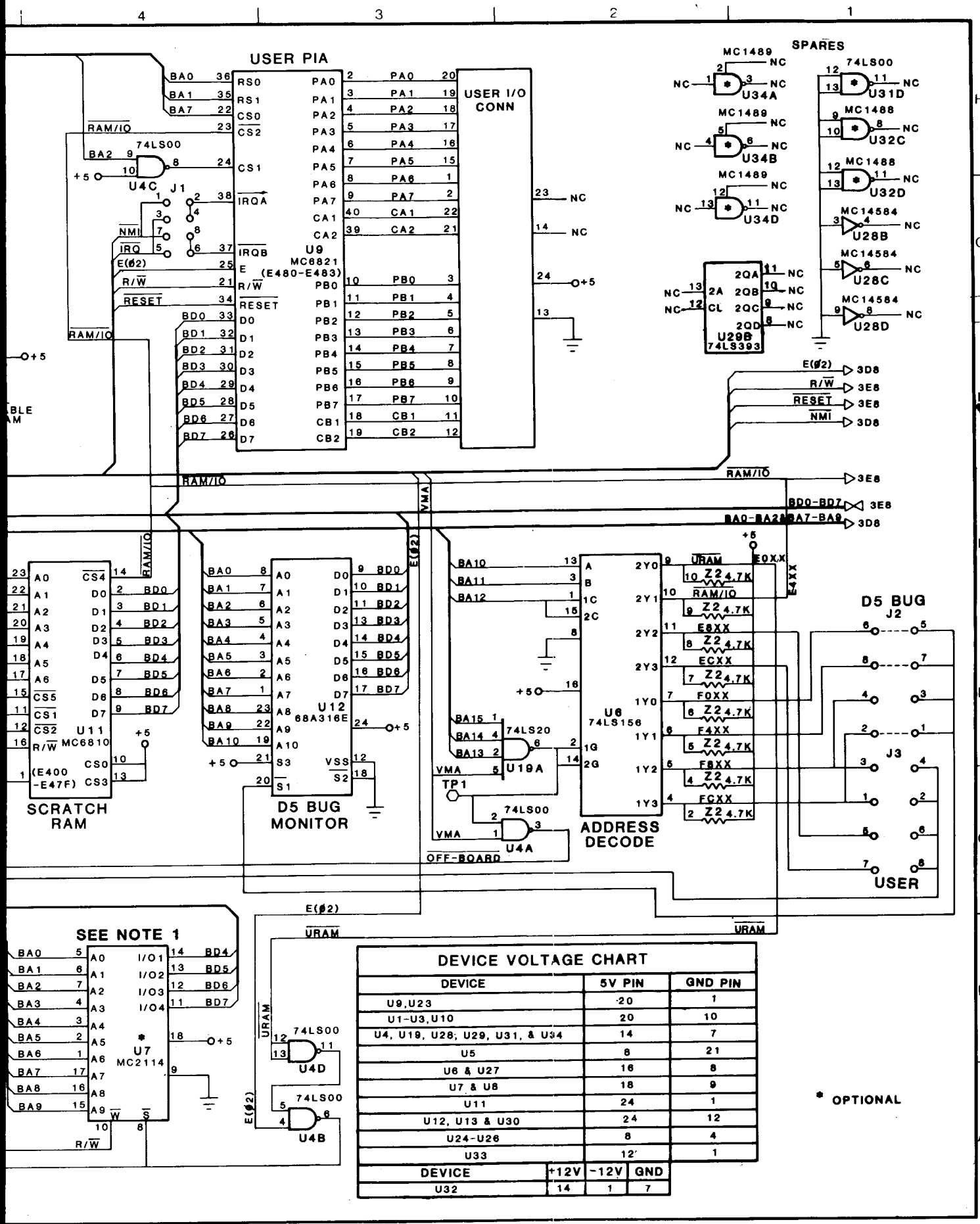
Sheet 2: Schematic Diagram

Sheet 3: Schematic Diagram

NOTE: This is sheet 1 of Figure 5.1. The schematic diagram consists of 2 sheets.



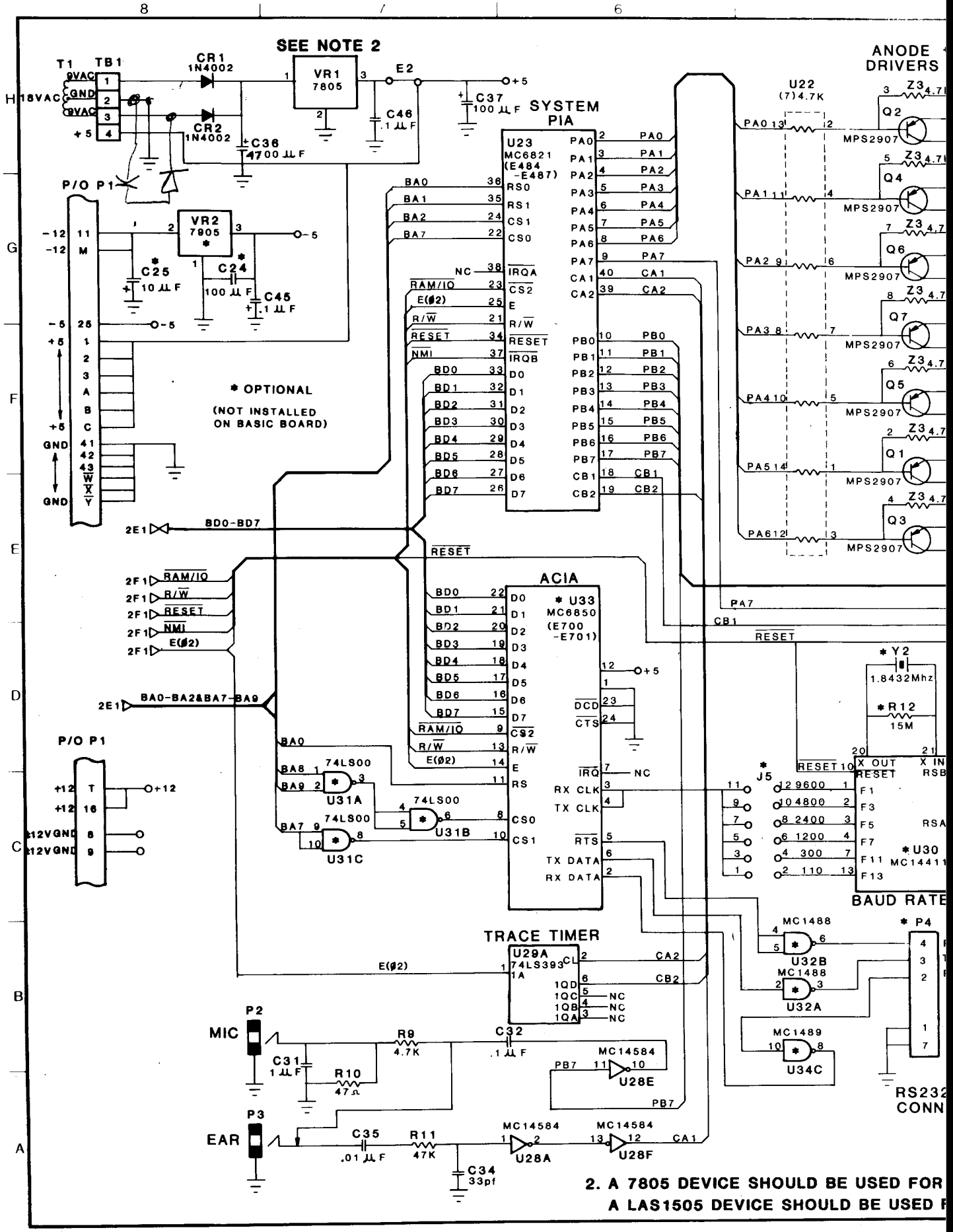
1. U7 AND U8 ARE OPTIONAL FOR THE MEK6802D5, STANDARD FOR THE MEK6802D5E.



DEVICE VOLTAGE CHART		
DEVICE	5V PIN	GND PIN
U9,U23	20	1
U1-U3,U10	20	10
U4, U19, U28; U29, U31, & U34	14	7
U5	8	21
U6 & U27	16	8
U7 & U8	18	9
U11	24	1
U12, U13 & U30	24	12
U24-U26	8	4
U33	12	1

DEVICE	+12V	-12V	GND
U32	14	1	7

FIGURE 5.1 SCHEMATIC DIAGRAM (SHEET 2 OF 3) 5-5/5-6


SEE NOTE 2
T1
TB1
CR1
1N4002
CR2
1N4002
VR1
7805
VR2
7905
C36
4700 µF
C37
100 µF
C46
.1 µF
C25
10 µF
C24
100 µF
C45
.1 µF
E2
+5
-5
-12
P/O P1
* OPTIONAL
(NOT INSTALLED
ON BASIC BOARD)
SYSTEM
PIA
U23
MC6821
(E484
-E487)
ANODE
DRIVERS
U22
(7)4.7K
Q1
Q2
Q3
Q4
Q5
Q6
Q7
MPS2907
BD0-BD7
RESET
RAM/IO
R/W
E(ø2)
NMI
BA0-BA2&BA7-BA9
ACIA
* U33
MC6850
(E700
-E701)
* Y2
1.8432Mhz
* R12
15M
* J5
BAUD RATE
* U30
MC14411
74LS00
U31A
U31B
U31C
TRACE TIMER
U29A
74LS393
MC1488
U32A
U32B
MC1489
U34C
* P4
RS232
CONN
P2
MIC
P3
EAR
R9
4.7K
C32
.1 µF
C31
1 µF
R10
47Ω
C35
.01 µF
R11
47K
C34
33pf
MC14584
U28E
U28A
U28F
2. A 7805 DEVICE SHOULD BE USED FOR
A LAS1505 DEVICE SHOULD BE USED F

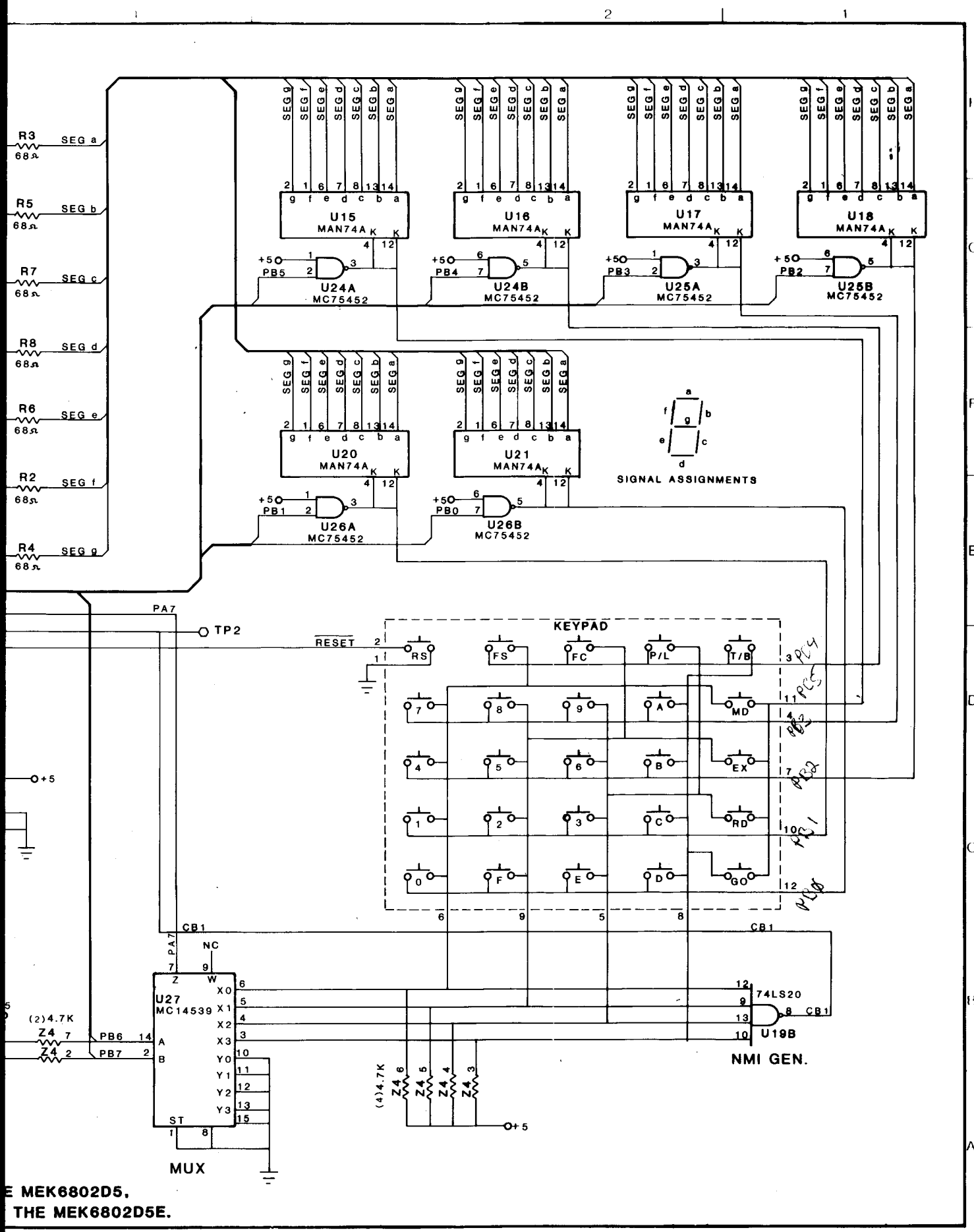


FIGURE 5.1 SCHEMATIC DIAGRAM
(SHEET 3 OF 3) 5-7/5-8

CHAPTER 6

PARTS LIST

6.1 General

This chapter consists of Figure 6.1, Board Outline and Component Layout, and Table 6.1, which is a listing of parts contained in the MEK6802D5 Microcomputer.

TABLE 6.1. MEK6802D5 PARTS LIST

Item No.	Quan.	Description	Part Number	Reference Designation
1	1	PC Board, Basic	MEK6802D5	--
2	1	Integrated Circuit	MCM68A316E	U12 (D5BUG)
3	1	Integrated Circuit	74LS00	U4
4	1	Integrated Circuit	74LS20	U19
5	1	Integrated Circuit	74LS156	U6
6	1	Integrated Circuit	74LS393	U29
7	1	Integrated Circuit	MC6802	U5
8	2	Integrated Circuit	MC6821P	U23, U9
9	1	Integrated Circuit	MC6810	U11
10	3	Integrated Circuit	75452	U24, U25, U26
11	1	Integrated Circuit	MC14539	U27
12	1	Integrated Circuit	74LS244	U10
13	1	Integrated Circuit	MC14584	U28
14	1	Resistor, 4.7K 1/4 Watt, 5%		R9
15	1	Resistor, 47 ohm 1/4 Watt, 5%		R10

TABLE 6.1. MEK6802D5 PARTS LIST (cont'd)

Item No.	Quan.	Description	Part Number	Reference Designation
16	2	Resister, 47K 1/4 Watt, 5%		R1, R11
17	7	Resister, 68 ohm 1/4 Watt, 5%		R2,R3,R4,R5, R6,R7,R8
18	3	Resister Package, SIP, 4.7K		Z1,Z3,Z4
19	1	Resister Package, SIP, 4.7K		Z2
20	1	Resistor Package, DIP, 4.7K		U22
21	1	Capacitor, 33 pF, Mica		C34
22	1	Capacitor, 0.01 uF, Mono (ceramic)		C35
23	2	Capacitor, 27 pF, Mica		C7,C8
24	30	Capacitor, 0.1 uF, Mono (ceramic)		C1-C6,C9-C16, C18-C23,C26-C30,C32,C33, C38,C45,C46
25	1	Capacitor, 100 uF, 35 V, Elect		C37
26	1	Capacitor, 2.2 pF		C17
27	1	Capacitor, 1.0 uF, Mono (ceramic)		C31
28	1	Capacitor, 4700 uF, 16 V		C36
29	3	Socket, IC, 20-pin		U1-U3
30	2	Socket, IC, 18-pin		U7-U8

TABLE 6.1. MEK6802D5 PARTS LIST (cont'd)

Item No.	Quan.	Description	Part Number	Reference Designation
31	4	Socket, IC, 24-pin		U11-U14
32	3	Socket, IC, 40-pin		U5,U9,U23
33	3	Mini-jump		J2
34	2	Phone Jack, Enclosed		P2,P3
35	2	Test Point, White		TP1, TP2
36	1	Crystal, 3.579 MHz		Y1
37	1	Waffer Assy, 6		J4
38	7	Waffer Assy, 4		J1-4
39	6	LED, 7-segment	Man 74A	U15-U18, U20-U21
40	1	Hex Keypad		
41	1	Transformer		T1
42	1	Cable Assy, 3 Conductor		
43	1	Terminal Block, 4 Position		TB1
44	1	Voltage Regulator +5 V	7805	VR1
45	2	Diode	1N4002	CR1, CR2
46	1	Jumper, Zero ohm		E2
47	7	Transistor	MPS 2907	Q1-Q7
48	1	Heat Sink		VR1

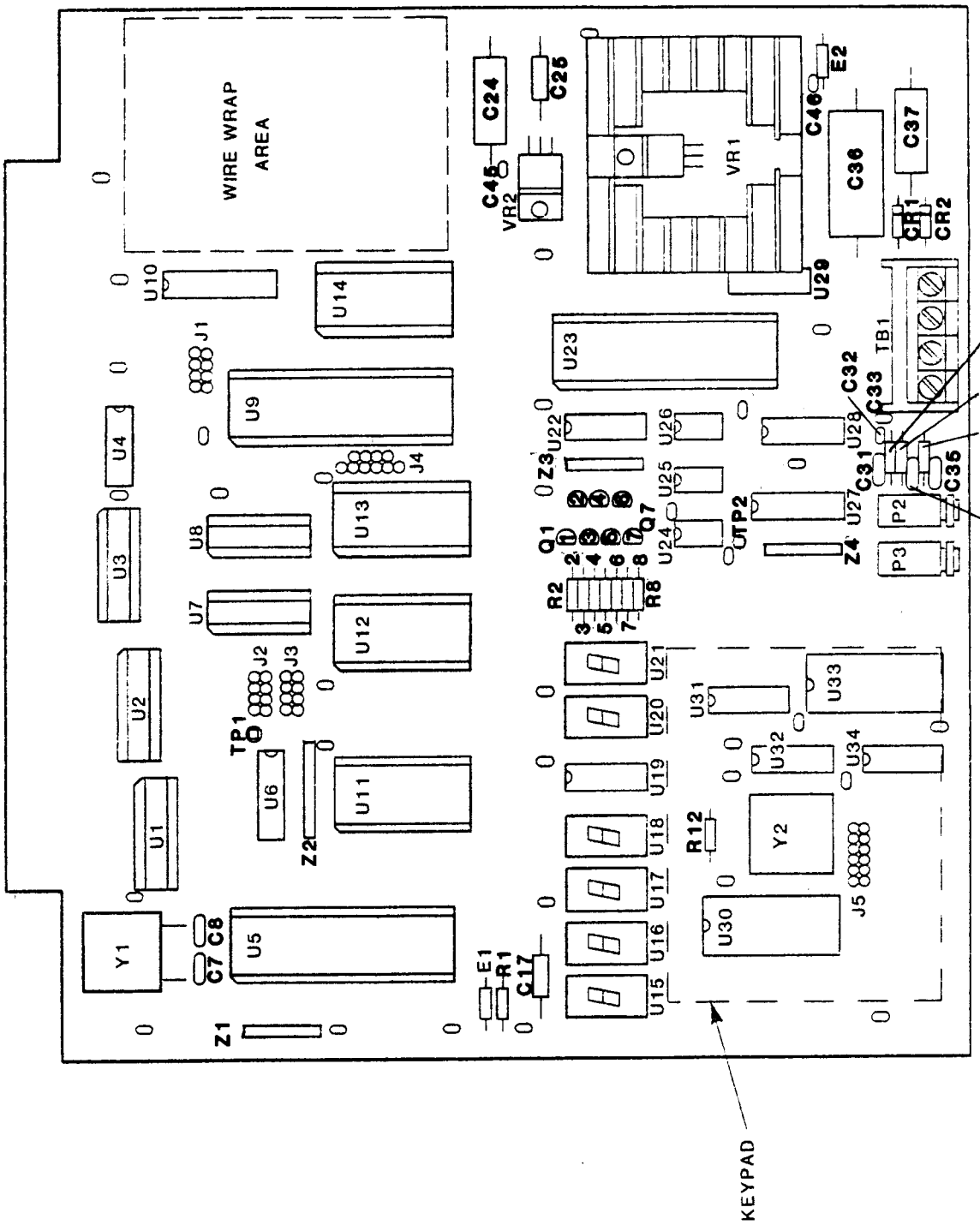


FIGURE 6.1. BOARD OUTLINE AND COMPONENT LAYOUT

CHAPTER 7

SOFTWARE LISTING AND DESCRIPTIONS

7.1 General

This chapter consists of the subroutine descriptions, the D5BUG software listing, and other useful information for the MEK6802D5.

7.2 Subroutine Descriptions

7.2.1 RESET ($F000)

Routine: Hardware Reset (cold start)

Descripton:
Sets interrupt mask
Clears all RAM from $E400-E47F
Initializes system PIA
Establishes default user stack pointer ($E418)
Passes control to PROMPT routine

7.2.2 PROMPT ($F024)

Routine: Software Reset (warm start)

Description:
Initializes system stack pointer ($E47E)
Sets-up software flags:
ROLPAS = 1
UPROG = 0
ROIFLG = 0
KYFLG = 0
FNCFL = 0
Clears 7 -Seg displays
Displays prompt symbol (-)
Establishes FUNSEL as active main program

7.2.3 GET ($F04E)

Routine: Subroutine to read, decode, and debounce a keypad switch.

Effects: All registers are altered by this routine. The A-register has the key code in it on return.

RAM location KYFLG will have $01 in it on return and the code of the key that was depressed will be in location KEY.

7.2.3 GET (cont'd)

Routines Called: DLY25

Stack Requirement: 4 bytes.

Description: The keys of the keypad are electrically arranged into 6 rows and 4 columns. The "RS" (reset) key is not part of this row/column arrangement. GET is called when a key is depressed and the return does not take place until about 25 milliseconds after the key is released. While the key is closed, GET searches the matrix to find the key's row then its column. The Row/Column information is used to address a look-up table from which a key code value is fetched.

This code is stored to the RAM location KEY and is also in the A-register when GET returns. A RAM location called KYFLG is set to $Ø1 to indicate that a key was entered, when another routine recognizes this flag, it should read the KEY code and clear KYFLG, to acknowledge that the key was seen.

Normally the GET routine is serviced through an NMI which results when any key is closed but the interrupt output of the KPD PIA could be disabled allowing GET to work on a polled basis. Polling means that periodically the PIA's control register is read ("Polled") to see if a key was depressed since the last time the program looked.

Calling Routines: NMINT

KEY CODE SUMMARY

KEY	ROW	COL	"CODE"
0	0	0	00
F	0	1	0F
E	0	2	0E
D	0	3	0D
1	1	0	01
2	1	1	02
3	1	2	03
C	1	3	0C
4	2	0	04
5	2	1	05
6	2	2	06
B	2	3	0B
7	3	0	07
8	3	1	08
9	3	2	09
A	3	3	0A
FS	4	0	84
FC	4	1	85
P/L	4	2	86
T/B	4	3	87
MD	5	0	80
EX	5	1	81
RD	5	2	82
GO	5	3	83

FUNCTION KEYS

KEY	"CODE"
MD	80
EX	81
RD	82
GO	83
FS	84
FC	85
P/L	86
T/B	87

NOTE: Function codes have MSB set to allow testing neg cond code

FIGURE 7.1. KEY CODE AND FUNCTION SUMMARY

7.2.4 PUT ($F0BB)

Routine: Utility program to display information on the 7-segment displays and call the active main program as a subroutine once each millisecond.

Effects: The A, B, and X-registers are altered by PUT.

Routines Called: DLY1, routine specified by MNPTR.

Stack Requirement: 1 in addition to main program requirement (Minimum-3 bytes total).

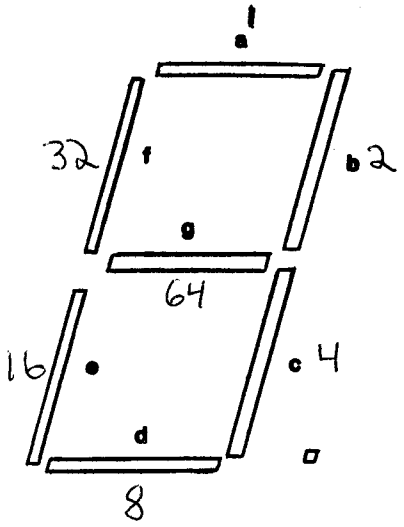
Description: The PUT routine is the heart of the operating system because the displays must be continuously scanned. Anode information for the desired display is presented to the parallel anode lines and the appropriate cathode is driven turning on the proper segments. A given digit is turned on for about one millisecond then the cathode is disabled, the anode information changed to that of the next digit, and the next cathode is enabled.

During the short period between digits a subroutine who's address is stored at RAM location MNPTR is executed. Upon RESET or PROMPT, MNPTR is set to the address of the function select routine (FUNSEL). When FUNSEL wishes to pass control to some other routine, it stores the address of that routine at MNPTR which 'activates' the new routine (and de-activates FUNSEL).

Another way of putting this is to say that the display routine calls 'some program' once per millisecond and 'some program' is the program whose address is stored at RAM location MNPTR.

The PUT routine obtains the segment information from a 6-byte buffer called DISBUF, which contains 7-segment data for the displays with the first byte corresponding to the leftmost display. In addition the B-register keeps track of which digit is to be enabled with the third-from-the-left bit corresponding to the leftmost display and the LSB corresponding to the rightmost display.

NOTICE: The anode interface drivers are ground true so the PUT routine inverts the data in DISBUF before it is stored to the hardware so that the data in DISBUF will be high true.



D.P. not used on MEK6802D5

MSB							LSB
Not Used	g	f	e	d	c	b	a
	64	32	16	8	4	2	1

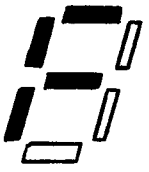
A high (1) in a bit location corresponds to a lighted display segment (for DISBUF only)

7-segment bit relationship to display

EXAMPLE: To display the character "F" the data in DISBUF would be:

a - 1
b - 0
c - 0
d - 0
e - 1
f - 1
g - 1

Not
Used - don't care; make 0



01110001 or $71

FIGURE 7.2. CHARACTER DISPLAY

7.2.5 DYSCOD ($F120)

Routine: Convert hexadecimal data from HEXBUF into 7-segment codes and store in DISBUF.

Effects: The contents of the A, B, and X-registers are saved and restored prior to returning from DYSCOD.

On return the RAM locations DISBUF through DISBUF+5 contain the 7-segment codes for the hexadecimal information in HEXBUF through HEXBUF+2.

Routines Called: ADDAX

Stack Requirement: 9 bytes.

Description: The six nibbles of hexadecimal information from HEXBUF through HEXBUF+2 are each converted to a byte of 7-segment anode information which is stored in the six byte display buffer (DISBUF). The most significant 4 bits of location HEXBUF correspond to the location DISBUF and the least significant 4 bits of HEXBUF+2 correspond to the location DISBUF+5.

The resulting 7-segment information is displayed on the D5's six LED displays by the PUT routine. The first byte of DISBUF corresponds to the leftmost display and the sixth byte (DISBUF+5) corresponds to the rightmost display.

7.2.6 DLY25, DLY1, DLYX ($F169) ($F171) ($F179)

Routine: These routines cause delays of 25 milliseconds, one (1) millisecond, or a variable delay based on the value in the X-register.

Effects: The contents of the X-register are saved and restored prior to returning from the routines. The A and B-registers are not altered by the delay routines.

Stack Requirement: 2 bytes.

Description: Calling DLY25 or DLY1 will cause an execution time delay of about 25 milliseconds or 1 millisecond respectively. Calling DLYX will cause a delay based on the value in the X-register according to the following formula:

$$[(X-1)\ (8\ \text{cyc})\ \ (1.11763\ \text{us/cyc})] \ + \ (27.94\ \text{us})$$

7.2.6 DLY25, DLY1, DLYX (cont'd)

A minimum delay of 27.94 us results from an X-register value of 1 ($0001) and a maximum delay of 0.5842 seconds results from an X-register value of 0 ($0000 - 1 = $FFFF = 65,535).

7.2.7 ADDAX ($F183)

Routine: Add A-register to X-register.

Effects: A and X-registers altered by ADDAX.

Stack Requirement: 2 bytes.

Description: The value in the A-register is added to the value in the X-register and the result is returned in the X-register.

7.2.8 CLRDS ($F195)

Routine: Clear LED display digits per a mask in the A-register.

Effects: The A-register is altered by CLRDS.

The contents of the X-register are saved and restored prior to returning from CLRDS.

Stack Requirement: 2 bytes.

Description: Each of the low order six bits of the A-register correspond to one of the D5's LED displays with the LSB of the A-register corresponding to the rightmost display. For each bit that is a logic 1, the corresponding display buffer location will be cleared resulting in that display goind dark.

Example: 0011 1100 ($3C) would clear the first four LED displays.

7.2.9 ROLL2 ($F1AA)

Routine: Numeric entry routine for 2-digit (1-byte) values.

Requirements: Least significant 4-bits of A-register contain the new hexidecimal digit and the most significant 4-bits must be cleared.

7.2.9 ROLL2 (cont'd)

ROLPAS flag (when set) indicates "first pass" which results in the location being cleared prior to rolling the new digit in from right.

Effects: A-register altered by ROLL2, X-register contents saved and restored prior to return from ROLL2.

Stack Requirement: 2 bytes.

Description: The address of the location to be operated on by ROLL2 is specified in HEXBUF and HEXBUF+1. A new hex digit is passed to ROLL2 in the least significant 4-bits of the A-register. This new digit is shifted in from the right so that the digit which was in the least significant 4-bits of the location is now in the most significant 4-bits. The digit which was in the most significant 4-bits is lost.

If the flag ROLPAS was not equal to zero, then the location is cleared before the new digit is rolled in. ROLPAS is cleared to signal it is no longer the first pass.

7.2.10 ROLL4 ($F1CC)

Routine: Numeric entry routine for 4-digit (2-byte) values.

Requirements: ROLPAS flag (when set) indicates "first pass" which results in the locatins HEXBUF and HEXBUF+1 being cleared prior to rolling the new digit in from the right.

The least significant 4-bits of the A-register contains the new digit and the most significant 4-bits of the A-register must be cleared.

Effects: A-register is altered by ROLL4, B-register is saved and restored prior to returning from ROLL4.

Stack Requirement: 3 bytes.

Description: The locations into which ROLL4 shifts the new digit are HEXBUF and HEXBUF+1. The new digit is passed to ROLL4 in the least significant 4-bits of the A-register. This new digit is shifted into the first two bytes of HEXBUF such that the new digit ends up in the least significant 4-bits of HEXBUF+1. What was in the low 4-bits ends up in the high 4-bits of HEXBUF+1 and what was in the high 4-bits of HEXBUF+1

7.2.10 ROLL4 (cont'd)

ends up in the low 4-bits of HEXBUF. What was in the low 4-bits of HEXBUF ends up in the high 4-bits of HEXBUF, and what was in the high 4-bits of HEXBUF is lost.

If the ROLPAS flag was not equal to zero then HEXBUF and HEXBUF+1 will be cleared prior to shifting in the new digit, and ROLPAS will be cleared to indicate no longer first pass.

7.2.11 RDKEY ($F1EF)

Routine: Read and acknowledge a keypad key.

Requirements: Normally RDKEY would be called because KYFLG was set indicating that a key had been depressed.

Effects: KEY value returned in A-register.

KYFLG cleared to acknowledge that the key has been recognized.

Stack Requirement: 2 bytes.

Description: Load the key code from KEY into the A-register and clear KYFLG to acknowledge that the key has been recognized.

7.2.12 TIN ($F533)

Routine: Read 1 byte of data from tape.

Effects: The A and B-register contents are altered by TIN.

On return the location BYTE ($E459) contains the character that was recovered from tape. The A-register also contains the recovered data.

Routines Called: FEDGE

Stack Requirement: 4 Bytes.

Description: TIN is used to read 1 Kansas City standard format character from cassette tape. The character format is:

a) Logic Ø Bit-Time = 4 cycles of 1200 Hz.

7.2.12 TIN (cont'd)

b) Logic 1 Bit-Time = 8 cycles of 2400 Hz.

c) Characters consist of:

i) 1 Logic Ø Bit-Time as a "Start" bit.
ii) 8 Data Bit-Times (LSB first).
iii) At least 2 logic 1 Bit-Times as "Stop" Bits.

Software timing techniques are used and an error tolerant algorithm is employed to assure reliable data recovery. The TIN routine is in control from before a start bit until just after the eighth data bit in the serial stream.

During the stop bit-times the previous data character is processed by other software. At the 300 Baud data rate the stop bits take 13.3 milliseconds.

When reading successive data characters, the software which processes the previous byte read must finish in time to call TIN at least one full cycle before the next start bit-time. This will allow TIN to regain synchronization with the serial data stream.

The time required from the last cycle of the instruction that calls TIN until the first test for an edge is exactly 47 processor cycles or 52.5 microseconds.

The time from the end of the last cycle of the last data bit-time until the completion of TIN's RTS instruction is 100 cycles or 111.2 microseconds.

Generally it will be easy to process the previous character in time to get back to TIN for the succeeding character.

7.2.13 PNCHB ($F608)

Routine: Format and send one byte of data to the tape interface.

Effects: The contents of the A and B-registers are altered during PNCHB.

Routines Called: BITØ, BIT1, INVRT.

Stack Requirement: 6 bytes.

7.2.13 PNCHB (cont'd)

Description: Refer to the description of TIN for the format of tape characters.

Software timing techniques are used to create a serial stream of cycles of the proper duration to form Kansas City Standard audio signals.

When punching consecutive bytes of data it is vital to maintain timing control between successive bytes, not just within the character.

PNCHB consumes exactly 35 MPU cycles from the last MPU cycle of the calling instruction until the first start bit edge is sent. Timing is then automatically controlled by PNCHB while the character is being transmitted.

After transmitting the second last edge of the last stop bit PNCHB returns. The edge which marks the completion of the last stop bit cycle also marks the beginning of the succeeding start bit for the next character. It is supplied as the first edge transmitted by the next call to PNCHB.

The time allowed from the last cycle of PNCHB's RTS until the next edge must be transmitted is exactly 159 MPU cycles.

159 minus the 35 for entry to PNCHB the next time leaves exactly 124 cycles which must be consumed by external software. For example, if it takes 50 cycles to set up for the next character, the remaining 74 (124-50) cycles must be consummed in dummy delays before calling PNCHB again.

7.2.14 PUNCH ($F630)

Routine: Format and punch an entire file to the cassette.

Effects: The A, B, and X-register contents are altered during PUNCH.

Routines Called: PNCHB

Stack Requirement: 8 Bytes.

Description: PUNCH is a time-tuned subroutine.

7.2.14 PUNCH (cont'd)

Before calling PUNCH store the beginning address of the data to be punched at RAM location BEGAD ($E460,1) and the ending address at ENDAD ($E462,3).

The file format is as follows:

a) 30 seconds of $FF characters as leader.
b) Block start character "S" ($53).
c) Begin address high byte.
d) Begin address low byte.
e) End address high byte.
f) End address low byte.
g) Data in binary form starting with data at Begin address to and including data at End address.
h) One byte checksum. Checksum is the two's complement summation of all data bytes plus the begin and end address characters. When loading, the simple sum of all data in the file starting with the first byte of the begin address and including the checksum character should be zero.

7.2.15 LOAD ($F69C)

Routine: Load or verify a D5 format tape file to memory from cassette.

Effects: The A, B, and X-register contents are altered by LOAD.

Routines Called: TIN.

Stack Requirement: 6 bytes.

Description: The LOAD routine is used to read a tape data file into memory or verify a tape file against the contents of memory. If the least significant bit of the flag FNCFL ($E43E) is set, a load takes place, otherwise a verify is performed.

Normally if a checksum error is detected, an error message is displayed on the 7-segment LED's.

This may be overriden by setting the MSB of FNCFL ($E43E) prior to calling LOAD. When the override feature is used, the Z condition code may be checked upon return from LOAD to determine if the load was acceptable. If Z is true the load was good.

GENERAL NOTE FOR CASSETTE ROUTINES

The cassette routines rely on critical execution time-tuning to work properly.

To use the individual routines outside the context of the D5BUG operating system, take care to abide by the timing requirements of the individual routines.

To assist in this effort, the number of processor cycles required by each instruction is presented as the first character(s) in the comment portion of each listing line for those routines.

7.3 Program Address Descriptions

TABLE 7.1. D5BUG USEFUL PROGRAM ADDRESSES
(Also Refer To Complete D5BUG Listing)

Name of Routine	ADDR in D5BUG	Description
RESET	$F000	Cold Restart
PROMPT	$F024	Warm Start
GET	$F04E	Routine to Read a Key (Uses NMI routine so a user would not normally use this routine directly.
PUT	$F0BB	Display Data on 7-segment readouts and calls functioning subroutines.
DYSCOD	$F120	Decode Hex to 7-segment.
DLY25	$F169	Delay 25 milliseconds.
DLY1	$F171	Delay 1 millisecond.
DLYX	$F179	Delay based on X-Reg (about 1/2 second Max).
ADDAX	$F183	Add A-Register to X-Register.
CLRDS	$F195	Clear display digits per A-Register mask.
ROLL2	$F1AA	Numeric entry routine for 2-digit values (one byte). Address being operated on specified in "HEXBUF".
ROLL4	$F1CC	Numeric entry routine for 4-digit values (2-Bytes).
RDKEY	$F1EF	Read and acknowledge key.
TIN*	$F533	Read 1 byte from tape.
PNCHB*	$F630	Format and punch a whole flie.
LOAD*	$F69C	Load a whole file (if "FNCFL" $\neq$ 0). Verify a whole file (if "FNCFL" = 0).

*Tape routines have critically tuned execution times. Refer to detailed explanations.

7.4 Scratch RAM Descriptions

TABLE 7.2. SUMMARY OF MOST USED D5BUG SCRATCH RAM LOCATIONS
(Also See Complete D5BUG Listing)

Name of Routine	ADDR in D5BUG	Description
MNPTR	$E419,A	Pointer to active sub-program.
KEY	$E41B	Entered key code from keypad.
KYFLG	$E41C	Flag to indicate a key is pending.
DISBUF	$E41D-E422	(6) bytes correspond to (6) 7-segment displays. Contains 7-segment codes.
ROLPAS	$E423	Flag to indicate first digit entry.
HEXBUF	$E42C,D,E	3-byte buffer for hexadecimal information. Each byte corresponds to (2) 7-segment display digits.
USP	$E42F,30	User stack pointer pseudo-register.
UCC	$E431	User condition codes pseudo-register.
UB	$E432	User B-register pseudo-register.
UA	$E433	User A-register pseudo-register.
UX	$E434,5	User X-register pseudo-register.
UPC	$E436,7	User Program Counter pseudo-register.
UIRQV	$E43C,D	Points to user's IRQ service routine.
FNCFL	$E43E	Flag to indicate special (or alternate) function.
FNCPNT	$E43F,40	Points to ADDR of user's special function table.
BYTE	$E459	Data byte read from cassette or to be punched to cassette.
BEGAD	$E460,1	Beginning Address (for punch).
ENDAD	$E462,3	Ending Address (for punch).

*Tape routines have critically tuned execution times. Refer to detailed explanations.

7.5 Hardware Address Information

TABLE 7.3. USEFUL HARDWARE ADDRESS INFORMATION

Item	Description
D5BUG ROM (U12)	$F000 - F7FF but also "mirrors" such that contents shown in the listing as addresses ($F400-F7FF) also appear at addresses ($FC00-FFFF). This mapping is controlled by jumper arrangement J2.
User ROM (U13)	ROM type controlled by J4 jumper arrangement (may be 1K x 8 or 2K x 8; single supply or triple supply; ROM or PROM) Mapping controlled by J3 jumper arrange ment may be mapped in any combination of the 1K address areas ... $E800 - EBFF; $EC00 - EFFF; $F800 - FBFF; or $FC00 - FFFF.
System Scratch RAM	Located at $E400 - E47F. This RAM is further divided into (3) sections: $E400 - E418 provided for user's stack $E419 - E463, D5BUG scratch RAM $E464 - E47F, D5BUG stack
User RAM (in U5)	$0000 - 007F inside MC6802 (U5) available for user. Must be disabled by jumper E1 if expansion buffer (U3) is installed.
Optional RAM (U7, U8)	$E000 - E3FF optional MCM2114 RAMs available for user.
System PIA	$E484 - E487 which further breaks down to: $E484 - Data Port A $E485 - Control Reg A $E486 - Data Port B $E487 - Control Reg B Also incomplete address decoding causes "mirroring" which makes this PIA also appear at addresses which meet this requirement: 1110 01XX 0XXX X0 XX (4 Regs in PIA) where "X's" are don't care. The total number of "images" is (64) with the first image at $E484 - E487 and the last image at $E7F8 - E7FB.

7.5 Hardware Address Information (cont'd)

TABLE 7.3. USEFUL HARDWARE ADDRESS INFORMATION (cont'd)

Item	Description
User PIA	$E480 - E483 which further breaks down to: $E480 - Data Port A $E481 - Control Reg A $E482 - Data Port B $E483 - Control Reg B Also as with the system PIA above, incomplete Addr. decoding is responsible for "mirroring" at all addresses meeting the requirement: 1110 01XX 1XXX X1 XX or a total of (64) "images" with the first image at $E480 - E483 and the last image at $E7FC - E7FF.
ACIA (U33*) *Optional - not used when keypad is present	$E700, E701 which further breaks down to: $E700 Control/status register $E701 Data register Also as with the PIA's above, incomplete address decoding is responsible for "mirroring" at all addresses meeting the requirement: 1110 0111 0XXX XXX X 2 REGS In ACIA or a total of (64) "images" with the first image at $E700, E701 and the last image at $E77E, E77F.
OFF-BOARD	$0000 - $DFFF, this 56K address area is assumed to be external to the MEK6802D5 for expansion purposes. To use this address space, install U1, U2, and U3 buffers and E1 (jumper to disable RAM in MC6802, U5).

7.6 D5BUG Listing

The following is the D5BUG Program Listing. It consists of 35 pages. Refer to paragraph 7.7 for a description of this listing.

```
0001                                NAM     RESET
0002                                OPT     CREF,LLEN=80
0003A F000                          ORG     $F000
0004                        ************************************************************
0005                        *
0006                        * RESET - COLD START ROUTINE
0007                        *
0008                        ************************************************************
0009A F000 01               RESET   NOP             SET INTERRUPT MASK
0010A F001 0F                       SEI             .
0011A F002 CE E3FF  A               LDX     #$E3FF  CLEAR RAM
0012A F005 08               CLRLOP  INX             .
0013A F006 6F 00    A               CLR     0,X     .
0014A F008 8C E487  A               CPX     #$E487  .
0015A F00B 26 F8 F005               BNE     CLRLOP  .
0016A F00D CE E484  A               LDX     #$E484  INITIALIZE SYSTEM PIA
0017A F010 86 7F    A               LDAA    #$7F    .
0018A F012 A7 00    A               STAA    0,X     .
0019A F014 86 FF    A               LDAA    #$FF    .
0020A F016 A7 02    A               STAA    2,X     .
0021A F018 86 06    A               LDAA    #$06    .
0022A F01A A7 01    A               STAA    1,X     .
0023A F01C A7 03    A               STAA    3,X     .
0024A F01E CE E418  A               LDX     #$E418  DEFAULT USER STACK
0025A F021 FF E42F  A               STX     USP     .
0026                        ************************************************
0027                        *
0028                        * PROMPT - ROUTINE TO SET UP PROMPT CONDITIONS
0029                        *
0030                        ************************************************
0031A F024 8E E47E  A       PROMPT  LDS     #STKTOP INIT SYSTEM STACK
0032A F027 86 01    A               LDAA    #1      SET FIRST PASS
0033A F029 B7 E423  A               STAA    ROLPAS  .
0034A F02C 7F E43B  A               CLR     UPROG   INIT FLAGS
0035A F02F 7F E438  A               CLR     ROIFLG  .
0036A F032 7F E41C  A               CLR     KYFLG   .
0037A F035 7F E43E  A               CLR     FNCFL   .
0038A F038 86 40    A               LDAA    #$40    DISPLAY PROMPT
0039A F03A B7 E41D  A               STAA    DISBUF  .
0040A F03D 86 1F    A               LDAA    #%00011111 .
0041A F03F BD F195  A               JSR     CLRDS   .
0042A F042 CE F0E5  A               LDX     #FUNSEL EXECUTE FUNCTION SELECT
0043A F045 FF E419  A               STX     MNPTR   .
0044A F048 BD F7AE  A               JSR     ENNMI   ENABLE NMI
0045A F04B 7E F0BB  A               JMP     PUT     & GO
```

```
0048                    ************************************************************
0049                    *
0050                    * GET - ROUTINE TO READ A KEY
0051                    *
0052                    ************************************************************


0054A F04E CE E484  A GET     LDX    #PIA      POINT AT PIA
0055A F051 86 FF    A         LDAA   #$FF      .
0056A F053 A7 00    A         STAA   KPCOL,X   TO TURN OFF DISPLAYS
0057A F055 86 3F    A         LDAA   #%00111111 COL 0, ALL ROWS
0058A F057 A7 02    A LPCOL   STAA   KPROW,X   STORE INFO TO KEY MATRIX
0059A F059 6D 00    A         TST    KPCOL,X   MSB IS MUX BIT
0060A F05B 2A 06 F063         BPL    COLFND    BIT-7 LOW MEANS COL FOUND
0061A F05D 8B 40    A         ADDA   #$40      INC COL BITS TO MUX
0062A F05F 24 F6 F057         BCC    LPCOL     CONTINUE FOR ALL COLS
0063A F061 20 EB F04E         BRA    GET       KEY BOUNCED, START OVER
0064A F063 84 C0    A COLFND  ANDA   #%11000000 MASK TO SAVE ONLY COL
0065A F065 B7 E41B  A         STAA   KEY       WILL UPDATE LATER; JUST TEMP SAV
0066A F068 C6 20    A         LDAB   #%00100000 ROW 5
0067A F06A 17         LPROW   TBA              COPY ROW INFO TO A-REG
0068A F06B BA E41B  A         ORAA   KEY       COMBINE WITH COL INFO
0069A F06E A7 02    A         STAA   KPROW,X   DRIVE KEY MATRIX
0070A F070 6D 00    A         TST    KPCOL,X   MSB LOW = CLOSURE
0071A F072 2A 05 F079         BPL    ROWFND
0072A F074 54                 LSRB             NEXT LOWER ROW BIT
0073A F075 26 F3 F06A         BNE    LPROW     LOOP TILL ALL ROWS TRIED
0074A F077 20 D5 F04E         BRA    GET       KEY BOUNCED; START OVER
0075A F079 4F         ROWFND  CLRA             PREPARE TO FIND BINARY ROW #
0076A F07A 54         LPFND   LSRB             LOOP BUILDS BINARY ROW #
0077A F07B 25 03 F080         BCS    DUNROW    WHEN BIT FALLS OFF; A-REG HAS #
0078A F07D 4C                 INCA
0079A F07E 20 FA F07A         BRA    LPFND
0080A F080 79 E41B  A DUNROW  ROL    KEY
0081A F083 49                 ROLA
0082A F084 79 E41B  A         ROL    KEY
0083A F087 49                 ROLA             A-REG IS 000RRRCC
0084                    * A-REG NOW CONTAINS OFFSET FOR KEY LOOK-UP
0085A F088 6D 00    A CLOP    TST    KPCOL,X   SEE IF KEY STILL DOWN
0086A F08A 2A FC F088         BPL    CLOP      WAIT TILL LET UP
0087A F08C BD F169  A         JSR    DLY25     DELAY TO DEBOUNCE
0088A F08F CE F0A3  A         LDX    #KYTBL    POINT AT TOP OF TABLE
0089A F092 BD F183  A         JSR    ADDAX     CALC ADDR OF KEY CODE
0090A F095 A6 00    A         LDAA   ,X        GET KEY CODE
0091A F097 B7 E41B  A         STAA   KEY       SAVE KEY VALUE
0092A F09A C6 01    A         LDAB   #1
0093A F09C F7 E41C  A         STAB   KYFLG     INDICATE KEY PENDING
0094A F09F F6 E486  A         LDAB   PIAROW    TO CLEAR NMI
0095A F0A2 39         DIDDLE  RTS              ** RETURN **
0096                    *
0097                    * THIS RTS IS USED AS A DO-NOTHING SUB
0098                    * SO SYST CAN BE DISABLED EXCEPT DISPLAY
0099                    *
```

```
0101                    ************************************************************
0102                    *
0103                    * KYTBL - KEY VALUE TABLE
0104                    *
0105                    ************************************************************
0106A F0A3     00     A KYTBL   FCB     $00       '0' KEY
0107A F0A4     0F     A         FCB     $0F       'F'
0108A F0A5     0E     A         FCB     $0E       'E'
0109A F0A6     0D     A         FCB     $0D       'D'
0110A F0A7     01     A         FCB     $01       '1'
0111A F0A8     02     A         FCB     $02       '2'
0112A F0A9     03     A         FCB     $03       '3'
0113A F0AA     0C     A         FCB     $0C       'C'
0114A F0AB     04     A         FCB     $04       '4'
0115A F0AC     05     A         FCB     $05       '5'
0116A F0AD     06     A         FCB     $06       '6'
0117A F0AE     0B     A         FCB     $0B       'B'
0118A F0AF     07     A         FCB     $07       '7'
0119A F0B0     08     A         FCB     $08       '8'
0120A F0B1     09     A         FCB     $09       '9'
0121A F0B2     0A     A         FCB     $0A       'A'
0122A F0B3     84     A         FCB     $84       'FS' FUNCTION SET
0123A F0B4     85     A         FCB     $85       'FC' FUNCTION CLEAR
0124A F0B5     86     A         FCB     $86       'P/L' PUNCH/LOAD
0125A F0B6     87     A         FCB     $87       'T/B' TRACE/BREAK
0126A F0B7     80     A         FCB     $80       'MD' MEMORY DISPLAY
0127A F0B8     81     A         FCB     $81       'EX' ESCAPE
0128A F0B9     82     A         FCB     $82       'RD' REGISTER DISPLAY
0129A F0BA     83     A         FCB     $83       'GO' GO
```

```
0131                    ************************************************************
0132                    *
0133                    * PUT - DISPLAYS DATA IN DISBUF & CALLS THE
0134                    *       FUNCTIONING SUBROUTINE
0135                    *
0136                    ************************************************************
0137A F0BB C6 20    A PUT     LDAB   #%00100000 INIT DIG ENABLE PATTERN
0138A F0BD CE E41A  A LP1P    LDX    #DISBUF-3 POINT AT DISPLAY BUFFER
0139A F0C0 17                 TBA              MAKE EXTRA COPY
0140A F0C1 08           LP2P  INX              POINT AT NXT DIGIT INFO
0141A F0C2 48                 ASLA             ADD 1 TO 'X' FOR EACH SHIFT
0142A F0C3 24 FC F0C1         BCC    LP2P      LOOP DEVELOPS DIGIT INFO ADDR
0143A F0C5 A6 00    A         LDAA   ,X        GET SEG INFO
0144A F0C7 43                 COMA             ANODE DRIVERS ARE GND TRUE
0145A F0C8 B7 E484  A         STAA   ANOD      STORE ANODE INFO TO PIA
0146A F0CB F7 E486  A         STAB   CATH      ENABLE DIGIT CATHODE
0147A F0CE BD F171  A         JSR    DLY1      ON FOR 1 MILLISECOND
0148A F0D1 86 FF    A         LDAA   #%11111111 1'S TURN OFF SEGS
0149A F0D3 B7 E484  A         STAA   ANOD      TURN OFF ALL SEGS
0150A F0D6 B7 E486  A         STAA   CATH      ENABLE ALL KPD ROWS
0151A F0D9 37                 PSHB             HAS ROTATING DIGIT ENABLE
0152A F0DA FE E419  A         LDX    MNPTR     GET ADDRESS OF ACTIVE MAIN PROG
0153A F0DD AD 00    A         JSR    ,X        EXECUTE IT
0154                    ****
0155                    **** SEE MANUAL
0156                    ****
0157A F0DF 33                 PULB             RECOVER DIGIT ENABLE
0158A F0E0 54                 LSRB             NEXT DIGIT
0159A F0E1 26 DA F0BD         BNE    LP1P      NOT THRU WHOLE CYCLE
0160A F0E3 20 D6 F0BB         BRA    PUT       PAST LAST DIGIT
0161                    *
```

```
                        ************************************************************
0164                    *
0165                    * FUNSEL - ROUTINE TO SELECT A FUNCTION FROM A KEY INPUT
0166                    *
0167                    ************************************************************
0168
0169A F0E5 7D E41C  A FUNSEL TST     KYFLG     KEY PENDING ?
0170A F0E8 26 01 F0EB       BNE     KEYNOW    YES,TEST IT
0171A F0EA 39               RTS               ** RETURN ** NO KEY PENDING
0172                    *
0173A F0EB BD F1EF  A KEYNOW JSR     RDKEY     GET & ACKNOWLEDGE KEY
0174A F0EE 2B 15 F105       BMI     FUNKY     IF FUNCTION KEY
0175A F0F0 7D E43E  A       TST     FNCFL
0176A F0F3 26 0B F100       BNE     UFNK
0177A F0F5 BD F1CC  A       JSR     ROLL4     # ENTRY SO ROLL IT IN
0178A F0F8 BD F120  A       JSR     DYSCOD    CONVERT TO 7-SEG
0179A F0FB 86 03    A       LDAA    #%00000011
0180A F0FD 7E F195  A       JMP     CLRDS     BLANK LAST 2 DIGITS
0181                    *
0182A F100 FE E43F  A UFNK   LDX     FNCPNT    POINT AT USER FUNCTION TABLE
0183A F103 20 03 F108       BRA     HASH      .
0184                    *
0185A F105 CE F110  A FUNKY  LDX     #SYSFNC   POINT AT SYSTEM FUNCTION TBL
0186A F108 48               HASH    ASLA      2 BYTES PER ENTRY
0187A F109 BD F183  A       JSR     ADDAX     DEVELOP POINTER
0188A F10C EE 00    A       LDX     ,X        GET JMP ADDR
0189A F10E 6E 00    A       JMP     ,X        ** GO THERE **
0190                    *
0191                    *
0192A F110    F1F6  A SYSFNC FDB     MEMBEG    'MD'
0193A F112    F024  A        FDB     PROMPT    'EX'
0194A F114    F2CA  A        FDB     REGBEG    'RD'
0195A F116    F6F3  A        FDB     GO        'GO'
0196A F118    F4C5  A        FDB     FSET      'FS'
0197A F11A    F4D1  A        FDB     FCLR      'FC'
0198A F11C    F4D7  A        FDB     TAPBEG    'P/L'
0199A F11E    F388  A        FDB     BRKBEG    'T/B'
0200                    *
0201
```

```
0204                    ******************************************************
0205                    *
0206                    * MISC - MISC ROUTINES
0207                    *
0208                    ******************************************************
0209                    * DECODE HEX TO 7-SEGMENT
0210                    *
0211A F120 36           DYSCOD PSHA             SAVE REGS
0212A F121 37                  PSHB             .
0213A F122 FF E426  A          STX    XSAV1     .
0214A F125 CE E42C  A          LDX    #HEXBUF   POINT AT HEX INFO
0215A F128 A6 00    A LP01     LDAA   ,X        GET HEX BYTE
0216A F12A 16                  TAB              MAKE EXTRA COPY
0217A F12B 54                  LSRB             RIGHT JUSTIFY HIGH NIBBLE
0218A F12C 54                  LSRB             .
0219A F12D 54                  LSRB             .
0220A F12E 54                  LSRB             HIGH ORDER DIGIT IN B-REG
0221A F12F 84 0F    A          ANDA   #$0F      LOW ORDER DIGIT IN A-REG
0222A F131 37                  PSHB             SAVE ON STACK
0223A F132 36                  PSHA             .
0224A F133 08                  INX              NEXT HEX BYTE
0225A F134 8C E42F  A          CPX    #HEXBUF+3 DONE ?
0226A F137 26 EF F128          BNE    LP01      LOOP 3 TIMES
0227A F139 CE E422  A          LDX    #DISBUF+5 LAST DISPLAY BUFFER DIGIT
0228A F13C C6 05    A          LDAB   #5        LOOP INDEX
0229A F13E FF E428  A LP02     STX    XTMP1     SAVE TEMPORARILY
0230A F141 CE F159  A          LDX    #DYSTBL   POINT AT LOOK-UP TABLE
0231A F144 32                  PULA             GET A HEX DIGIT TO CONVERT
0232A F145 BD F183  A          JSR    ADDAX     POINT AT 7-SEG EQUIV
0233A F148 A6 00    A          LDAA   ,X        GET IT
0234A F14A FE E428  A          LDX    XTMP1     RECOVER POINTER TO DISP BUFFER
0235A F14D A7 00    A          STAA   ,X        STORE CONVERTED DIG
0236A F14F 09                  DEX              NEXT DISPLAY POS
0237A F150 5A                  DECB             LOOP INDEX
0238A F151 2A EB F13E          BPL    LP02      CONTINUE FOR 6 DIGITS
0239A F153 FE E426  A          LDX    XSAV1     RECOVER ENTRY STATUS
0240A F156 33                  PULB
0241A F157 32                  PULA
0242A F158 39                  RTS              ** RETURN **
0243                    *
0244                    *
0245A F159    3F    A DYSTBL FCB    %00111111 '0'
0246A F15A    06    A        FCB    %00000110 '1'
0247A F15B    5B    A        FCB    %01011011 '2'
0248A F15C    4F    A        FCB    %01001111 '3'
0249A F15D    66    A        FCB    %01100110 '4'
0250A F15E    6D    A        FCB    %01101101 '5'
0251A F15F    7D    A        FCB    %01111101 '6'
0252A F160    07    A        FCB    %00000111 '7'
0253A F161    7F    A        FCB    %01111111 '8'
0254A F162    67    A        FCB    %01100111 '9'
0255A F163    77    A        FCB    %01110111 'A'
0256A F164    7C    A        FCB    %01111100 'B'
0257A F165    39    A        FCB    %00111001 'C'
0258A F166    5E    A        FCB    %01011110 'D'
0259A F167    79    A        FCB    %01111001 'E'
0260A F168    71    A        FCB    %01110001 'F'
```

```
0262                        *
0263                        * DELAY SUBS
0264                        *
0265A F169 FF E424  A DLY25  STX     XSAVD     SAVE X ENTRY VALUE
0266A F16C CE 0AEA  A       LDX     #2794     25 MS ENTRY POINT
0267A F16F 20 0B F17C       BRA     DLYLP
0268A F171 FF E424  A DLY1  STX     XSAVD     SAVE ENTRY VAL
0269A F174 CE 006D  A       LDX     #109      1 MS COUNT
0270A F177 20 03 F17C       BRA     DLYLP
0271A F179 FF E424  A DLYX  STX     XSAVD     REQUIRED FOR SIMILARITY TO DLY1/25
0272A F17C 09         DLYLP DEX
0273A F17D 26 FD F17C       BNE     DLYLP     LOOP TILL X=0
0274A F17F FE E424  A       LDX     XSAVD     RECOVER ENTRY VALUE
0275A F182 39               RTS               ** RETURN **
0276                        *
0277                        * SUBROUTINE TO ADD X=X+A
0278                        *
0279A F183 FF E424  A ADDAX STX     XSAVD     TO ALLOW CALCS
0280A F186 BB E425  A       ADDA    XSAVD+1   ADD LOW BYTES
0281A F189 B7 E425  A       STAA    XSAVD+1   UPDATE
0282A F18C 24 03 F191       BCC     ARND      IF NO CARRY; YOU'RE DONE
0283A F18E 7C E424  A       INC     XSAVD     ADD CARRY TO HIGH BYTE
0284A F191 FE E424  A ARND  LDX     XSAVD     RESULT TO X-REG
0285A F194 39               RTS               ** RETURN **
0286                        *
0287                        * CLEAR DISPLAY PER A-REG
0288                        *
0289A F195 FF E426  A CLRDS STX     XSAV1     SAVE ENTRY VALUE
0290A F198 CE E422  A       LDX     #DISBUF+5 RIGHTMOST DIGIT
0291A F19B 44         CLRLP LSRA
0292A F19C 24 02 F1A0       BCC     ARNCLR    IF BIT IN A-REG NOT SET
0293A F19E 6F 00    A       CLR     ,X
0294A F1A0 09         ARNCLR DEX              NEXT DISPLAY
0295A F1A1 8C E41C  A       CPX     #DISBUF-1 DONE ?
0296A F1A4 26 F5 F19B       BNE     CLRLP     CONTINUE 6 TIMES
0297A F1A6 FE E426  A       LDX     XSAV1     RECOVER ENTRY VALUE
0298A F1A9 39               RTS               ** RETURN **
0299                        *
0300A F1AA FF E426  A ROLL2 STX     XSAV1     SAVE ENTRY VALUE
0301A F1AD FE E42C  A       LDX     HEXBUF    ADDR TO ROLL
0302A F1B0 7D E423  A       TST     ROLPAS    FIRST PASS ?
0303A F1B3 27 07 F1BC       BEQ     ARNCL2
0304A F1B5 7F E423  A       CLR     ROLPAS    THIS WAS PASS 1
0305A F1B8 6F 00    A       CLR     ,X        CLEAR LOC ON FIRST PASS
0306A F1BA 20 08 F1C4       BRA     R2OUT
0307A F1BC 68 00    A ARNCL2 ASL    ,X
0308A F1BE 68 00    A       ASL     ,X
0309A F1C0 68 00    A       ASL     ,X
0310A F1C2 68 00    A       ASL     ,X        SHIFT ROLL BYTE 4 PLACES
0311A F1C4 AA 00    A R2OUT ORAA    ,X        COMBINE NEW DATA
0312A F1C6 A7 00    A       STAA    ,X        UPDATE LOC
0313A F1C8 FE E426  A       LDX     XSAV1     RECOVER ENTRY VAL
0314A F1CB 39               RTS               ** RETURN **
```

```
0316                        *
0317                        * ROLL 4 HEX INTO HEXBUF
0318                        *
0319A F1CC 37               ROLL4   PSHB              SAVE ENTRY VALUES
0320A F1CD 7D E423  A               TST     ROLPAS    PASS 1 ?
0321A F1D0 27 0B F1DD               BEQ     ARNCL4    NO,CONTINUE
0322A F1D2 7F E423  A               CLR     ROLPAS    YES,CLEAR FIRST PASS FLAG &
0323A F1D5 7F E42C  A               CLR     HEXBUF    CLR FIRST 4 DIGITS ON FIRST PASS
0324A F1D8 B7 E42D  A               STAA    HEXBUF+1  THEN PUT NEW DATA IN 4TH
0325A F1DB 20 10 F1ED               BRA     R4OUT     .
0326A F1DD 48               ARNCL4  ASLA              LEFT JUSTIFY NEW DIGIT
0327A F1DE 48                       ASLA              .
0328A F1DF 48                       ASLA              .
0329A F1E0 48                       ASLA              .
0330A F1E1 C6 03    A               LDAB    #3        LOOP INDEX
0331A F1E3 49               RO4LP   ROLA              ROLLA INTO HEXBUF
0332A F1E4 79 E42D  A               ROL     HEXBUF+1  .
0333A F1E7 79 E42C  A               ROL     HEXBUF    .
0334A F1EA 5A                       DECB              .
0335A F1EB 2A F6 F1E3               BPL     RO4LP     .
0336A F1ED 33               R4OUT   PULB              RECOVER B-REG
0337A F1EE 39                       RTS               ** RETURN **
0338                        *
0339A F1EF 7F E41C  A RDKEY CLR     KYFLG     READ & ACKNOWLEDGE KEY
0340A F1F2 B6 E41B  A               LDAA    KEY       .
0341A F1F5 39                       RTS               .
```

```
0344                    ************************************************************
0345                    *
0346                    * MEMCH - MEMORY CHANGE/DISPLAY/OFFSET ROUTINE
0347                    *
0348                    ************************************************************
0349A F1F6 CE F205  A MEMBEG LDX     #MEMCH
0350A F1F9 FF E419  A        STX     MNPTR     INIT MAIN POINTER
0351A F1FC 7F E43E  A        CLR     FNCFL     SET FUNCTION FLAG TO ZERO
0352A F1FF FE E42C  A        LDX     HEXBUF    POINT AT ADDR TO DISPLAY
0353A F202 7E F2BA  A        JMP     NEWMEM    EXIT TO UPDATE DISPLAY
0354                    *
0355A F205 7D E41C  A MEMCH  TST     KYFLG     SEE IF ANY KEY PENDING
0356A F208 26 01 F20B        BNE     MEMNOW
0357A F20A 39                RTS               ** RETURN **
0358                    *
0359A F20B BD F1EF  A MEMNOW JSR     RDKEY     GET & ACKNOWLEDGE KEY
0360A F20E FE E42C  A        LDX     HEXBUF    SAVES STEPS LATER
0361A F211 F6 E43E  A        LDAB    FNCFL     SEE IF IN OFFSET MODE
0362A F214 27 77 F28D        BEQ     NORMAL    (NOT OFFSET MODE)
0363A F216 2B 53 F26B        BMI     CALDUN    IF OFFSET CALC FINISHED
0364A F218 4D                TSTA              CHECK KEY
0365A F219 2B 0D F228        BMI     OFFUN     IF FUNCTION KEY
0366A F21B BD F1CC  A        JSR     ROLL4     ENTER NUMBER KEY
0367A F21E BD F120  A OFFOUT JSR     DYSCOD    CONVERT TO 7-SEG
0368A F221 CE 0077  A OFFEND LDX     #$0077    "A"
0369A F224 FF E421  A        STX     DISBUF+4  STORE TO LAST DIGITS
0370A F227 39         OFFRET RTS               ** RETURN **
0371                    *
0372A F228 81 83    A OFFUN  CMPA    #$83      'GO' ?
0373A F22A 26 FB F227        BNE     OFFRET    IF NOT; EXIT
0374A F22C FE E42C  A        LDX     HEXBUF    GET DESTINATION OF BRANCH
0375A F22F 09                DEX               ADJ INSTEAD OF ADJ'ING THE SOURCE
0376A F230 FF E42C  A        STX     HEXBUF    UPDATE
0377A F233 B6 E42D  A        LDAA    HEXBUF+1  LOW BYTE OF DESTINATION
0378A F236 F6 E42C  A        LDAB    HEXBUF    HI BYTE
0379A F239 B0 E42B  A        SUBA    MEMSAV+1  SUBTRACT LOW BYTES
0380A F23C F2 E42A  A        SBCB    MEMSAV    SUBTRACT W/ CARRY
0381A F23F 4D                TSTA              CHECH POLARITY OF LOW ORDER RESULT
0382A F240 2A 01 F243        BPL     ARNINC    IF LO POS DON'T INC HI
0383A F242 5C                INCB              IF LOW WAS NEG INC HI  $FF - $00
0384A F243 5D         ARNINC TSTB              IF B NOW ZERO; OFFSET IS IN RANGE
0385A F244 26 11 F257        BNE     BADOFF    IF NOT; TOO FAR
0386A F246 B7 E42E  A        STAA    HEXBUF+2  SAVE RESULT
0387A F249 BD F120  A        JSR     DYSCOD    CONVERT TO 7-SEG
0388A F24C 86 3C    A        LDAA    #%00111100 CLEAR FIRST 4 DISPLAYS
0389A F24E BD F195  A        JSR     CLRDS
0390A F251 86 80    A        LDAA    #$80
0391A F253 B7 E43E  A        STAA    FNCFL     INDICATE CALC DONE; & OK
0392A F256 39                RTS               ** RETURN **
0393                    *
0394A F257 CE BAD0  A BADOFF LDX     #$BAD0
0395A F25A FF E42C  A        STX     HEXBUF
0396A F25D BD F120  A        JSR     DYSCOD    WRITE "BAD" IN FIRST 3 DISPLAYS
0397A F260 86 07    A        LDAA    #%00000111
0398A F262 BD F195  A        JSR     CLRDS     CLEAR UNUSED DIGITS
0399A F265 86 FF    A        LDAA    #$FF
0400A F267 B7 E43E  A        STAA    FNCFL     INDICATE OFFSET NOT VALID
0401A F26A 39                RTS               ** RETURN **
```

```
0402                        *
0403A F26B 5C               CALDUN INCB             IF IT WAS $FF IT'S NOW 0
0404A F26C 27 13 F281              BEQ     BADCAL   OFFSET WAS BAD
0405A F26E FE E42A  A              LDX     MEMSAV   RECOVER MEM ADDR
0406A F271 81 85    A              CMPA    #$85     FUNCTION CLEAR KEY ?
0407A F273 27 13 F288              BEQ     MEMBAK   YES,DONT SAVE OFFSET
0408A F275 81 83    A              CMPA    #$83     'GO' ?
0409A F277 26 AE F227              BNE     OFFRET   'GO' IS ONLY VALID KEY HERE
0410A F279 B6 E42E  A              LDAA    HEXBUF+2 GET CALC'D OFFSET
0411A F27C A7 00    A              STAA    ,X       STORE TO MEM
0412A F27E 08                      INX              ADV TO NEXT MEM ADDR
0413A F27F 20 07 F288              BRA     MEMBAK   BACK TO MEM CHANGE
0414                        *
0415A F281 81 80    A       BADCAL CMPA    #$80     'MD' ?
0416A F283 26 A2 F227              BNE     OFFRET   'MD' IS THE ONLY VALID KEY HERE
0417A F285 FE E42A  A              LDX     MEMSAV   RECOVER MEM ADDRESS
0418A F288 7F E43E  A       MEMBAK CLR     FNCFL    SIGNAL NOT IN OFFSET MODE
0419A F28B 20 2D F2BA              BRA     NEWMEM   RE-ENTER MEM CHANGE
0420                        *
0421A F28D 4D               NORMAL TSTA             SET COND CODES
0422A F28E 2A 25 F2B5              BPL     NUM      IF NUMBER KEY
0423A F290 81 80    A              CMPA    #$80     'MD' ?
0424A F292 26 03 F297              BNE     NXM1     NO,CHECK FOR "GO"
0425A F294 09                      DEX              YES,BACK UP
0426A F295 20 23 F2BA              BRA     NEWMEM   .
0427                        *
0428A F297 81 83    A       NXM1   CMPA    #$83     'GO' ?
0429A F299 26 03 F29E              BNE     NXM2     NO,CHECK FOR "FS"
0430A F29B 08                      INX              YES,ADVANCE
0431A F29C 20 1C F2BA              BRA     NEWMEM   .
0432A F29E 81 84    A       NXM2   CMPA    #$84     'FS' ?
0433A F2A0 26 1D F2BF              BNE     MEMOUT   NO MORE VALID KEYS
0434A F2A2 86 3F    A              LDAA    #%00111111
0435A F2A4 BD F195  A              JSR     CLRDS    .
0436A F2A7 86 01    A              LDAA    #1       .
0437A F2A9 B7 E43E  A              STAA    FNCFL    SET OFFSET MODE
0438A F2AC B7 E423  A              STAA    ROLPAS   SET FIRST PASS
0439A F2AF FF E42A  A              STX     MEMSAV   SAVE MEM CHG POINTER
0440A F2B2 7E F221  A              JMP     OFFEND
0441                        *
0442A F2B5 BD F1AA  A       NUM    JSR     ROLL2    ENTER NEW DIGIT
0443A F2B8 20 05 F2BF              BRA     MEMOUT   DON'T SET FIRST PASS
0444                        *
0445A F2BA 86 01    A       NEWMEM LDAA    #1
0446A F2BC B7 E423  A              STAA    ROLPAS   SET FIRST PASS FLAG
0447                        *
0448A F2BF A6 00    A       MEMOUT LDAA    ,X       GET DATA TO DISPLAY
0449A F2C1 B7 E42E  A              STAA    HEXBUF+2 UPDATE HEX BUFFER
0450A F2C4 FF E42C  A              STX     HEXBUF   UPDATE ADDR
0451A F2C7 7E F120  A              JMP     DYSCOD   CONV TO 7-SEG
0452                        *
```

```
0456                    ********************************************************
0457                    *
0458                    * REGDIS - REGISTER DISPLAY/CHANGE ROUTINE
0459                    *
0460                    ********************************************************
0461                    *
0462A F2CA 7D E43E  A REGBEG TST     FNCFL     SEE IF IN VERIFY
0463A F2CD 27 06 F2D5        BEQ     NOTVRF
0464A F2CF 7F E43E  A        CLR     FNCFL     SIGNAL VERIFY
0465A F2D2 7E F4DC  A        JMP     LDTAP     GO VERIFY TAPE
0466A F2D5 CE F2EE  A NOTVRF LDX     #REGDIS
0467A F2D8 FF E419  A        STX     MNPTR     INIT MAIN POINTER
0468A F2DB CE F0BB  A        LDX     #PUT      SET SO RTS...
0469A F2DE FF E47D  A        STX     STKTOP-1  WILL BE TO PUT
0470A F2E1 8E E47C  A        LDS     #STKTOP-2 INIT STACKPOINTER
0471A F2E4 7F E441  A        CLR     REGNO     INIT REG # = UPC
0472A F2E7 86 01    A        LDAA    #1
0473A F2E9 B7 E423  A        STAA    ROLPAS    INDICATE FIRST PASS
0474A F2EC 20 40 F32E        BRA     REGOUT    TO UPDATE DISPLAY
0475                    *
0476A F2EE 7D E41C  A REGDIS TST     KYFLG     SEE IF ANY KEY PENDING
0477A F2F1 26 01 F2F4        BNE     REGNOW
0478A F2F3 39                RTS               ** RETURN ** NO KEY
0479                    *
0480A F2F4 BD F1EF  A REGNOW JSR     RDKEY     GET & ACKNOWLEDGE KEY
0481A F2F7 2B 05 F2FE        BMI     REGFNC    IF FUNCTION KEY
0482A F2F9 BD F1CC  A        JSR     ROLL4
0483A F2FC 20 30 F32E        BRA     REGOUT    UPDATE DISPLAY & EXIT
0484                    *
0485A F2FE 81 80    A REGFNC CMPA    #$80      'MD' ?
0486A F300 26 0D F30F        BNE     NXR1
0487A F302 B6 E441  A        LDAA    REGNO
0488A F305 4A                DECA
0489A F306 2A 02 F30A        BPL     ARNR1
0490A F308 86 05    A        LDAA    #5        WRAP AROUND
0491A F30A B7 E441  A ARNR1  STAA    REGNO     UPDATE
0492A F30D 20 10 F31F        BRA     NEWREG    SET UP NEW REG ON EXIT
0493                    *
0494A F30F 81 83    A NXR1   CMPA    #$83      'GO'
0495A F311 26 11 F324        BNE     RUNONE    IGNORE INVALID ENTRY
0496A F313 B6 E441  A        LDAA    REGNO
0497A F316 4C                INCA
0498A F317 81 06    A        CMPA    #6        PAST ?
0499A F319 26 01 F31C        BNE     ARNR2
0500A F31B 4F                CLRA              WRAP AROUND
0501A F31C B7 E441  A ARNR2  STAA    REGNO     UPDATE
0502A F31F 86 01    A NEWREG LDAA    #1
0503A F321 B7 E423  A        STAA    ROLPAS
0504                    *
0505A F324 81 87    A RUNONE CMPA    #$87      T/B KEY ?
0506A F326 26 06 F32E        BNE     REGOUT    NO,RETURN
0507A F328 CE F2CA  A        LDX     #REGBEG   YES,SET UP RETURN ADDR
0508A F32B 7E F701  A        JMP     ROI       .
0509                    *
0510A F32E B6 E441  A REGOUT LDAA    REGNO
0511A F331 48                ASLA
0512A F332 48                ASLA              4-BYTES PER BLOCK ENTRY
0513A F333 CE F370  A        LDX     #REGTBL   TOP OF INFO TABLE
```

```
0514A F336 BD F183  A          JSR    ADDAX      POINT AT TABLE ENTRY
0515A F339 A6 03    A          LDAA   3,X        GET 7-SEG INFO
0516A F33B 36                  PSHA              SAVE ON STACK
0517A F33C A6 02    A          LDAA   2,X        .
0518A F33E 36                  PSHA              .
0519A F33F EE 00    A          LDX    ,X         GET ADDR OF DESIRED REG
0520A F341 7D E423  A          TST    ROLPAS     SEE IF NEW REG
0521A F344 27 0A F350          BEQ    NOTNEW
0522A F346 A6 00    A          LDAA   ,X         STORE CURRENT VAL TO DISPLAY
0523A F348 B7 E42C  A          STAA   HEXBUF     .
0524A F34B A6 01    A          LDAA   1,X        .
0525A F34D B7 E42D  A          STAA   HEXBUF+1   .
0526                      *
0527A F350 BD F120  A NOTNEW   JSR    DYSCOD     TO CONVERT TO 7-SEG
0528A F353 32                  PULA              RECOVER DISPLAY CODES
0529A F354 B7 E421  A          STAA   DISBUF+4   & STORE TO DISP BUFFER
0530A F357 32                  PULA              .
0531A F358 B7 E422  A          STAA   DISBUF+5   .
0532A F35B 2A 08 F365          BPL    ARNR3      .
0533A F35D 7F E41D  A          CLR    DISBUF     CLEAR UNUSED DISPLAYS
0534A F360 7F E41E  A          CLR    DISBUF+1   .
0535A F363 20 05 F36A          BRA    ONLY1      .
0536A F365 B6 E42C  A ARNR3    LDAA   HEXBUF     UPDATE HIGH OF PSEUDO REG
0537A F368 A7 00    A          STAA   ,X         .
0538A F36A B6 E42D  A ONLY1    LDAA   HEXBUF+1   .
0539A F36D A7 01    A          STAA   1,X        UPDATE LOW BYTE
0540A F36F 39                  RTS               ** RETURN **
0541                      *
0542                      *
0543A F370    E436  A REGTBL   FDB    UPC
0544A F372    73    A          FCB    %01110011,%00111001
0545                      *
0546A F374    E432  A          FDB    UA-1
0547A F376    00    A          FCB    %00000000,%11110111
0548                      *
0549A F378    E431  A          FDB    UB-1
0550A F37A    00    A          FCB    %00000000,%11111100
0551                      *
0552A F37C    E434  A          FDB    UX
0553A F37E    06    A          FCB    %00000110,%01011110
0554                      *
0555A F380    E42F  A          FDB    USP
0556A F382    6D    A          FCB    %01101101,%01110011
0557                      *
0558A F384    E430  A          FDB    UCC-1
0559A F386    39    A          FCB    %00111001,%10111001
0560                      *
```

```
0564                    ************************************************************
0565                    *
0566                    * BRKBEG - BREAKPOINT EDITOR
0567                    *
0568                    ************************************************************
0569A F388 7D E43E  A BRKBEG TST     FNCFL     FUNCTION FLAG SET ?
0570A F38B 26 01 F38E       BNE     BRKEDT    YES,EDIT BREAKPIONTS
0571A F38D 39               RTS               NO,TAKE NO ACTION
0572A F38E CE F39C  A BRKEDT LDX     #BRKPNT   SET MNPTR WITH BREAKPOINT ROUTINE
0573A F391 FF E419  A        STX     MNPTR     .
0574A F394 86 01    A        LDAA    #$01      SET UP FOR ADDR INPUT
0575A F396 B7 E423  A        STAA    ROLPAS    .
0576A F399 7E F3C0  A        JMP     DISBRK    DISPLAY NEXT BKPT
0577A F39C 7D E41C  A BRKPNT TST     KYFLG     KEY PENDING ?
0578A F39F 26 01 F3A2       BNE     BRKTST    YES,DECODE KEY ?
0579A F3A1 39               RTS               NO,RETURN TO PUT
0580A F3A2 BD F1EF  A BRKTST JSR     RDKEY     GET & ACKNOWLEDGE KEY
0581A F3A5 81 0F    A        CMPA    #$0F      HEX ?
0582A F3A7 22 06 F3AF       BHI     NOTHEX    NO,CHECK FOR FUNCTION
0583A F3A9 BD F1CC  A        JSR     ROLL4     YES, ROLL INTO HEXBUF
0584A F3AC 7E F120  A        JMP     DYSCOD    DISPLAY  & RETURN TO PUT
0585A F3AF 81 84    A NOTHEX CMPA    #$84      FS KEY ?
0586A F3B1 26 02 F3B5       BNE     CKFC      NO,TRY FC
0587A F3B3 20 3C F3F1       BRA     BKTOTB    YES,ENTER AS BKPT & RETURN
0588A F3B5 81 85    A CKFC   CMPA    #$85      FC KEY ?
0589A F3B7 26 03 F3BC       BNE     CKGO      NO,CHECK FOR GO
0590A F3B9 7E F42E  A        JMP     BKFMTB    YES,REMOVE A BKPT
0591A F3BC 81 83    A CKGO   CMPA    #$83      GO KEY ?
0592A F3BE 26 30 F3F0       BNE     DISDUN    YES,DISPLAY NEXT BKPT & RETURN
0593                    *
0594                    * DISBRK - DISPLAY NEXT BREAKPOINT
0595                    *
0596A F3C0 B6 E444  A DISBRK LDAA    BRKNO     GET # INTO HEXBUF
0597A F3C3 B7 E42E  A        STAA    HEXBUF+2  ANY BREAKPOINTS ?
0598A F3C6 27 1B F3E3       BEQ     BACK      NO,RETURN
0599A F3C8 FE E442  A        LDX     BKPNTR    YES,DISPLAY NEXT ONE
0600A F3CB 08          BKLOOP INX               .
0601A F3CC 08                 INX               .
0602A F3CD 08                 INX               .
0603A F3CE 08                 INX               .
0604A F3CF 8C E459  A         CPX     #BRKEND   END OF TAB
0605A F3D2 26 03 F3D7       BNE     NOTEND    NO,GO TEST FOR BKPT
0606A F3D4 CE E445  A        LDX     #BRKTAB   YES,WRAP AROUND
0607A F3D7 6D 03    A NOTEND TST     3,X       BREAKPOINT ?
0608A F3D9 27 F0 F3CB       BEQ     BKLOOP    NO,TRY NEXT LOC
0609A F3DB FF E442  A        STX     BKPNTR    YES,MOVE POINTER
0610A F3DE EE 00    A        LDX     0,X       GET BKPT ADDR
0611A F3E0 FF E42C  A        STX     HEXBUF    & DISPLAY IT
0612A F3E3 BD F120  A BACK   JSR     DYSCOD    .
0613A F3E6 7D E444  A        TST     BRKNO     ANY BREAKPOINTS ?
0614A F3E9 26 05 F3F0       BNE     DISDUN    YES,RETURN
0615A F3EB 86 FE    A        LDAA    #$FE      MASK ALL BUT LSD
0616A F3ED BD F195  A        JSR     CLRDS     .
0617A F3F0 39          DISDUN RTS               RETURN TO PUT
```

```
0619                        *
0620                        * BKTOTB-ENTER A BREAKPOINT FROM HEXBUF INTO
0621                        *        THE TABLE & UPDATE BRKNO
0622                        *
0623A F3F1 BD F4A0  A BKTOTB JSR     FNDBRK    BREAKPOINT EXIST ?
0624A F3F4 25 32 F428       BCS     FULL      YES,RETURN
0625A F3F6 8D 46 F43E       BSR     BKNO      FIND OPEN SPACE
0626A F3F8 B6 E444  A       LDAA    BRKNO     GET # OF BREAKPOINTS
0627A F3FB 81 05    A       CMPA    #$05      FULL ?
0628A F3FD 2C 29 F428       BGE     FULL      YES
0629                      * CHECK FOR RAM
0630A F3FF FE E42C  A       LDX     HEXBUF    TEST FOR RAM
0631A F402 A6 00    A       LDAA    0,X       .
0632A F404 43               COMA              .
0633A F405 63 00    A       COM     0,X       .
0634A F407 A1 00    A       CMPA    0,X       RAM ?
0635A F409 26 1D F428       BNE     FULL      NO ,RETURN
0636A F40B 43               COMA              YES, RESTORE DATA
0637A F40C A7 00    A       STAA    0,X       .
0638                      * ENTER BKPT INTO TABLE
0639A F40E FE E442  A       LDX     BKPNTR    POINT INTO BREAKPOINT TAB
0640A F411 A7 02    A       STAA    2,X       SAV OPCODE
0641A F413 B6 E42C  A       LDAA    HEXBUF    GET OP CODE ADDR
0642A F416 F6 E42D  A       LDAB    HEXBUF+1  .
0643A F419 A7 00    A       STAA    0,X       INSERT BREAKPOINT
0644A F41B E7 01    A       STAB    1,X       .
0645A F41D 7C E444  A       INC     BRKNO     COUNT BREAKPOINT
0646A F420 6C 03    A       INC     3,X       FLAG BRAKPOINT
0647A F422 7C E42E  A       INC     HEXBUF+2  UPDATE BKPT NO.
0648A F425 BD F120  A       JSR     DYSCOD    .
0649A F428 86 01    A FULL  LDAA    #$01      RESET ROLPAS
0650A F42A B7 E423  A       STAA    ROLPAS    .
0651A F42D 39               RTS               & RETURN
0652                        *
0653                        * BKFMTB - REMOVE A BREAKPOINT FROM BUFFER
0654                        *          & UPDATE BRKNO
0655                        *
0656A F42E 8D 70 F4A0 BKFMTB BSR    FNDBRK    BKPT (DISBUF) IN TABLE ?
0657A F430 24 8E F3C0       BCC     DISBRK    NO , RETURN
0658A F432 FE E442  A       LDX     BKPNTR    YES , GET ITS ADDR
0659A F435 6F 03    A       CLR     3,X       & REMOVE IT.
0660A F437 6F 02    A       CLR     2,X       REMOVE OP CODE
0661A F439 7A E444  A       DEC     BRKNO     UPDATE COUNT
0662A F43C 20 82 F3C0       BRA     DISBRK    DISPLAY BKPT & RETURN
0663                        *
0664                        * BKNO - FIND NUMBER OF BREAKPOINTS, UPDATE BRKNO
0665                        *        & PUT ADDR OF LAST OPEN SPACE INTO BKPNTR
0666                        *
0667A F43E 7F E444  A BKNO  CLR     BRKNO
0668A F441 CE E445  A       LDX     #BRKTAB
0669A F444 6D 03    A BKLOP TST     3,X       BREAKPOINT HERE ?
0670A F446 27 05 F44D       BEQ     NEXT1     NO, TRY NEXT ENTRY
0671A F448 7C E444  A       INC     BRKNO     YES,COUNT IT
0672A F44B 20 03 F450       BRA     ISBKPT    SO DONT SAVE ADDR
0673A F44D FF E442  A NEXT1 STX     BKPNTR    & SAVE ADDR
0674A F450 08         ISBKPT INX              POINT TO NEXT ENTRY
0675A F451 08               INX               .
0676A F452 08               INX               .
```

```
0677A F453 08                INX             .
0678A F454 8C E459  A        CPX     #BRKEND DONE ?
0679A F457 26 EB F444        BNE     BKLOP   NO,CONTINUE
0680A F459 86 01    A        LDAA    #$01    RESET ROLPAS
0681A F45B B7 E423  A        STAA    ROLPAS  .
0682A F45E 39                RTS             YES
0683                 ********************************************************
0684                 *
0685                 * INBKS - INSERT BREAKPOINTS FROM TABLE TO MEM
0686                 *
0687                 ********************************************************
0688A F45F 7D E444  A INBKS  TST     BRKNO   BREAKPOINTS ?
0689A F462 27 20 F484        BEQ     NOBPT   NO,RETURN
0690A F464 CE E445  A        LDX     #BRKTAB YES,INSTALL'EM
0691A F467 6D 03    A CKBKPT TST     3,X     BREAKPOINT ?
0692A F469 27 10 F47B        BEQ     NEXT2   NO,TRY NEXT ENTRY
0693                 * INSTALL THE BREAKPOINT
0694A F46B FF E442  A        STX     BKPNTR  SAVE X
0695A F46E 86 3F    A        LDAA    #$3F    SWI
0696A F470 EE 00    A        LDX     0,X     GET ADDR
0697A F472 E6 00    A        LDAB    0,X     GET OP CODE
0698A F474 A7 00    A        STAA    0,X     STORE SWI
0699A F476 FE E442  A        LDX     BKPNTR  RESTORE X
0700A F479 E7 02    A        STAB    2,X     SAVE OPCODE
0701                 * NEXT ENTRY
0702A F47B 08        NEXT2   INX             .
0703A F47C 08                INX             .
0704A F47D 08                INX             .
0705A F47E 08                INX             .
0706A F47F 8C E459  A        CPX     #BRKEND DONE ?
0707A F482 26 E3 F467        BNE     CKBKPT  NO,CONTINUE
0708A F484 39        NOBPT   RTS
0709                 ********************************************************
0710                 *
0711                 * OUTBKS - REMOVE BREAKPOINTS FROM MEM
0712                 *
0713                 ********************************************************
0714A F485 CE E445  A OUTBKS LDX     #BRKTAB POINT TO BREAKPOINT TAB
0715A F488 A6 02    A REMOV1 LDAA    2,X     OP CODE ?
0716A F48A 27 0A F496        BEQ     NEXT3   NO,TRY NEXT ENTRY
0717                 * REMOVE BREAKPOINT FROM RAM
0718A F48C FF E442  A        STX     BKPNTR  SAVE X
0719A F48F EE 00    A        LDX     0,X     GET MEM ADDR
0720A F491 A7 00    A        STAA    0,X     INSERT OPCODE
0721A F493 FE E442  A        LDX     BKPNTR  RESTORE X
0722                 * NEXT ENTRY
0723A F496 08        NEXT3   INX             .
0724A F497 08                INX             .
0725A F498 08                INX             .
0726A F499 08                INX             .
0727A F49A 8C E459  A        CPX     #BRKEND DONE ?
0728A F49D 26 E9 F488        BNE     REMOV1  NO,CONTINUE
0729A F49F 39                RTS             YES,RETURN
```

```
0731                        *
0732                        * FDBRK - FIND BREAKPOINT (HEXBUF) IN BRKTAB
0733                        *        BKPNTR POINTS AT BREAKPOINT & CARRY
0734                        *        IS SET IF BREKPOINT EXISTS,ELSE C IS ="0"
0735                        *
0736A F4A0 B6 E42C  A FNDBRK LDAA   HEXBUF    BREAKPOINT MSB
0737A F4A3 F6 E42D  A        LDAB   HEXBUF+1  BREAKPOINT LSB
0738A F4A6 CE E445  A        LDX    #BRKTAB   BREAKPOINT TAB
0739A F4A9 A1 00    A BRKLOP CMPA   0,X       MATCH ?
0740A F4AB 27 0B F4B8        BEQ    CKLSB     YES
0741A F4AD 08           NEXT INX               NO POINT TO NEXT
0742A F4AE 08                INX               .
0743A F4AF 08                INX               .
0744A F4B0 08                INX               .
0745A F4B1 8C E459  A        CPX    #BRKEND   DONE ?
0746A F4B4 26 F3 F4A9        BNE    BRKLOP    NO,CONTINUE
0747A F4B6 0C                CLC               YES,BUT NO BKPT
0748A F4B7 39                RTS
0749A F4B8 E1 01    A CKLSB  CMPB   1,X       MATCH ?
0750A F4BA 26 F1 F4AD        BNE    NEXT      NO,TRY NEXT ENTRY
0751A F4BC 6D 03    A        TST    3,X       BREAKPOINT ACTIVE ?
0752A F4BE 27 ED F4AD        BEQ    NEXT      NO,TRY AGAIN
0753A F4C0 0D                SEC               YES,FOUND IT
0754A F4C1 FF E442  A        STX    BKPNTR    SAVE ADDR
0755A F4C4 39                RTS
```

```
0758                    ************************************************************
0759                    *
0760                    * FSET - SET FUNCTION FLAG & DISPLAY "FS"
0761                    *
0762                    ************************************************************
0763A F4C5 86 01     A FSET    LDAA    #$01      TO SET FUNCTION FLAG
0764A F4C7 CE 716D   A         LDX     #$716D    CODE FOR 'FS'
0765A F4CA B7 E43E   A FOUT    STAA    FNCFL
0766A F4CD FF E421   A         STX     DISBUF+4 .
0767A F4D0 39                  RTS               RETURN TO PUT
0768                    ************************************************************
0769                    *
0770                    * FCLR - CLEAR FUNCTION FLAG & LAST 2 DIGITS
0771                    *
0772                    ************************************************************
0773A F4D1 4F          FCLR    CLRA              TO CLEAR FUNCTION FLAG
0774A F4D2 CE 0000   A         LDX     #$0000    TO CLEARLAST 2 DIGITS
0775A F4D5 20 F3 F4CA          BRA     FOUT
```

```
0778                    ********************************************************
0779                    *
0780                    * TAPES - SOFTWARE CASSETTE TAPE INTERFACE
0781                    *
0782                    ********************************************************
0783A F4D7 7D E43E  A TAPBEG TST     FNCFL     SEE IF PUNCH OR LOAD
0784A F4DA 27 06 F4E2        BEQ     PCH
0785A F4DC BD F69C  A LDTAP  JSR     LOAD      DO LOAD (OR VERF)
0786A F4DF 7E F024  A        JMP     PROMPT    WHEN DONE
0787                    *
0788A F4E2 CE F4EC  A PCH    LDX     #BEGEND   POINT AT BEGEND ROUTINE
0789A F4E5 FF E419  A        STX     MNPTR     ACTIVATE
0790A F4E8 86 BB    A        LDAA    #$BB
0791A F4EA 20 1D F509        BRA     CONOUT    DISPLAY BB IN LAST DISPLAYS
0792                    *
0793A F4EC 7D E41C  A BEGEND TST     KYFLG     SEE IF KEY PENDING
0794A F4EF 26 01 F4F2        BNE     ADNOW
0795A F4F1 39                RTS               ** RETURN ** NO KEY
0796                    *
0797A F4F2 BD F1EF  A ADNOW  JSR     RDKEY     READ & ACKNOWLEDGE KEY
0798A F4F5 2B 05 F4FC        BMI     FUNK      FUNCTION KEY
0799A F4F7 BD F1CC  A        JSR     ROLL4     ENTER NEW NUMBER
0800A F4FA 20 16 F512        BRA     DYSOUT    CONVERT TO 7-SEG & LEAVE
0801                    *
0802A F4FC 86 EE    A FUNK   LDAA    #$EE
0803A F4FE B1 E42E  A        CMPA    HEXBUF+2  END ADDR DONE ?
0804A F501 27 12 F515        BEQ     DOPCH     GO DO PUNCH
0805A F503 FE E42C  A        LDX     HEXBUF    SAVE ENTERED ADDR
0806A F506 FF E460  A        STX     BEGAD
0807A F509 B7 E42E  A CONOUT STAA    HEXBUF+2  'EE' OR 'BB' TO LAST DISPLAYS
0808A F50C 7F E42C  A        CLR     HEXBUF    CLEAR FIRST FOUR NIBBLES
0809A F50F 7F E42D  A        CLR     HEXBUF+1
0810A F512 7E F120  A DYSOUT JMP     DYSCOD    CONV & RETURN
0811                    *
0812A F515 FE E42C  A DOPCH  LDX     HEXBUF    SAVE ENTERED ADDR
0813A F518 FF E462  A        STX     ENDAD
0814A F51B BD F630  A        JSR     PUNCH     PUNCH TAPE
0815A F51E 7E F024  A        JMP     PROMPT    WHEN DONE
0816                    *
```

```
0818                    ************************************************************
0819                    * FEDGE - ROUTINE TO LOCATE AN EDGE (POS OR NEG)
0820                    *         AND DETRMINE DISTANCE TO IT (TIME)
0821                    *              EXECUTION TIME TUNED
0822                    ************************************************************
0823                    *                                  8 FOR BSR
0824A F521 86 05    A FEDGE  LDAA   #5          2 START COUNT=FIXED (-1)
0825A F523 F6 E484  A        LDAB   PIADP       4 CLEAR INTERRUPT
0826A F526 01                NOP                2 DELAY
0827A F527 4C          LOOPF INCA               2 DURATION COUNT IN A-REG
0828A F528 F6 E485  A        LDAB   PIACR       4 CHECK FOR EDGE FOUND
0829A F52B 2A FA F527        BPL    LOOPF       4 IF NOT;KEEP LOOKING
0830A F52D C8 02    A        EORB   #$02        2 INVERT EDGE SENSE CONTROL
0831A F52F F7 E485  A        STAB   PIACR       5 PIA LOOKS FOR OTHER EDGE
0832A F532 39                RTS                5 **RETURN**


0834                    ************************************************************
0835                    * TIN - READ 1 BYTE FROM TAPE
0836                    *       TIME TUNED
0837                    *
0838                    ************************************************************
0839                    *                                  9 FOR JSR
0840A F533 86 FF    A TIN    LDAA   #$FF        2
0841A F535 B7 E459  A        STAA   BYTE        5 INITIALIZE BYTE
0842A F538 7F E45A  A        CLR    CYCNT       6
0843A F53B 7F E45B  A        CLR    CYCNT+1     6 INIT BIT-TIME COUNT
0844A F53E 7F E45C  A        CLR    GOODlS      6 INIT LOGIC SENSE
0845A F541 8D DE F521        BSR    FEDGE       [22/21+-5] SYNC TO AN EDGE
0846A F543 7D F543  A        TST    *           6 DELAY
0847A F546 7D F546  A NOTSH  TST    *           6 DELAY
0848A F549 B7 E45D  A        STAA   OLD         5 "
0849A F54C 8D D3 F521        BSR    FEDGE       [22/21+-5] MEASURE TO NEXT EDGE
0850A F54E 81 1B    A        CMPA   #27         2 <1.5 SHORT HALF ?
0851A F550 2C F4 F546        BGE    NOTSH       4 MUST FIND SHORT FIRST
0852A F552 B7 E45D  A LOOPS  STAA   OLD         5 SAVE LAST COUNT
0853A F555 8D CA F521        BSR    FEDGE       [22/21+-5] MEASURE TO NEXT
0854A F557 16                TAB                2 MAKE EXTRA COPY
0855A F558 FB E45D  A        ADDB   OLD         4 SUM OF LAST 2
0856A F55B C1 2B    A        CMPB   #43         2 >2.33 NOM. SHORTS?
0857A F55D 2F F3 F552        BLE    LOOPS       4 KEEP LOOKING FOR LONG
0858                    *
0859                    * EDGE SENSE SET-UP TO SENSE TRAILING EDGE OF CYCLES
0860                    * & YOU ARE IN THE MIDDLE OF THE FIRST LONG CYCLE
0861                    *
0862A F55F 7E F562  A        JMP    *+3         3 DELAY
0863A F562 F6 E484  A        LDAB   PIADP       4 CLEAR INTERRUPT FLAG
0864A F565 8B 05    A        ADDA   #5          2 COMPENSATE FOR PROCESSING
0865A F567 20 10 F579        BRA    SYNCIN      4 BRANCH INTO COUNT LOOP
0866A F569 86 00    A LPOUT  LDAA   #0          2 INIT BIT-TIME COUNT
0867A F56B 20 00 F56D        BRA    LPMID       4 DELAY
0868A F56D 7F E45A  A LPMID  CLR    CYCNT       6
0869A F570 B7 E45B  A        STAA   CYCNT+1     5 ESTABLISH BIT-TIME COUNT
0870A F573 7F E45C  A        CLR    GOODlS      6 INIT LOGIC SENSE
0871A F576 86 0A    A LPIN   LDAA   #10         2 FIXED TIME (-1)= INIT COUNT
0872A F578 4C          LOOPl INCA               2 A-REG HOLDS DURATION COUNT
0873A F579 F6 E485  A SYNCIN LDAB   PIACR       4 EDGE YET?
```

```
0874A F57C 2A FA F578          BPL    LOOP1     4 IF NOT;KEEP LOOKING
0875A F57E F6 E484  A          LDAB   PIADP     4 CLEAR INTERRUPT FLAG
0876A F581 7D F581  A          TST    *         6 DELAY TO MAKE PASS TIME...
0877A F584 01                  NOP              2 EVEN MULTIPLE OF LOOP TIME
0878A F585 81 34    A          CMPA   #52       2 <1.4 SHORT ?
0879A F587 2D 05 F58E          BLT    SHRT      4
0880A F589 7C E45C  A          INC    GOOD1S    6 GOOD1S POS MEANS 0
0881A F58C 20 05 F593          BRA    WITHIN    4
0882A F58E 7A E45C  A SHRT     DEC    GOOD1S    6 GOOD1S NEG MEANS 1
0883A F591 20 00 F593          BRA    WITHIN    4 DELAY
0884A F593 F6 E45A  A WITHIN   LDAB   CYCNT     4 HIGH BYTE
0885A F596 BB E45B  A          ADDA   CYCNT+1   4 ADD CURRENT TO BIT-TIME COUNT
0886A F599 B7 E45B  A          STAA   CYCNT+1   5 UPDATE
0887A F59C C9 00    A          ADCB   #0        2 ADD IN CARRY
0888A F59E F7 E45A  A          STAB   CYCNT     5 UPDATE HIGH BYTE
0889A F5A1 26 03 F5A6          BNE    CHKOVR    4 IF CARRY; BIT MAY BE OVER
0890A F5A3 01                  NOP              2 DELAY
0891A F5A4 20 04 F5AA          BRA    NOTOVR    4 BIT NOT OVER
0892A F5A6 81 17    A CHKOVR   CMPA   #23       2 (279-256)
0893A F5A8 2C 0A F5B4          BGE    BITOVR    4 BIT-TIME EXPIRED
0894A F5AA C6 05    A NOTOVR   LDAB   #5        [38] 2
0895A F5AC 5A                  DECB             "    2
0896A F5AD 2A FD F5AC          BPL    *-1       "    4
0897A F5AF 7E F5B2  A          JMP    *+3       3
0898A F5B2 20 C2 F576          BRA    LPIN      4
0899                    *
0900                    * END OF A BIT-TIME
0901                    *
0902A F5B4 78 E45C  A BITOVR   ASL    GOOD1S    6 LOGIC SENSE TO CARRY
0903A F5B7 76 E459  A          ROR    BYTE      6 SHIFT NEW BIT INTO BYTE
0904A F5BA 24 08 F5C4          BCC    TINDUN    4 DONE WHEN START FALLS OUT
0905A F5BC 81 5D    A          CMPA   #93       2 >2.5 NOM. SHORTS ?
0906A F5BE 2D A9 F569          BLT    LPOUT     4 NO; BIT-TIME STARTS AT 0
0907A F5C0 86 24    A          LDAA   #36       2 YES; TRY MAINTAIN FRAMING
0908A F5C2 20 A9 F56D          BRA    LPMID     4 NEXT BIT-TIME
0909                    *
0910                    * DATA BYTE READ; CLEAN-UP AND LEAVE
0911                    *
0912A F5C4 B6 E459  A TINDUN   LDAA   BYTE      4 GET CURRENT BYTE
0913A F5C7 BB E45E  A          ADDA   CHKSM     4 ADD TO CHECKSUM
0914A F5CA B7 E45E  A          STAA   CHKSM     5 UPDATE
0915A F5CD B6 E459  A          LDAA   BYTE      4 GET RECEIVED DATA IN A-REG
0916A F5D0 39                  RTS              5 ** RETURN **
```

```
0918                    ************************************************************
0919                    * BIT1 - SEND A LOGIC 1 BIT-TIME
0920                    *        LESS 177 CLOCK CYCLES
0921                    *           TIME TUNED
0922                    ************************************************************
0923                    *                           8 FOR BSR
0924A F5D1 C6 0F    A BIT1   LDAB   #15         2 # SHORT H-CYCS (-1)
0925A F5D3 BD F5FF  A LOOPB1 JSR    INVRT       [20/5] TRANSMIT EDGE
0926A F5D6 86 18    A        LDAA   #24         [152] 2 DELAY
0927A F5D8 4A                DECA               " 2
0928A F5D9 2A FD F5D8        BPL    *-1         " 4
0929A F5DB 20 00 F5DD        BRA    *+2         4 DELAY
0930A F5DD 5A                DECB               2 1 LESS HALF CYCLE
0931A F5DE 26 F3 F5D3        BNE    LOOPB1      4 TILL 2ND LAST EDGE
0932A F5E0 BD F5FF  A        JSR    INVRT       [20/5] 15TH EDGE IN BIT-TIME
0933A F5E3 39                RTS                5 **RETURN** 177 CYC TO NXT


0935                    ************************************************************
0936                    * BIT0 - SEND A LOGIC 0 BIT-TIME
0937                    *        LESS 177 CLOCK CYCLES
0938                    *           TIME TUNED
0939                    ************************************************************
0940                    *                           8 FOR BSR
0941A F5E4 C6 07    A BIT0   LDAB   #7          2 # LONG H-CYCS (-1)
0942A F5E6 BD F5FF  A LOOPB0 JSR    INVRT       [20/5] TRANSMIT EDGE
0943A F5E9 86 38    A        LDAA   #56         [344] 2 DELAY
0944A F5EB 4A                DECA               " 2
0945A F5EC 2A FD F5EB        BPL    *-1         " 4
0946A F5EE 01                NOP                2 DELAY
0947A F5EF 5A                DECB               2 1 LESS TO GO
0948A F5F0 26 F4 F5E6        BNE    LOOPB0      4 TILL 2ND LAST EDGE
0949A F5F2 BD F5FF  A        JSR    INVRT       [20/5] 7TH EDGE IN BIT-TIME
0950A F5F5 86 1D    A        LDAA   #29         [182] 2 DELAY
0951A F5F7 4A                DECA               " 2
0952A F5F8 2A FD F5F7        BPL    *-1         " 4
0953A F5FA 7E F5FD  A        JMP    *+3         3 DELAY
0954A F5FD 01                NOP                2 "
0955A F5FE 39                RTS                5 **RETURN** 177 CYC TO NXT


0957                    ************************************************************
0958                    * INVRT - ROUTINE TO TRANSMIT A RISING
0959                    *         OR FALLING EDGE TO THE CASSETTE
0960                    *            TIME TUNED
0961                    ************************************************************
0962                    *                           9 FOR JSR
0963A F5FF 86 80    A INVRT  LDAA   #$80        2
0964A F601 B8 E486  A        EORA   PIADPB      4
0965A F604 B7 E486  A        STAA   PIADPB      5 INVERT OUTPUT
0966A F607 39                RTS                5 ** RETURN **
```

```
0968                    ************************************************************
0969                    * PNCHB - PUNCH 1 BYTE TO TAPE. INCLUDES
0970                    *         START BIT,DATA,AND ALL BUT LAST HALF-CYCLE
0971                    *         OF STOP BITS
0972                    *                    TIME TUNED
0973                    ************************************************************
0974                    *                              9 FOR JSR
0975A F608 B7 E459  A PNCHB  STAA   BYTE      5 SAVE BYTE TO PUNCH
0976A F60B 8D D7 F5E4        BSR    BIT0      [30/<177>] SEND START BIT
0977A F60D 86 09    A        LDAA   #9        2 # BITS IN BYTE (+2 STOP) (-1)
0978A F60F B7 E45F  A        STAA   NBITS     5 ESTABLISH BIT COUNT
0979A F612 7D F612  A        TST    *         6 DELAY
0980A F615 86 13    A LPPOUT LDAA   #19       [122] 2 DELAY
0981A F617 4A                DECA             " 2
0982A F618 2A FD F617        BPL    *-1       " 4
0983A F61A 0D                SEC              2 SO LAST 2 BIT TIMES = 1'S
0984A F61B 76 E459  A        ROR    BYTE      6 LOGIC SENSE TO CARRY
0985A F61E 25 05 F625        BCS    D01       4 IF LOGIC 1
0986A F620 8D C2 F5E4        BSR    BIT0      [30/<177>] XMIT A 0 BIT-TIME
0987A F622 7E F62A  A        JMP    ENDBIT    3
0988A F625 8D AA F5D1 D01    BSR    BIT1      [30/<177>] XMIT A 1 BIT-TIME
0989A F627 7E F62A  A        JMP    ENDBIT    3 MATCHING DELAY
0990A F62A 7A E45F  A ENDBIT DEC    NBITS     6 1 LESS BIT-TIME TO GO
0991A F62D 2A E6 F615        BPL    LPPOUT    4 CONTINUE FOR BYTE+STOP BITS
0992A F62F 39                RTS              5 ** RETURN ** 159 CYC TO NXT
```

```
0994                    ************************************************************
0995                    * PUNCH - FORMAT AND PUNCH A CASSETTE DATA FILE
0996                    *         INCLUDING LEADER AND CHECKSUM
0997                    *             EXECUTION TIME TUNED
0998                    ************************************************************
0999                    *                              9 FOR JSR
1000A F630 CE 0348  A PUNCH  LDX    #840      3 COUNT FOR 30-SEC LEADER
1001A F633 86 FF    A LLOOP  LDAA   #$FF      2 LEADER CHARACTER
1002A F635 C6 10    A        LDAB   #16       [104] 2 DELAY
1003A F637 5A                DECB             " 2
1004A F638 2A FD F637        BPL    *-1       " 4
1005A F63A BD F608  A        JSR    PNCHB     [44/<159>] PUNCH A LEADER CHAR
1006A F63D 09                DEX              4
1007A F63E 26 F3 F633        BNE    LLOOP     4 CONTINUE FOR 30-SEC
1008                    *
1009                    * LEADER FINISHED
1010                    *
1011A F640 86 53    A        LDAA   #'S       2 BLOCH START CHAR
1012A F642 C6 10    A        LDAB   #16       [104] 2 DELAY
1013A F644 5A                DECB             " 2
1014A F645 2A FD F644        BPL    *-1       " 4
1015A F647 BD F608  A        JSR    PNCHB     [44/<159>] PUNCH START CHAR
1016A F64A 01                NOP              2 DELAY
1017A F64B 7F E45E  A        CLR    CHKSM     6 INITIALIZE CHECKSUM
1018A F64E CE E460  A        LDX    #BEGAD    3 POINT AT FIRST ADDR BYTE0LT
1019A F651 A6 00    A ADLOOP LDAA   0,X
1020A F653 16                TAB              2 EXTRA COPY
1021A F654 FB E45E  A        ADDB   CHKSM     4 ADDR IS PART OF CHECKSUM
1022A F657 F7 E45E  A        STAB   CHKSM     5 UPDATE
1023A F65A 01                NOP              2 DELAY
1024A F65B C6 0D    A        LDAB   #13       [86] 2
1025A F65D 5A                DECB             " 2
1026A F65E 2A FD F65D        BPL    *-1       " 4
1027A F660 BD F608  A        JSR    PNCHB     [44/<159>] PUNCH ADDR BYTE
1028A F663 08                INX              4 ADV TO NXT ADDR BYTE
1029A F664 8C E464  A        CPX    #BEGAD+4  3 DONE YET ?
1030A F667 26 E8 F651        BNE    ADLOOP    4 CONTINUE FOR 4 ADDR CHARS
1031                    *
1032                    * READY TO PUNCH DATA
1033                    *
1034A F669 01                NOP              2 DELAY
1035A F66A 01                NOP              2 DELAY
1036A F66B FE E460  A        LDX    BEGAD     5 GET BEG ADDR OF DATA
1037A F66E A6 00    A DLOOP  LDAA   0,X        5 GET A DATA BYTE
1038A F670 16                TAB              2 EXTRA COPY
1039A F671 FB E45E  A        ADDB   CHKSM     4 ADD TO CHECKSUM
1040A F674 F7 E45E  A        STAB   CHKSM     5 UPDATE
1041A F677 F7 E45E  A        STAB   CHKSM     5 DELAY
1042A F67A C6 0B    A        LDAB   #11       [74] 2
1043A F67C 5A                DECB             " 2
1044A F67D 2A FD F67C        BPL    *-1       " 4
1045A F67F BD F608  A        JSR    PNCHB     [44/<159>] PUNCH DATA BYTE
1046A F682 7E F685  A        JMP    *+3       3 DELAY
1047A F685 BC E462  A        CPX    ENDAD     5 SEE IF DONE
1048A F688 27 03 F68D        BEQ    DUNDAT    4 IF FINISHED
1049A F68A 08                INX              4 ELSE ADV TO NXT
1050A F68B 20 E1 F66E        BRA    DLOOP     4 AND CONTINUE LOOP
```

```
1052                    *
1053                    * READY TO PUNCH CHECKSUM
1054                    *
1055A F68D 70 E45E  A DUNDAT NEG     CHKSM     6 SUM INCL, CHECK WILL BE 0
1056A F690 B6 E45E  A        LDAA    CHKSM     4 PREPARE TO SEND
1057A F693 C6 14    A        LDAB    #20       [128] 2
1058A F695 5A                DECB              " 2
1059A F696 2A FD F695        BPL     *-1       " 4
1060A F698 BD F608  A        JSR     PNCHB     [44/<159>] PUNCH CHECKSUM
1061A F69B 39                RTS               5 ** RETURN **
```

```
1063                    ************************************************************
1064                    *
1065                    * LOAD - LOAD OR VERIFY A DATA FILE FROM
1066                    *        CASSETTE TAPE
1067                    *
1068                    ************************************************************
1069                    *                              9 FOR A JSR
1070A F69C BD F533  A LOAD   JSR    TIN       [56/101+-5] READ A BYTE FROM TAPE
1071A F69F 81 53    A       CMPA   #'S       2 BLOCK START ?
1072A F6A1 26 F9 F69C       BNE    LOAD      4 NO; TRY AGAIN
1073                    *
1074                    * BLOCK START FOUND; NOW READ BEG & END ADDR'S
1075                    *
1076A F6A3 CE E460  A       LDX    #BEGAD    3 POINT AT ADDR AREA
1077A F6A6 7F E45E  A       CLR    CHKSM     6 INITIALIZE CHECKSUM
1078A F6A9 BD F533  A LOPAD  JSR    TIN       [56/101+-5] GET ADDR CHAR
1079A F6AC A7 00    A       STAA   0,X       6 STORE RECIEVED ADDR CHAR
1080A F6AE 08               INX              4 POINT AT NEXT ADDR LOC
1081A F6AF 8C E464  A       CPX    #BEGAD+4  3 DONE GETTING ADDR'S ?
1082A F6B2 26 F5 F6A9       BNE    LOPAD     4 NO; CONTINUE
1083                    *
1084                    * READY TO READ DATA
1085                    *
1086A F6B4 FE E460  A       LDX    BEGAD     5 POINT TO WHERE DATA GOES
1087A F6B7 BD F533  A LOPDAT JSR    TIN       [56/101+-5] GET DATA FROM TAPE
1088A F6BA 7D E43E  A       TST    FNCFL     6 SEE IF LOAD OR VERF ?
1089A F6BD 27 04 F6C3       BEQ    VERF      4 IF NOT SET; IT'S VERF
1090A F6BF A7 00    A       STAA   0,X       6 IT'S LOAD SO STORE DATA
1091A F6C1 20 04 F6C7       BRA    LOPBOT    4 GO TO BOTTOM OF LOOP
1092A F6C3 A1 00    A VERF   CMPA   0,X       5 JUST COMPARE TO MEM
1093A F6C5 26 11 F6D8       BNE    BAD       4 IF NON-COMPARE; SIGNAL ERROR
1094A F6C7 BC E462  A LOPBOT CPX    ENDAD     5 DONE ?
1095A F6CA 27 03 F6CF       BEQ    CHKCHK    4 IF SO; CHECK CHECKSUM
1096A F6CC 08               INX              4 POINT AT NEXT DATA LOC
1097A F6CD 20 E8 F6B7       BRA    LOPDAT    4 AND CONTINUE LOAD/VERF
1098                    *
1099                    * DATA FINISHED... NOW CHECK CHECKSUM
1100                    *
1101A F6CF BD F533  A CHKCHK JSR    TIN       [56/105+-5] GET CHECKSUM
1102A F6D2 7D E45E  A       TST    CHKSM
1103A F6D5 26 01 F6D8       BNE    BAD       4 IF NOT ZERO; BAD CHECKSUM
1104A F6D7 39               RTS              5 ** RETURN **
1105                    *
1106A F6D8 FF E434  A BAD    STX    UX        6 SO USER CAN SEE END ADDR
1107A F6DB B7 E433  A       STAA   UA        5 SO USER CAN CHECK IT
1108A F6DE 7D E43E  A       TST    FNCFL     CHECK FOR ERROR OVERRIDE
1109A F6E1 2A 01 F6E4       BPL    STOP
1110A F6E3 39               RTS              ** RETURN ** NO MESSAGE
1111                    *
1112A F6E4 CE 7177  A STOP   LDX    #$7177    "FA"
1113A F6E7 FF E41D  A       STX    DISBUF
1114A F6EA CE 0638  A       LDX    #$0638    "IL"
1115A F6ED FF E41F  A       STX    DISBUF+2
1116A F6F0 7E F735  A       JMP    ALTBAD    PRINT "FAIL ??"
```

```
1119                    ************************************************
1120                    *
1121                    * GO - GO TO USER PROGRAM
1122                    *
1123                    ************************************************
1124A F6F3 7D E423  A GO      TST     ROLPAS    HEX DATA PRIOR TO 'GO' ?
1125A F6F6 26 06 F6FE         BNE     CONTIN    IF NOT; ASSUME UPC
1126A F6F8 FE E42C  A         LDX     HEXBUF    GET ENTERED VALUE
1127A F6FB FF E436  A         STX     UPC       STORE AS GO ADDR
1128A F6FE CE F70B  A CONTIN  LDX     #GO1      RETURN ADDR AFTER ROI
1129A F701 FF E439  A ROI     STX     ROIBAK    SAVE IN RAM
1130A F704 86 01    A         LDAA    #1
1131A F706 B7 E438  A         STAA    ROIFLG    SIGNAL SINGLE TRACE
1132A F709 20 03 F70E         BRA     GOTO      EXIT (NO BREAKS)
1133                    * COME HERE AFTER RUNNING ONE INSTRUCTION
1134A F70B BD F45F  A GO1     JSR     INBKS     INSTALL BREAKPOINTS
1135A F70E BE E42F  A GOTO    LDS     USP       GET USER'S STACK POINTER
1136A F711 86 55    A         LDAA    #$55      START TEST FOR EXISTANCE OF STK
1137A F713 36                 PSHA
1138A F714 32                 PULA
1139A F715 81 55    A         CMPA    #$55      DID IT GO ?
1140A F717 26 10 F729         BNE     BADSTK    NO; STACK IS BAD
1141A F719 B6 E437  A         LDAA    UPC+1     LOW BYTE
1142A F71C 36                 PSHA              STACK FOR RTS
1143A F71D B6 E436  A         LDAA    UPC       HIGH BYTE
1144A F720 36                 PSHA
1145A F721 86 AA    A         LDAA    #$AA      SEE IF STACK STILL OK
1146A F723 36                 PSHA
1147A F724 32                 PULA
1148A F725 81 AA    A         CMPA    #$AA
1149A F727 27 1E F747         BEQ     GOEXIT    OK; FINAL EXIT SEQ
1150A F729 CE 406D  A BADSTK  LDX     #$406D    MESSAGE "-SP- ??" TO 7-SEGS
1151A F72C FF E41D  A         STX     DISBUF
1152A F72F CE 7340  A         LDX     #$7340
1153A F732 FF E41F  A         STX     DISBUF+2
1154A F735 CE 5353  A ALTBAD  LDX     #$5353
1155A F738 FF E421  A         STX     DISBUF+4
1156A F73B 8E E47E  A         LDS     #STKTOP   INIT TO GOOD AREA
1157A F73E CE F0A2  A         LDX     #DIDDLE   DO-NNOTHING SUB
1158A F741 FF E419  A         STX     MNPTR     STORE AS MAIN PROG
1159A F744 7E F0BB  A         JMP     PUT       ONLY ESCAPE IS RESET OR 'EX'
1160                    *
1161A F747 FE E434  A GOEXIT  LDX     UX        RECOVER USER STATUS
1162A F74A F6 E432  A         LDAB    UB
1163A F74D B6 E433  A         LDAA    UA
1164A F750 36                 PSHA              TEMP SAVE ON USER'S STACK
1165A F751 86 01    A         LDAA    #1
1166A F753 B7 E43B  A         STAA    UPROG     FLAG SIGNALS IN USER PROG
1167A F756 7D E438  A         TST     ROIFLG    TRACE EXIT ?
1168A F759 27 12 F76D         BEQ     ABSOUT    IF NOT;; JUST GET GOING
1169A F75B 86 3C    A         LDAA    #$3C
1170A F75D B7 E485  A         STAA    PIACRA    HOLDS TRACE COUNTER RESET
1171A F760 B6 E486  A         LDAA    PIAPB     READ TO CLEAR ANY INT FLAG
1172A F763 86 0E    A         LDAA    #$0E
1173A F765 B7 E487  A         STAA    PIACRB    ENABLE TRACE NMI
1174A F768 86 34    A         LDAA    #$34
1175A F76A B7 E485  A         STAA    PIACRA    RELEASE TIMER
1176A F76D B6 E431  A ABSOUT  LDAA    UCC       TIMED EXIT TO USER PROG
```

```
                        GO

1177A F770 06                   TAP             SET USER COND CODES
1178A F771 32                   PULA            GET USER A-REG; DON'T MESS 'CC'
1179A F772 39                   RTS             *** EXIT TO USER PROG ***
1180                *
```

```
1184                    ******************************************************
1185                    *
1186                    * INTERRUPTS - INTERRUPT HANDLING ROUTINES
1187                    *
1188                    ******************************************************
1189A F773 01           NMINT   NOP             SET IRQ FLAG
1190A F774 0F                   SEI             .
1191A F775 86 04    A           LDAA    #$04    PIA DISABLE CODE
1192A F777 B7 E487  A           STAA    PIACRB  DISABLE NMI'S DURING SERVICE
1193A F77A B6 E487  A           LDAA    PIACRB  READ INT STATUS
1194A F77D 2A 12 F791           BPL     SAVE    IF RETURN FROM TRACE
1195                    * KEY CLOSURE CAUSED NMI
1196A F77F BD F04E  A           JSR     GET     FIND AND DEBOUNCE KEY
1197A F782 81 81    A           CMPA    #$81    'EX' ?
1198A F784 27 03 F789           BEQ     ABORT
1199A F786 8D 26 F7AE           BSR     ENNMI   RE-ENABLE INTERRUPT
1200A F788 3B                   RTI             *** DONE: RETURN ***
1201                    * 'EX' KEY; PROMPT OR ABORT
1202A F789 7D E43B  A ABORT     TST     UPROG   ESCAPE FROM USER PROG ?
1203A F78C 26 03 F791           BNE     SAVE    IF ESCAPE FROM USER PROG
1204A F78E 7E F024  A           JMP     PROMPT  *** ALREADY IN OP-SYST ***
1205A F791 BF E42F  A SAVE      STS     USP     SAVE POINTER TO USER REGS
1206A F794 8E E47E  A           LDS     #STKTOP INIT TO SYST AREA
1207A F797 8D 23 F7BC           BSR     SVSTAT  RECOVER STATUS AT 'EX' TIME
1208A F799 8D 13 F7AE           BSR     ENNMI   RE-ENABLE KEY NMI
1209A F79B 7F E43B  A           CLR     UPROG   SIGNAL NOT IN USER PROGRAM
1210A F79E 7D E438  A           TST     ROIFLG  IS THIS RETURN FROM TRACE ?
1211A F7A1 27 08 F7AB           BEQ     NOTROI  IF NOT
1212A F7A3 7F E438  A           CLR     ROIFLG  SIGNAL NOT ROI NOW
1213A F7A6 FE E439  A           LDX     ROIBAK  GET RETURN ADDR
1214A F7A9 6E 00    A           JMP     0,X     AND RETURN FROM ROI
1215A F7AB 7E F2CA  A NOTROI    JMP     REGBEG  *** TO REG DISPLAY ***
1216                    *
1217                    *                       DUMMY READ DROPS THE FLAG
1218A F7AE B6 E486  A ENNMI     LDAA    PIAPB   TO CLEAR FLAGS
1219A F7B1 86 07    A           LDAA    #$07    ENABLE KEY INTERRUPT CODE
1220A F7B3 B7 E487  A           STAA    PIACRB  TO PIA CONTROL REG
1221A F7B6 86 FF    A           LDAA    #$FF
1222A F7B8 B7 E486  A           STAA    PIAPB   ENABLE ALL KEY ROWS
1223A F7BB 39                   RTS             ** RETURN **
1224                    *
1225                    *
1226A F7BC BE E42F  A SVSTAT    LDS     USP     POINT AT STACKED STATUS
1227A F7BF CE E431  A           LDX     #UCC    POINT AT PSEUDO REG AREA
1228A F7C2 32           SVLOOP  PULA            GET STACKED BYTE
1229A F7C3 A7 00    A           STAA    ,X      STORE AT PSEUDO REG RAM LOC
1230A F7C5 08                   INX             POINT AT NEXT REG LOC
1231A F7C6 8C E438  A           CPX     #UPC+2  PAST END ?
1232A F7C9 26 F7 F7C2           BNE     SVLOOP  IF NOT CONTINUE LOOP
1233A F7CB BF E42F  A           STS     USP     SAVE USER SP AT INTERRUPT TIME
1234A F7CE 8E E47C  A           LDS     #STKTOP-2 SET FOR RETURN
1235A F7D1 39                   RTS             ** RETURN **
1236                    *
1237                    *
1238A F7D2 01           SWINT   NOP             SET IRQ FLAG
1239A F7D3 0F                   SEI             .
1240A F7D4 BF E42F  A           STS     USP     POINTER TO USER'S REGS
1241A F7D7 8E E47E  A           LDS     #STKTOP INIT TO SYST AREA
```

```
1242A F7DA 8D E0 F7BC         BSR     SVSTAT   RECOVER BREAK STATUS
1243A F7DC FE E436  A         LDX     UPC      BACK UP PROG CNTR
1244A F7DF 09                 DEX              .
1245A F7E0 FF E436  A         STX     UPC      .
1246A F7E3 BD F485  A         JSR     OUTBKS   TAKE OUT BREAKPOINTS
1247A F7E6 7F E43B  A         CLR     UPROG    SIGNAL NOT IN USER PROG
1248A F7E9 7E F2CA  A         JMP     REGBEG   *** TO REG DISPLAY ***
1249                  *
1250                  *
1251A F7EC FE E43C  A UIRQ    LDX     UIRQV    GET USER IRQ VECTOR
1252A F7EF 6E 00    A         JMP     0,X      *** GO TO USER SERVICE ROUTINE ***
1253                  *
```

```
1256                    ************************************************************
1257                    *
1258A E419              *       ORG     $E419
1259                    *
1260                    * DEFS - DEFINITIONS & SCRATCH LOCATIONS
1261                    *
1262                    ************************************************************
1263A E419     0002   A MNPTR   RMB     2       POINTER TO ACTIVE SUBROUTINE
1264A E41B     0001   A KEY     RMB     1       KEY DATA
1265A E41C     0001   A KYFLG   RMB     1       KEY PENDING FLAG
1266A E41D     0006   A DISBUF  RMB     6       DISPLAY BUFFER
1267A E423     0001   A ROLPAS  RMB     1       FIRST PASS OF DATA ROL-ENT
1268A E424     0002   A XSAVD   RMB     2       X SCRATCH
1269A E426     0002   A XSAV1   RMB     2       .
1270A E428     0002   A XTMP1   RMB     2       .
1271A E42A     0002   A MEMSAV  RMB     2       SAVE MEM POINTER DURING OFFSET CAL
1272A E42C     0003   A HEXBUF  RMB     3       HEX INPUT BUFFER
1273A E42F     0002   A USP     RMB     2       USER STACK POINTER
1274A E431     0001   A UCC     RMB     1       USER CONDITION CODE
1275A E432     0001   A UB      RMB     1       USER B REGISTER
1276A E433     0001   A UA      RMB     1       USER A REGISTER
1277A E434     0002   A UX      RMB     2       USER X REGISTER
1278A E436     0002   A UPC     RMB     2       USER PROGRAM COUNTER
1279A E438     0001   A ROIFLG  RMB     1       RUN-ONE-INSTRUCTION FLAG
1280A E439     0002   A ROIBAK  RMB     2       ADDR TO RETURN TO AFTER ROI
1281A E43B     0001   A UPROG   RMB     1       FLAG INDICATES IN-USER-PROG
1282A E43C     0002   A UIRQV   RMB     2       ADDR OF USER'S IRQ SERVICE ROUTINE
1283A E43E     0001   A FNCFL   RMB     1       SPECIAL FUNCTION FLAG
1284A E43F     0002   A FNCPNT  RMB     2       POINT TO USER'S SPECIAL FUNCTION
1285A E441     0001   A REGNO   RMB     1       REGISTER NUMBER (USED IN REGDIS)
1286A E442     0002   A BKPNTR  RMB     2       POINTS INTO BREAKPOINT TABLE
1287A E444     0001   A BRKNO   RMB     1       # OF BREAKPOINTS IN TABLE
1288A E445     0014   A BRKTAB  RMB     20      BREAKPOINT TABLE
1289           E459   A BRKEND  EQU     *       END OF TABLE
1290                    *
1291                    * CASSETTE INTERFACE SCRATCH LOCATIONS
1292                    *
1293A E459     0001   A BYTE    RMB     1       DATA BUFFER
1294A E45A     0002   A CYCNT   RMB     2       CYCLE COUNT REG
1295A E45C     0001   A GOOD1S  RMB     1       # OF GOOD 1'S
1296A E45D     0001   A OLD     RMB     1
1297A E45E     0001   A CHKSM   RMB     1       CHECKSUM REG
1298A E45F     0001   A NBITS   RMB     1
1299A E460     0002   A BEGAD   RMB     2       BEGINNING ADDRESS
1300A E462     0002   A ENDAD   RMB     2       END ADDRESS
1301                    *
1302           E484   A PIA     EQU     $E484   SYSTEM PIA BASE ADDR
1303           0000   A KPCOL   EQU     $0      KEYPAD COL PORT OFFSET
1304           0002   A KPROW   EQU     $2      KEYPAD ROW PORT OFFSET
1305           E484   A ANOD    EQU     $E484   DISPLAY SEG ANODES
1306           E486   A CATH    EQU     $E486   DISPAY CATHODES
1307           E486   A PIAROW  EQU     $E486   EXTENDED MODE ROW PORT ADDR
1308           E486   A PIADPB  EQU     $E486   PIA DATA PORT B
1309           E485   A PIACR   EQU     $E485   PIA CONTROL REG A
1310           E484   A PIADP   EQU     $E484   PIA DATA PORT A
1311           E485   A PIACRA  EQU     $E485   PIA CONTROL REG A
1312           E487   A PIACRB  EQU     $E487   PIA CONTROL REG B
1313           E486   A PIAPB   EQU     $E486   PIA DATA PORT B
```

```
1314          E47E  A STKTOP EQU    $E47E    TOP OF SYSTEM STACK
1315                *
1316                * SYSTEM VECTORS
1317                *
1318                * ON MEK6802D5 EITHER UPPER HALF
1319                * OF D5BUG ($F400-F7FF) MUST "MIRROR"
1320                * INTO ADDRESSES ($FC00-FFFF) OR
1321                * ELSE USER MUST SUPPLY PROM
1322                * MAPPED IN ($FC00-FFFF) AREA WHICH
1323                * CONTAINS ALTERNATE VECTORS.
1324                * IN THE CASE OF "MIRRORING" THE
1325                * FOLLOWING VECTORS WOULD ALSO
1326                * APPEAR AT THE NORMAL 6802
1327                * VECTOR LOCATIONS ($FFF8-FFFF)
1328A F7F8                     ORG    $F7F8
1329A F7F8    F7EC  A         FDB    UIRQ     USER IRQ VECTOR
1330A F7FA    F7D2  A         FDB    SWINT    SOFTWARE INTERRUPT VECTOR
1331A F7FC    F773  A         FDB    NMINT    NON MASKABLE INTERRUPT VECTOR
1332A F7FE    F000  A         FDB    RESET    RESTART VECTOR
1333                          END
     ERRORS 00000--00000
```

```
F789 ABORT    1198  1202*
F76D ABSOUT   1168  1176*
F183 ADDAX    0089  0187  0232  0279* 0514
F651 ADLOOP   1019* 1030
F4F2 ADNOW    0794  0797*
F735 ALTBAD   1116  1154*
E484 ANOD     0145  0149  1305*
F1BC ARNCL2   0303  0307*
F1DD ARNCL4   0321  0326*
F1A0 ARNCLR   0292  0294*
F191 ARND     0282  0284*
F243 ARNINC   0382  0384*
F30A ARNR1    0489  0491*
F31C ARNR2    0499  0501*
F365 ARNR3    0532  0536*
F3E3 BACK     0598  0612*
F6D8 BAD      1093  1103  1106*
F281 BADCAL   0404  0415*
F257 BADOFF   0385  0394*
F729 BADSTK   1140  1150*
E460 BEGAD    0806  1018  1029  1036  1076  1081  1086  1299*
F4EC BEGEND   0788  0793*
F5E4 BIT0     0941* 0976  0986
F5D1 BIT1     0924* 0988
F5B4 BITOVR   0893  0902*
F42E BKFMTB   0590  0656*
F3CB BKLOOP   0600* 0608
F444 BKLOP    0669* 0679
F43E BKNO     0625  0667*
E442 BKPNTR   0599  0609  0639  0658  0673  0694  0699  0718  0721  0754
              1286*
F3F1 BKTOTB   0587  0623*
F388 BRKBEG   0199  0569*
F38E BRKEDT   0570  0572*
```

```
E459 BRKEND  0604  0678  0706  0727  0745  1289*
F4A9 BRKLOP  0739* 0746
E444 BRKNO   0596  0613  0626  0645  0661  0667  0671  0688  1287*
F39C BRKPNT  0572  0577*
E445 BRKTAB  0606  0668  0690  0714  0738  1288*
F3A2 BRKTST  0578  0580*
E459 BYTE    0841  0903  0912  0915  0975  0984  1293*
F26B CALDUN  0363  0403*
E486 CATH    0146  0150  1306*
F6CF CHKCHK  1095  1101*
F5A6 CHKOVR  0889  0892*
E45E CHKSM   0913  0914  1017  1021  1022  1039  1040  1041  1055  1056
             1077  1102  1297*
F467 CKBKPT  0691* 0707
F3B5 CKFC    0586  0588*
F3BC CKGO    0589  0591*
F4B8 CKLSB   0740  0749*
F088 CLOP    0085* 0086
F195 CLRDS   0041  0180  0289* 0389  0398  0435  0616
F005 CLRLOP  0012* 0015
F19B CLRLP   0291* 0296
F063 COLFND  0060  0064*
F509 CONOUT  0791  0807*
F6FE CONTIN  1125  1128*
E45A CYCNT   0842  0843  0868  0869  0884  0885  0886  0888  1294*
F0A2 DIDDLE  0095* 1157
F3C0 DISBRK  0576  0596* 0657  0662
E41D DISBUF  0039  0138  0227  0290  0295  0369  0529  0531  0533  0534
             0766  1113  1115  1151  1153  1155  1266*
F3F0 DISDUN  0592  0614  0617*
F66E DLOOP   1037* 1050
F171 DLY1    0147  0268*
F169 DLY25   0087  0265*
F17C DLYLP   0267  0270  0272* 0273
F179 DLYX    0271*
F625 DO1     0985  0988*
F515 DOPCH   0804  0812*
F68D DUNDAT  1048  1055*
F080 DUNROW  0077  0080*
F120 DYSCOD  0178  0211* 0367  0387  0396  0451  0527  0584  0612  0648
             0810
F512 DYSOUT  0800  0810*
F159 DYSTBL  0230  0245*
E462 ENDAD   0813  1047  1094  1300*
F62A ENDBIT  0987  0989  0990*
F7AE ENNMI   0044  1199  1208  1218*
F4D1 FCLR    0197  0773*
F521 FEDGE   0824* 0845  0849  0853
E43E FNCFL   0037  0175  0351  0361  0391  0400  0418  0437  0462  0464
             0569  0765  0783  1088  1108  1283*
E43F FNCPNT  0182  1284*
F4A0 FNDBRK  0623  0656  0736*
F4CA FOUT    0765* 0775
F4C5 FSET    0196  0763*
F428 FULL    0624  0628  0635  0649*
F4FC FUNK    0798  0802*
F105 FUNKY   0174  0185*
F0E5 FUNSEL  0042  0169*
```

```
F04E GET      0054* 0063  0074  1196
F6F3 GO       0195  1124*
F70B GO1      1128  1134*
F747 GOEXIT   1149  1161*
E45C GOOD1S   0844  0870  0880  0882  0902  1295*
F70E GOTO     1132  1135*
F108 HASH     0183  0186*
E42C HEXBUF   0214  0225  0301  0323  0324  0332  0333  0352  0360  0374
              0376  0377  0378  0386  0395  0410  0449  0450  0523  0525
              0536  0538  0597  0611  0630  0641  0642  0647  0736  0737
              0803  0805  0807  0808  0809  0812  1126  1272*
F45F INBKS    0688* 1134
F5FF INVRT    0925  0932  0942  0949  0963*
F450 ISBKPT   0672  0674*
E41B KEY      0065  0068  0080  0082  0091  0340  1264*
F0EB KEYNOW   0170  0173*
0000 KPCOL    0056  0059  0070  0085  1303*
0002 KPROW    0058  0069  1304*
E41C KYFLG    0036  0093  0169  0339  0355  0476  0577  0793  1265*
F0A3 KYTBL    0088  0106*
F4DC LDTAP    0465  0785*
F633 LLOOP    1001* 1007
F69C LOAD     0785  1070* 1072
F578 LOOP1    0872* 0874
F5E6 LOOPB0   0942* 0948
F5D3 LOOPB1   0925* 0931
F527 LOOPF    0827* 0829
F552 LOOPS    0852* 0857
F6A9 LOPAD    1078* 1082
F6C7 LOPBOT   1091  1094*
F6B7 LOPDAT   1087* 1097
F128 LP01     0215* 0226
F13E LP02     0229* 0238
F0BD LP1P     0138* 0159
F0C1 LP2P     0140* 0142
F057 LPCOL    0058* 0062
F07A LPFND    0076* 0079
F576 LPIN     0871* 0898
F56D LPMID    0867  0868* 0908
F569 LPOUT    0866* 0906
F615 LPPOUT   0980* 0991
F06A LPROW    0067* 0073
F288 MEMBAK   0407  0413  0418*
F1F6 MEMBEG   0192  0349*
F205 MEMCH    0349  0355*
F20B MEMNOW   0356  0359*
F2BF MEMOUT   0433  0443  0448*
E42A MEMSAV   0379  0380  0405  0417  0439  1271*
E419 MNPTR    0043  0152  0350  0467  0573  0789  1158  1263*
E45F NBITS    0978  0990  1298*
F2BA NEWMEM   0353  0419  0426  0431  0445*
F31F NEWREG   0492  0502*
F4AD NEXT     0741* 0750  0752
F44D NEXT1    0670  0673*
F47B NEXT2    0692  0702*
F496 NEXT3    0716  0723*
F773 NMINT    1189* 1331
F484 NOBPT    0689  0708*
```

```
F7C2 SVLOOP   1228* 1232
F7BC SVSTAT   1207  1226* 1242
F7D2 SWINT    1238* 1330
F579 SYNCIN   0865  0873*
F110 SYSFNC   0185  0192*
F4D7 TAPBEG   0198  0783*
F533 TIN      0840* 1070  1078  1087  1101
F5C4 TINDUN   0904  0912*
E433 UA       0546  1107  1163  1276*
E432 UB       0549  1162  1275*
E431 UCC      0558  1176  1227  1274*
F100 UFNK     0176  0182*
F7EC UIRQ     1251* 1329
E43C UIRQV    1251  1282*
E436 UPC      0543  1127  1141  1143  1231  1243  1245  1278*
E43B UPROG    0034  1166  1202  1209  1247  1281*
E42F USP      0025  0555  1135  1205  1226  1233  1240  1273*
E434 UX       0552  1106  1161  1277*
F6C3 VERF     1089  1092*
F593 WITHIN   0881  0883  0884*
E426 XSAV1    0213  0239  0289  0297  0300  0313  1269*
E424 XSAVD    0265  0268  0271  0274  0279  0280  0281  0283  0284  1268*
E428 XTMP1    0229  0234  1270*
```

7.7 D5BUG Listing Description

This description will be useful also for understanding other program listings in software.

Refer to page 001 RESET of the D5BUG. The left most five digits (or field) are called line number, shown under the PAGE heading. These numbers are used to keep track of the line number in the source code that a particular statement was on.

The letter appearing next to a line number (always A in the D5BUG)stands for ASCT, meaning "absolute section". This letter has meaning only when using a relocatable program.

The next four digits are ADDR, or address. This is the actual address in memory for the particular instruction. For instance, line 9 (ØØØØ9A) has FØØØ which is a location in the D5BUG ROM. Successive addresses are not necessarily in progressive order.

The next digits are called OPCODE (one byte) for the instruction. For line ØØØØ9A, the OPCODE is Ø1.

The next digits or field can be zero bytes, one byte, or two bytes. At line ØØØØ9A, there are no bytes to the right of the OPCODE, but at line ØØØ11A, there are two bytes (E3 and FF), and at line ØØØ13A there is one byte (ØØ). This general field is called the OPERAND.

At line ØØØØ9A, the notation NOP requires the OPERAND to be zero bytes. At line ØØØ11A, the notation LDX means OPCODE CE which requires the OPERAND to be two bytes which are E3 and FF. Thus, LDX is load x with E3 and FF as specified.

A special case of the OPERAND field is shown in line ØØØ15A. The OPCODE 26 is for BNE, which is "Branch Not Equal". The value F8 is the relative offset to CLRLOP. The FOO5 is a reference reminder that this is the "branch to" address.

In line ØØØ12A, the label CLRLOP is at address FØØ5, so in line ØØØ15A, the "branch to" address is FØØ5. In the ROM, the FOO5 is not present, but the notation 26 at FØØB is present, along with F8 at FØØC.

At the next line, there is an FØØD and CE. Thus the FØØ5 is not a part of the code, but merely tells where to branch to.

The next field may contain a letter (A, shown in the D5BUG listing) which again is ASCT.

The next field is a label field but is not used very often. It is used as a reference reminder for the assembler.

An asterisk (*) in the listing in the first column of any particular line of the source code, means that line is a comment. No code will be assigned by the assembler.

The mnemonic field (line ØØØØ9A has NOP, no operation), is the English equivalent of the OPCODE.

The mnemonics are all listed in the Instruction Set Summary booklet for the M6800 microprocessor.

Also in this field are assembler directives, such as in line ØØØØ1 - NAM, and OPT (option) in the next line. These do not generate any machine codes.

In line ØØØØ2, CREF and LLEN=80 stand for cross-reference and line length.

Another operand field is shown at line ØØ45A. The notation FOBB is the machine code operand and PUT is the source operand.

An operand can be a name, or a number as in line ØØ32. LDAA #1 means "load A immediate one". The # symbol means "immediate". Also in this line (ØØ32A), FØ27 is the address in memory, 86 is the opcode for load A immediate, 01 is load with zero one.

In line ØØØ38, the symbol $ means hex. In line ØØ4Ø, the % symbol stands for binary. In line ØØ23, 3, X specifies indexed addressing mode.

In line ØØ44, the notation JSR ENNMI creates the code BD F7AE, which is explained by the comment "enable NMI" in the last field to the right.

At the end of the listing (after line 1333), another listing, the cross reference, is shown.

This starts with F789, an address for the label ABORT.

The line numbers (in the previous listing) for ABORT are 1198 and 12Ø2*. The number with the * is the line number where the label was defined, and the other number is the line where ABORT was used.

In this listing, line E442 is BKPNTR. It is referenced to ten places in the previous listing, and is defined in line 1286.

The cross reference listings (table) thus can be used to find any particular label in another listing.