MEK68R2M

PROGRAMMABLE CRT VIDEO INTERFACE

The information in this document has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, such information does not convey to the purchaser of this product described, any license under the patent rights of Motorola, Inc. or others.

First Edition

Motorola Inc., 1979

INTRODUCTION

The MEK68R2M Programmable CRT Video Interface accessory board is designed to provide an interface between an MEK6802D3 Microcomputer Kit, an ASCII keyboard and either a professional video monitor or TV set.* Included with the board is a custom mask 2K ROM.

The MEK68R2M uses an advanced MC6845 CRT Controller Chip for increased user control and reliability. It features the following software programmable line and character formats:

16 lines of 32 characters

16 lines of 64 characters

20 lines of 80 characters

User defined character format

It is plug compatible with the MEK68MB Mother board, the MEK6802D3 Microcomputer Kit, and accessory boards. The MEK68R2M has a standard composite video output, and separate TTL video and sync outputs.

*NOTE: FFC type approval is required when using an RF modulator.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

CHAPTER 1

GENERAL DESCRIPTION AND PREPARATION FOR USE

1.0 General

The following paragraphs provide a physical description of the MEK68R2M, procedures for preparing the board and associated equipment for use, and procedures for operating the overall system. It is suggested that an MEK68MB Mother board be used to interface the MEK68R2M to the associated processor equipment.

1.1 Physical Description

The MEK68R2M is contained on a two-sided printed circuit card, which measures 8.25 inches (209.6 cm) wide by 7.00 inches (177.8 cm) high by 0.062 inches (1.58 cm) thick (board only). Molex connectors mounted to the board form a 60 pin connector. All components are mounted on one side of the board with the tallest component projecting less than one-half inch from the surface of the board.

1.2 Preparation For Use

The MEK68R2M board and associated equipment should be prepared for use as described in the following paragraphs.

1.2.1 Jumper Options

Zero ohm resistors are installed on the board as a means of selecting optional capabilities. The following paragraphs describe the board configuration as shipped from the factory and information necessary to change this configuration (see to Table 1 for optional configurations).

TABLE 1-1. OPTIONAL CONFIGURATIONS

Function	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E12	E13
16 Lines x 32 Characters		X	X						X			
16 Lines x 64 Characters		X		X						X		
20 Lines x 80 Characters	X		X							X		
User Defined	X			X						X		
USA Standard					X							
Europe Standard						X						
Keyboard Strobe H								X				
Keyboard Strobe L							X					

X = Jumper Inserted

1.2.1 Jumper Options (cont'd)

A) The board is shipped with zero ohm resistors in positions labeled E2, E3, and E9 so as to provide a CRT screen format of 16 lines of 32 characters. Move these three resistors as follows to obtain a different format.

Screen Format	Resistors in Positions
16 lines of 64 characters	E2, E4, and E10
20 lines of 80 characters	E1, E3, and E10

1.2.1 Jumper Options (cont'd)

Screen Format	Resistors in Positions
User defined (address of CRTC data table must be entered at $8115 and $8116)	E1, E4, and E10

B) Boards shipped to destinations in the U.S.A. have a zero ohm resistor installed at position E5. Boards shipped to Europe will not have a resistor at this location; instead a zero ohm resistor will be inserted in location E6.

C) All boards shipped will have a zero ohm resistor installed at position E8 to accommodate a keyboard with an acitve high strobe feature. If a keyboard featuring an active low strobe is used, the resistor should be moved to position E7.

1.2.2 D3BUG2 Installation

The routines for operating the MEK68R2M Video Display are included in the D3BUG2 1.0 ROM. This ROM is to be installed in the ROM socket on the MEK 6802D3 Microcomputer Kit. Prepare the D3 kit for use with the R2M board as follows:

A) Move the user RAM from $ØØ8Ø to $818Ø, remove the jumpers from positions E4, and E5, and install them in positions E3 and E6.

B) Install the D3BUG2 1.0 ROM in U29.

1.2.3 ASCII Keyboard Interface

Refer to paragraph 2.4.2 in Chapter 2 for a listing of the socket pin assignments. Use this as a guide to connect the keyboard to socket U39.

1.2.4 Display Unit Connections

Non-modulated Composite Video

A composite video output is available for monitors which accept this signal. A direct connection may be made from J1 on the MEK68R2M to the composite video input of the monitor unit.

Separate TTL Video and Sync

Separate TTL video and sync signals are available for monitors which require TTL level inputs. These signals are available at U40, and may be connected directly to the TTL video, horizontal sync, and vertical sync inputs of the monitor. Refer to Appendix 1 for the output pins.

1.3 Startup Procedure

Proceed as follows to start up the R2M board and associated equipment.

Step 1. Check to see that all equipment is properly interconnected as described in paragraph 1.2.2 above.

Step 2. Apply dc voltage(s) to the equipment.

Step 3. Depress the RESET switch on the MEK6802D3 Microcomputer module and observe that the screen on the associated CRT or TV set is blank except for a cursor and the words D3BUG2 1.0 in the upper left corner.

Step 4. Proceed to paragraph 1.4.

1.4 Operating Example

This routine may be used to demonstrate the MEK68R2M's full character set. When executed, this routine fills the display memory with characters $ØØ

1.4 Operating Example (cont'd)

through $FF, repeating the pattern until the memory is full. Upon completion, the program returns control to D3BUG2 and displays the prompt.

```
                        ORG $0030
*
*           Routine to repetitiously fill the memory with the
*                   full character set
*
0030  CE 9000   FILMEM  LDX #$9000      Point to memory
0033  4F                CLRA           Character = $00
0034  A7 00     LOOP    STAA 0,X       Store character in memory
0036  4C                INCA           Next character  ---> A
0037  08                INX            Next memory Addr ---> X
0038  8C 9FFF            CPX #$9FFF     Done?
003B  26 F7              BNE LOOP       No, continue
003D  7E F0A5            JMP CONTRL     Yes, return to control
```

CHAPTER 2

THEORY OF OPERATION

2.0 General

This chapter describes the overall theory of operation for the various circuits found on the MEK68R2M board and how they relate to circuits in associated equipment.

2.1 Functional Description

The MEK68R2M Programmable CRT Video Interface board was designed to act as an interface between a microprocessor such as Motorola's MEK6802D3 Microcomputer Kit and either a video monitor or TV set. To interface to a TV set, either a UHF or a VHF modulator must be connected to the video output jack, or the TV set must be modified to accept nonmodulated composite video. Provision is made for connecting an ASCII keyboard and/or a light pen.

Figure 2-1 is the system block diagram for the circuits on the R2M board and Appendix 8 is the system schematic. Address, data and control information is accepted from the associated processor. Control and data information is transmitted to the processor and a composite video signal is transmitted to the associated CRT unit. The optional keyboard and light pen provide additional inputs to the board if used. Prior to operation of the board as a CRT interface, certain operating parameters must be defined. This definition is accomplished through the use of the D3BUG2 software program that resides in the associated processor and jumper options on the MEK68R2M board. An initialization sequence is performed

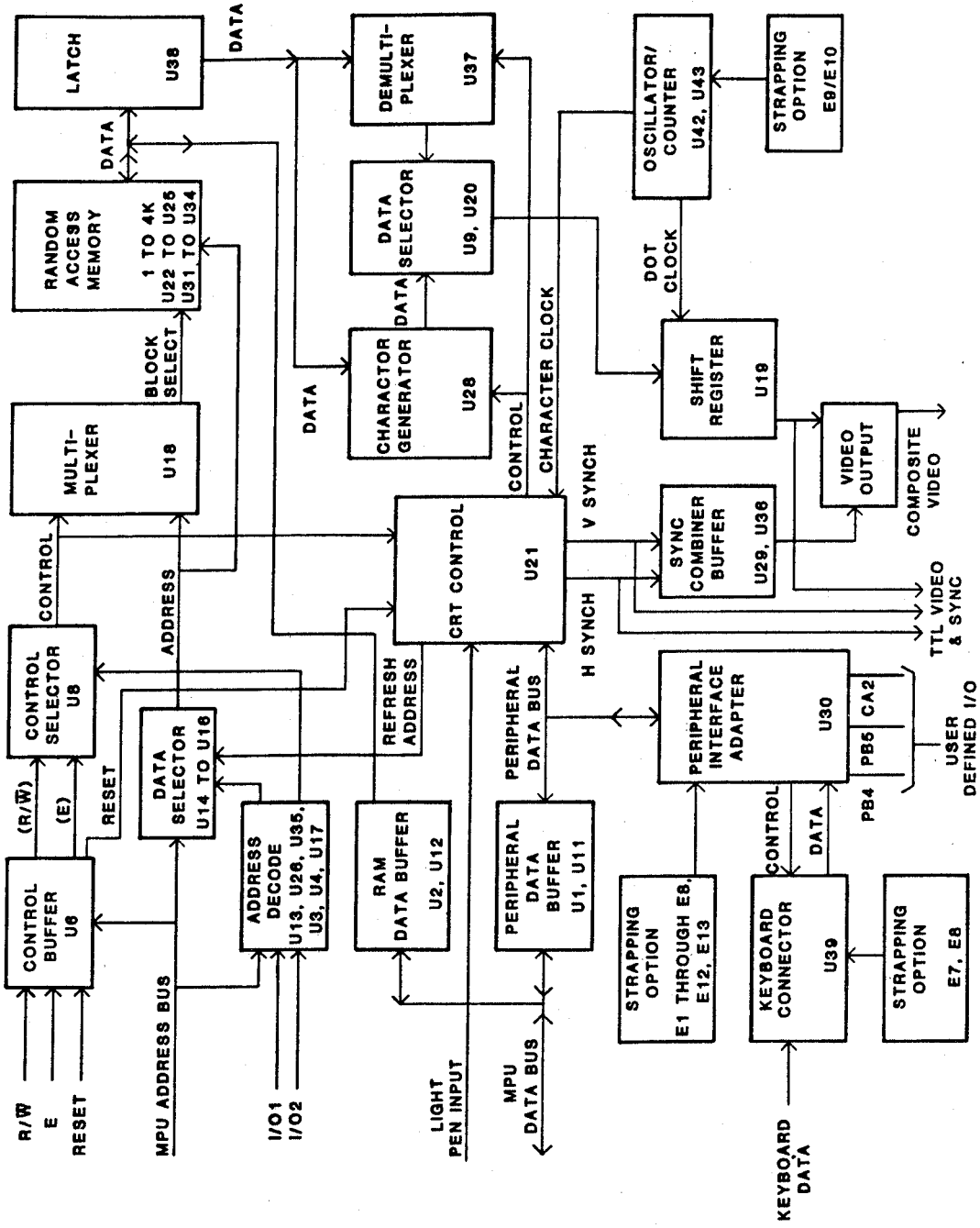


FIGURE 2-1. BLOCK DIAGRAM

2.1 Functional Description (cont'd)

whenever the system is reset. First, the CRT Controller (U21) on the interface board is addressed by the processor and data from the D3BUG2 1.0 initialization table is written into the register files of the CRT controller (CRTC).

This data defines the screen format and the cursor size. Next, the ASCII character code for a blank space is written into all locations in the RAM on the board (U22 and U31 in a basic system and U22 through U25 and U31 in a fully expanded system). Then, the peripheral interface adapter (PIA, U30) is programmed to accept ASCII code character information on the PAØ through PA6 data lines and an interrupt signal input on the CA1 line. After this is accomplished, the board is ready for normal operation. At this time, the words D3BUG2 1.0, a prompt (=), and a cursor appear at the upper left hand corner of the screen on the associated CRT.

2.1.1 Alphanumeric Character Input

When the keyboard is connected to U39, the PIA interfaces the keyboard to the associated processor. (It should be noted that a keyboard is required when using D3BUG2 1.0). When a key on the keyboard is depressed, an $\overline{\text{IRQA}}$ interrupt flag bit is set in the control register of the PIA. The associated processor than scans the contents of the control register, noting that an interrupt flag bit has been set, and the processor then responds by addressing the PIA (address $8044) and reading the seven bits of data applied to PAØ through PA6. This data passes from the DØ through D7 (D7 is read as a "Ø") output of the PIA through buffers U1

2.1.1 Alphanumeric Character Input (cont'd)

and U11 to the output terminals of the R2M board. The processor stores the ASCII code for the character selected. Then the processor addresses the CRTC on the board (U21) and through a series of write and read operations determines the location of the cursor.

The processor then addresses the RAM on the MEK68R2M board, using this 16 bit address (in the range of $9000 through $9FFF) and writes the stored ASCII code into the selected memory location via RAM data bus buffers U2 and U12.

It should be noted that address decode circuits on the MEK68R2M board respond to address bits generated in the associated processor to control the operation of the CRTC, PIA and data buffer circuits discussed above. See paragraph 2.1.6.

2.1.2 Data Display, Alphanumeric Mode

Whenever the address and data buses are not busy receiving data from the processor, the CRT controller sequentially addresses locations in RAM (U22 and U31, basic system; U22 through U25 and U31 through U34, expanded system). The data stored in each memory loation is applied to latch U38. The applied data is clocked through the latch to both character generator U28 and to multiplexer U37. If the data stored in memory is an ASCII character, the most significant bit (MSB) of the eight bits of data is a logic Ø. This low enables data selectors U9 and U20 to pass the output of the character generator to the input of shift register U19.

2.1.2 Data Display, Alphanumeric Mode (cont'd)

The shift register is clocked to produce an output signal that is applied to the video output circuits.

2.1.3 Graphic Mode Input/Display

Data representing a graphic symbol originates in the associated processor and is stored in the RAM on the R2M board in the same general manner as alphanumeric character data. However, the MSB of the eight bits of data is written as a logic 1 to denote that it is a graphic symbol rather than an ASCII character. This high enables data selector U9 and U20 to pass the output of multiplexer U37 to shift register U19. The clocked output of the shift register is applied to the video circuits.

2.1.4 Video Output

The ASCII character or graphic symbol data applied to shift register U19 is shifted out of the register at a clock rate determined by board jumper options (E9 for 32 characters on a line, E10 for 64/80 characters). The output of the shift register is combined with cursor, blanking, horizontal sync and vertical sync information to provide a composite video signal. The composite video and separate TTL video signals are available for connection to a video monitor.

2.1.5 Address Selector Circuit

The address selector circuit consists of four (U8, U14, U15, and U16) quad two-input multiplexers. Two sets of address inputs are applied to the circuit. One set is bits AØ through A11 from the processor and the other is screen refresh address bits MAØ through MA11 from the CRT con-

2.1.5 Address Selector Circuit (cont'd)

troller circuit (U21). When a circuit on the R2M board is addressed, the address decode circut (paragraph 2.1.6) develops an enabling signal. This signal causes the address selector circuit to pass bits AØ thru A11 from the processor to the RAM circuit. When the MEK68R2M is not addressed, the screen refresh address bits from the CRTC are applied to the RAM.

2.1.6 Address Decode Circuit

The address decode circuit performs several functions. Processor address bits A12 thru A15 are applied to inverters U5A and U5B and to NOR gate U13A. When the RAM is addressed (address $9ØØØ through $9FFF), the output of U13A (RAM SELECT) is driven high; this high enables U14 thru U16 in the data selector circuit which causes processor address bits AØ thru A11 to be connected to the RAM. This same high enables U8 in the control selector circuit causing the control signals from the processor (R/$\overline{W}$, E, etc.) to be applied to the RAM. The high output of U13A is inverted by U5E to produce a low that is inputted into U27 in the video circuit, blanking the screen during a memory access.

The high output of U13A and the read/write (R/$\overline{W}$) and enable (E) control signals are applied to inverter U5D, AND gates U35A and U35B, and NAND gates U26B and U26C. When the output of U13A (RAM SELECT) and the enable (E) signal are high, the output of U35B and U26C are controlled by the read/write control signal (R/$\overline{W}$). The data buffers U2 and U12 are enabled so as to connect the output of the RAM to connector P1 (data bus). When the output of U13B and the enable (E) signal are high, the output of U35A

2.1.6 Address Decode Circuit (cont'd)

and U26B are controlled by the read/write signal; buffers U1 and U11 are enabled and R/$\overline{W}$ controls the direction of data flow between P1 and the inputs of the P1A and CRTC.

2.2 Memory Organization

As shipped from the factory, the various circuits of the MEK68R2M are addressable as follows (note that addresses are given in hexadecimal).

Circuit	Assigned Address
CRT Controller	$8Ø42 and $8Ø43
Peripheral Interface Adapter	$8Ø44 through $8Ø47
Random Access Memory (1K Basic)	$9ØØØ through $93FF
Random Access Memory (3K Add-On)	$94ØØ through $9FFF

2.3 Display Data Format

Data to be displayed on the CRT are entered into the display memory as eight bit (DØ through D7) words. The format of each word determines whether the resultant display on a CRT is a graphic symbol or an alphanumeric character. If a graphic symbol is to be displayed, bit D7 is a logic 1, bit D6 is either a logic 1 (symbol half brightness) or a logic Ø (symbol full brightness), and bits D5 through DØ as required to define the symbol. If an alphanumeric character is to be displayed, bit D7 is a logic Ø and bits D6 through DØ are in ASCII code. The format of the data word, hence the symbol or character displayed, is under software control (see Figure 2-2, a and b). Note, even though the MEK68R2M is

2.3 Display Data Format (cont'd)

capable of displaying graphic information, no graphic operating system is included in the D3BUG2 1.0 ROM. Graphic patterns are to be generated under control of a user written program.

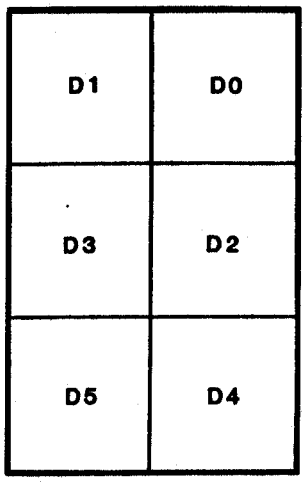


FIGURE 2.2a. BITS CONTROLLING VARIOUS PIXEL POSITIONS WITHIN A CHARACTER BLOCK

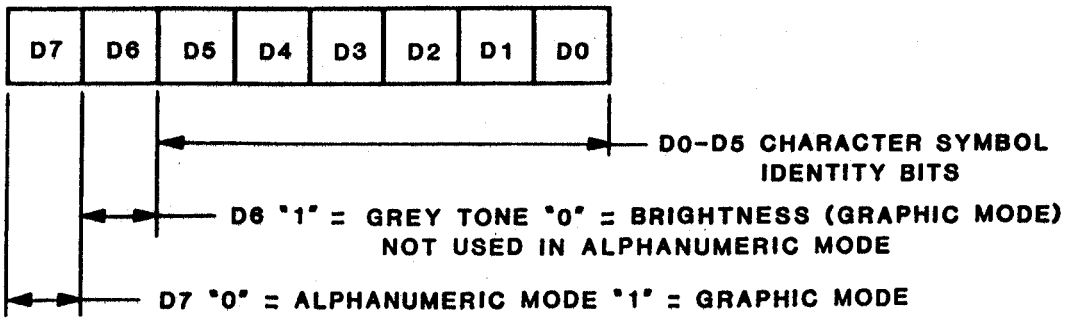


FIGURE 2.2b. FORMAT OF CHARACTER DATA

2.4 Expansion Capabilities

The capability of the MEK68R2M can be expanded by any of the following:

1) Adding Random Access Memory (RAM).

2) Connecting a keyboard to the board.

3) Connecting a light pen to the board.

2.4.1 Add-On RAM

The MEK68R2M board is shipped from the factory with 1K x 8 bits of RAM (Random Access Memory). Two 1K x 4 chips, one in position U22 and one in U31, form this RAM. This RAM is assigned address $9000 through $93FF. Additional RAM can be added as follows:

Type 2114 Chip Installed in	Capacity Added	Assigned Address
U23 and U32	1K x 8	$9400 through $9700
U23 to U25, U32 to U34	3K x 8	$9400 through $9FFF

Revision C and D boards are expanded by adding RAM as described in Figure 2.3 and by changing jumpers E14 thru E17 according to the following table.

TABLE 2.1. JUMPER CONFIGURATION

RAM On Board	Jumpers Installed			
	E14	E15	E16	E17
1K x 8	Yes	No	No	Yes
2K x 8	No	Yes	No	Yes
4K x 8	No	Yes	Yes	No

2.4.1 Add-On RAM (cont'd)

NOTE

When adding 1K of RAM to the R2M board, of revision A or B, a trace on solder side of U18 going to Pin 1 must be cut. A wire must be soldered from U18 Pin 1 to a pad marked SA10 (see Figure 2-3).

When fully expanding the RAM, the trace at Pin 2 of U18 must also be cut and a jumper connected from U18 Pin 2 to SA11 (see Figure 2-3).

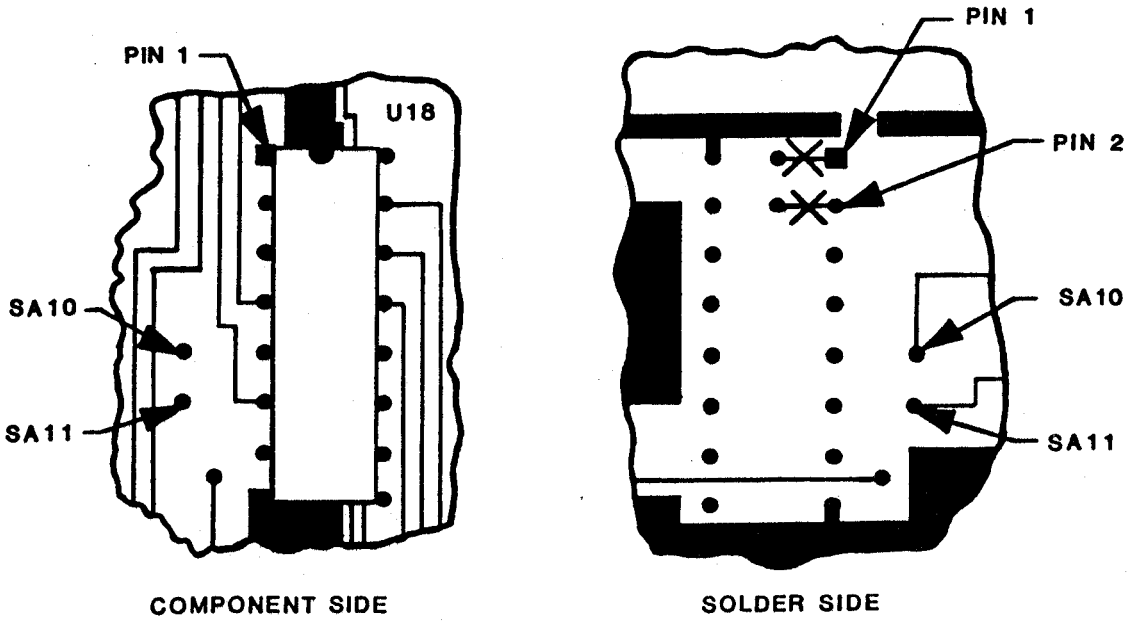


FIGURE 2.3. RAM EXTENSION

2.4.2 Keyboard

An ASCII keyboard can be connected to keyboard dip socket U39 using the following pin connector assignments.

Signal Name	Keyboard Dip Socket Pin	Use
KD1	1	Keyboard data bit 1 (LSB)
KD2	2	Keyboard data bit 2
KD3	3	Keyboard data bit 3

2.4.2 Keyboard (cont'd)

Signal Name	Keyboard Dip Socket Pin	Use
KD4	4	Keyboard data bit 4
KD5	5	Keyboard data bit 5
KD6	6	Keyboard data bit 6
KD7	7	Keyboard data bit 7
PB4	8	PB4 user defined I/O
KS	9	Keyboard Strobe Jumper option active Hi or Lo
--	10	N/C
-12 V	11	Voltage Supplies
-5 V	12	Voltage Supplies
GND	13	Voltage Supplies
+12 V	14	Voltage Supplies
GND	15	Voltage Supplies
+5 V	16	Voltage Supplies

Keyboard data is applied to the Section A Peripheral Data inputs (PAØ through PA6) of the PIA (U30). The keyboard strobe signal is applied to the CA1 input of the PIA (via jumper E7, if the strobe signal is active low, or E8 if the strobe signal is active high) and acts to set the interrupt flag of the control register A in the PIA.

CHAPTER 3

SOFTWARE DESCRIPTION (D3BUG2 1.0 EXPANSION ROM)

3.0 Introduction

The MEK68R2M Video Display board was designed to display information supplied by the MEK6802D3 Microcomputer board. D3BUG2 provides an easy to use interface between the D3 and the R2M boards. D3BUG2 allows the R2M user to manipulate all of the resources of the D3 and the R2M boards with any other MOKEP board in the user's system. The user has the capability to write programs and debug them easily using D3BUG2. Many of the subroutines which make up D3BUG2 are available for use in the user's programs, thus relieving the user of some software burden. Refer to Appendix 2 for the D3BUG2 listing and Appendix 3 for the flow charts.

D3BUG2 uses many of the routines in D3BUG, which is supplied on the MEK6802D3, to perform many of its own functions. D3BUG2 expands D3BUG into a 4K monitor capable of using the R2M's video display and an ASCII keyboard. Since D3BUG is still present the hex keypad and display are still available to the D3BUG2 user. Control can be passed at will by depressing a key on either the hex keypad or the ASCII keyboard.

D3BUG2 is a collection of debug routines which allows the user to develop software systems in the MC6802's machine language through the use of a CRT display and an ASCII keyboard or an RS-232 Terminal. The commands available include a memory dump command with ASCII equivalants, a register display command that allows four optional user defined registers to be displayed in addition to the MC6802's registers, an offset load command

which allows a program to be relocated anywhere in memory (see note), single and multiple instruction trace, 300 or 1200 baud tape punch, load, verify, and multiple screen formats, including European formats. Many subroutines developed for D3BUG2 are available to the user, including subroutines to manipulate the paging scheme implemented in the MOKEP system. D3BUG2 is a very powerful de-bugging tool with features that were at one time found only on more expensive development systems. It provides the basis of a very powerful software system supporting more advanced operating systems and high level languages.

3.1 Monitor Commands

The D3BUG2 commands are described in detail in order that the user may obtain full use of their abilities. The following commands are grouped by function and where possible alphabetically.

Command	Description
B	Displays the user's breakpoint table, with up to eight breakpoint addresses displayed.
D	Deletes all breakpoints from the table.
S	Sets a breakpoint in the table. Example: S 1 LF ØØØ1 A breakpoint at memory location ØØØ1 HEX is placed on the table.

NOTE: The program is only relocated physically. It may not execute correctly at its relocated address.

U Delete the specified breakpoint from the table.

Example: = U 1 LF

=

A breakpoint that was placed in the table by the S command is now deleted leaving the table empty, as signified by the blank line after the command and before the prompt.

C Continue from the present program counter address. This command is used principally for continuing from an encountered breakpoint, but may also be used by setting the program counter manually. The C command inserts all breakpoints in the table into the program after executing the instruction indicated by the program counter. Normal execution of the program resumes at the next instruction. Execution continues until another breakpoint is encountered.

CAUTION: Breakpoints are only removed when a breakpoint is encountered or the subroutine RMBKPT ($FDFD) is executed. It is suggested that RMBKPT be setup as user function 1 (see optional display registers discussion) when debugging programs.

G Go to the specified memory location and begin execution. Any breakpoints in the table are entered before execution begins. Execution stops when a breakpoint is encountered, the reset key is depressed, or the Escape command is given

from the MEK6802D3's keypad. The reset key clears the breakpoint table, but the Escape command does not. If the Escape command was used, the breakpoints can be removed by executing the RMBKPT subroutine (see CAUTION in the C command section).

E Examine a block of memory starting at the entered address with ASCII equivalents. The main use of this command is for observing large parts of programs and for searching for ASCII tables. The amount of memory examined is dependent upon the R2M's display format. To display another line, depress the space bar. The command monitor may be re-entered by depressing the carriage return key.

L Load a previously recorded program from an audio tape recorded using the MEK68I0 board. Two formats are available and are prompted by the following message (3 or 12). Either a 3 may be entered to choose the 300 baud format, which is J-BUG compatible, or a 12 may be entered to choose the 1200 baud format, which provides check-sum error detection not provided in the 300 baud format. A line feed or a space bar will start the load program. The user may abort command and return to the monitor by depressing the carriage return key. The user must position the tape and set the baud rate switch of the MEK68I0 to the desired speed.

3.1 Monitor Commands (cont'd)

O Offset load a program recorded on audio tape anywhere in memory. This command executes essentially the same as the L command, except with the following differences. After selecting the appropriate format, a prompt for the new address is displayed (see example). The address entered is the new starting address of the loaded program. After loading is complete, the original starting address of the program is displayed.

Example:

```
= 0  (3 or 12)? 12  LF
Beg Adr  = Ø        LF
True Adr = FØØØ
=
```

P Punch a program onto an audio tape using the MEK68I0 board. The tape format is selected as in the L command (described in the previous page). After selecting the appropriate format, a prompt for the starting address of the program to be punched is displayed (see example). The desired address is entered and a prompt for the ending address is displayed. After entering the ending address, depressing the line feed or space bar key will start the execution of the punch program. When completed the command prompt will re-appear. The recorder should be in the record mode before entering the ending address.

Example:

```
= P (3 or 12)? 12 LF
Beg Adr = FØØØ LF
End Adr = F7FF LF
=
```

V Verify a previously recorded program with the memory from which it was recorded. Tape format selection is as described in the L command. The Verify command does a byte for byte comparison between the recorded version on tape and the actual program in memory. If a mismatch occurs, then the following message is displayed:

CKSUM? XXXX

where XXXX is the hexidecimal address of the byte where the mismatch occurred.

Example:

```
= V (3 or 12)? 12 LF
=
```

In this case there was no mismatch and the command prompt reappears.

N Trace one instruction at a time while displaying the state of the registers after each trace. The program counter contains the address, which can be put there manually through the use of the R command or by the encountering of a breakpoint in a

running program. There are two basic register displays that can result from the execution of this command depending on the display format in use. If a 16 x 32 character display is being used, the following register display will occur.

```
= N
CC   BB   A    X      PC     S      RAM  ROM
XX   XX   XX   XXXX   XXXX   XXXX   X    X
=
```

where,

CC is the Condition Code Register

B is the B-Accumulator

A is the A-Accumulator

X is the Index Register

PC is the Program Counter

S is the Stack Pointer

RAM is the RAM Page Register

ROM is the ROM Page Register

If a 16 x 64 or a 20 x 80 display is being used the register display appears as shown below.

```
= N
CC  B   A   X     PC    RAM  ROM   REG1  REG2  REG3  REG4
XX  XX  XX  XXXX  XXXX  X    X     XXXX  XXXX  XXXX  XXXX
=
```

where,

REG1 is the first optional register

REG2 is the second optional register

REG3 is the third optional register

REG4 is the fourth optional register

And the rest are as previously defined.

The optional registers are two byte memory locations selected by the user. The addresses of these locations are entered with the "R" command.

T Traces one or more instructions up to 256 ($FF). The number of instructions to trace must be entered in hex.

Example:

```
= T 2 LF
CC  B   A   X     PC    S     RAM  ROM
XX  XX  XX  XXXX  XXXX  XXXX  X    X
CC  C   A   X     PC    S     RAM  ROM
XX  XX  XX  XXXX  XXXX  XXXX  X    X
=
```

For a 16 x 32 display.

M Displays the data at the entered address.

Example:

```
= M FØØØ   LF
= FØØØ   55
```

To advance to the next memory location depress the line feed key.

Example:

FØØØ 55 LF

FØØ1 AA

To back up to the previous location depress the M key.

Example:

FØØØ 55 LF

FØØ1 AA M

FØØØ 55

To change the data at the present location enter the new data and depress the space bar (SB).

Example:

FØØØ 55 AA SB ?

The ? signifies that no RAM is at that location. In this case the first location in the D3BUG2 ROM is being displayed.

Example:

= M Ø LF

ØØØØ XX 55 SB 55

Here memory location ØØØØ $_{HEX}$ is opened and changed to a 55 $_{HEX}$ as signified by the 55 after the SB. The XX after ØØØØ means unknown display. The memory change will allow any number of hex characters to be entered but will only accept the last two entered.

Example:

ØØØØ 55 1234567 SB 67

Offset command, O, is a command that can be invoked while using the memory change, M, command. This is very helpful in calculating the offset (second byte) of a branch instruction to its destination address. To use this command while in a memory change function depress O and enter a 16 bit Hex value which represents the absolute address of the destination of the branch instruction. Depress LF and the monitor will display the offset value if within the -127 to 128 byte range and redisplay the address and current value of previous line. If the routine finds the resultant to be out of range, a FF will be displayed. If the offset is within range, the value is automatically stored into memory.

Examples:

=	M	7F	O	Ø	LF
	ØØ7F		8Ø	O	FF LF
	ØØ7F		7F	O	1ØØ
	ØØ7F		FF	SB	7F

Here the two extremes are shown. The offset from memory location ØØ7F $_{HEX}$ to ØØØØ $_{HEX}$ is 8Ø $_{HEX}$. The offset from memory location ØØ7F $_{HEX}$ to ØØFF $_{HEX}$ is 7F $_{HEX}$. Memory location Ø1ØØ is out of branching range from ØØ7F as indicated by the FF_{HEX} result. This result is not entered into memory as shown in the last line of the example.

R Displays the user's registers including the RAM and ROM page register and, if in the 16 x 64 or the 20 x 80 character display, four user defined registers. The registers may then be changed one at a time. The user defined registers contain the addresses of some memory locations whose contents are of interest to the user when tracing a program. The last three registers can also be used for storing the jump addresses of the user defined functions which are described later. The register display command may be exited at anytime by depressing the carriage return key. A typical editing example is shown below:

Example:

```
= R
CC   B   A   X     PC    S     RAM   ROM
XX   XX  XX  XXXX  XXXX  XXXX   X     X
XX   Ø   LF
XX   Ø   LF
XX   Ø   LF
XXXX Ø   LF
XXXX Ø   LF
XXXX 81FF  LF
RAMPG =  X  Ø   LF
ROMPG =  X  Ø   LF
XXXX  REG = XXXX  FØØØ  LF
```

3.1. Monitor Commands (cont'd)

```
XXXX  REG = XXXX  F002  LF
XXXX  REG = XXXX  F004  LF
XXXX  REG = XXXX  F006  LF
=  R
CC   B    A    X      PC     S      RAM   ROM
00   00   00   0000   0000   81FF   0     0
00   CR
=
```

! Execute the user's function whose address is stored in the user defined register 2.

" Execute the user's function whose address is stored in the user defined register 3.

Execute the user's function whose address is stored in the user defined register 4.

CTL E Clear the MEK68R2M's display and home the cursor.

3.2 D3BUG2 Operation

D3BUG2 is a command monitor program supplied with the MEK68R2M, which allows the user of that board to develop MC6802 machine language programs. Development tools include single and multiple instruction trace, breakpoint traps for interrupting programs under software control, and display memory either one location at a time of blocks of locations with their ASCII equivalents displayed also. With these and other features in D3BUG2, the user can enter, display, debug, and execute his or hers programs.

MOTOROLA INC.

MEK68R2M CRT VIDEO INTERFACE CORRIGENDUM

Please make the following corrections to the MEK68R2M Manual.

Pages 3-13 & 3-14 Replace Test Program (Worm) with:

TEST PROGRAM (WORM)

```
00001A 0000                        ORG     $0000
00002A 0000    0004  A PRESNT FDB     WORM
00003A 0002    0020  A NEXT   FDB     ENDWRM
00004A 0004 DE 02    A WORM   LDX     NEXT      GET THE START ADDR.
00005A 0006 96 02    A        LDAA    NEXT      GET MOST SIG. BYTE OF ADDR
00006A 0008 81 80    A        CMPA    #$80      REACHED THE TOP OF RAM
00007A 000A 27 09 0015        BEQ     ARN       YES START OVER
00008A 000C 86 55    A        LDAA    #$55      LOAD TEST PATTERN
00009A 000E A7 1C    A        STAA    $1C,X     STORE IT AT STARTING ADDR.
00010A 0010 A1 1C    A        CMPA    $1C,X     HAS IT CHANGED?
00011A 0012 26 01 0015        BNE     ARN       YES REACHED THE END OF MEM
00012A 0014 39                RTS               NO, BACK TO THE MAIN PROGR
00013A 0015 86 47    A ARN    LDAA    #'G       A SUCCESSFUL TEST
00014A 0017 BD F156  A        JSR     OUTCHR    TELL THE WORLD
00015A 001A BD F363  A        JSR     EXKEY     ABORT?
00016A 001D 7E 8189  A        JMP     INIT      NO, CONTINUE ON
00017          0020  A ENDWRM EQU     *
00019A 8180                        ORG     $8180
00020A 8180 BD F6A7  A DISPLA JSR     CLRSCR
00021A 8183 CE 0000  A        LDX     #PRESNT
00022A 8186 BD F140  A        JSR     OUT4HS
00023A 8189 CE 0004  A INIT   LDX     #WORM     INITIALIZE WORM LOCATION
00024A 818C DF 00    A        STX     PRESNT
00025A 818E CE 0020  A        LDX     #ENDWRM
00026A 8191 DF 02    A        STX     NEXT
00027A 8193 CE 9000  A LOOP   LDX     #$9000
```

3501 Ed Bluestein Boulevard, Austin, Texas 78721/(512)928-6000

TEST PROGRAM (WORM) (cont'd)

```
00028A 8196 96 00    A          LDAA   PRESNT   DISPLAY LOCATION OF WORM
00029A 8198 BD F122  A          JSR    UNPACK
00030A 819B BD F114  A          JSR    HEXASC
00031A 819E A7 00    A          STAA   0,X
00032A 81A0 17                  TBA
00033A 81A1 BD F114  A          JSR    HEXASC
00034A 81A4 A7 01    A          STAA   1,X
00035A 81A6 96 01    A          LDAA   PRESNT+1
00036A 81A8 BD F122  A          JSR    UNPACK
00037A 81AB BD F114  A          JSR    HEXASC
00038A 81AE A7 02    A          STAA   2,X
00039A 81B0 17                  TBA
00040A 81B1 BD F114  A          JSR    HEXASC
00041A 81B4 A7 03    A          STAA   3,X
00042A 81B6 C6 1D    A          LDAB   #LENGTH
00043A 81B8 DE 00    A LOOP1    LDX    PRESNT
00044A 81BA A6 00    A          LDAA   0,X
00045A 81BC 08                  INX
00046A 81BD DF 00    A          STX    PRESNT
00047A 81BF DE 02    A          LDX    NEXT
00048A 81C1 A7 00    A          STAA   0,X
00049A 81C3 08                  INX
00050A 81C4 8C 8000  A          CPX    #$8000
00051A 81C7 26 03 81CC          BNE    ARN1
00052A 81C9 7E 0015  A          JMP    ARN
00053A 81CC DF 02    A ARN1     STX    NEXT
00054A 81CE 5A                  DECB
00055A 81CF 26 E7 81B8          BNE    LOOP1
00056A 81D1 DE 00    A          LDX    PRESNT
00057A 81D3 AD 00    A          JSR    0,X
00058A 81D5 20 BC 8193          BRA    LOOP
00059          001D  A LENGTH   EQU    $1D
00060          F6AC  A CLRSCR   EQU    $F6A7
00061          F140  A OUT4HS   EQU    $F140
00062          F122  A UNPACK   EQU    $F122
00063          F114  A HEXASC   EQU    $F114
00064          F156  A OUTCHR   EQU    $F156
00065          F366  A EXKEY    EQU    $F363
00066                           END
TOTAL ERRORS 00000
```

3.2 D3BUG2 Operation (cont'd)

In order to better understand the features of D3BUG2 an example program is needed. The example program is a memory test which ulitizes a test algorithm called Worm. The Worm algorithm causes a small program to be moved and executed throughout the computers available RAM memory. In this program the Worm is implemented by a main program which stays at one place and a subroutine which the main program moves through memory and calls to exercise the memory in which the subroutine resides. This program uses the MEK68R2M to display the results of the test. The listing of the test program is given below.

TEST PROGRAM (WORM)

```
00001A 0000                        ORG     $0000
00002A 0000    0004  A PRESNT     FDB     WORM
00003A 0002    0020  A NEXT      FDB     ENDWRM
00004A 0004 DE 02    A WORM      LDX     NEXT        GET THE START ADDR.
00005A 0006 96 02    A           LDAA    NEXT        GET MOST SIG. BYTE OF ADDR
00006A 0008 81 80    A           CMPA    #$80        REACHED THE TOP OF RAM
00007A 000A 27 09 0015           BEQ     ARN         YES START OVER
00008A 000C 86 55    A           LDAA    #$55        LOAD TEST PATTERN
00009A 000E A7 1C    A           STAA    $1C,X       STORE IT AT STARTING ADDR.
00010A 0010 A1 1C    A           CMPA    $1C,X       HAS IT CHANGED?
00011A 0012 26 01 0015           BNE     ARN         YES REACHED THE END OF MEM
00012A 0014 39                   RTS                 NO, BACK TO THE MAIN PROGR
00013A 0015 86 47    A ARN       LDAA    #'G         A SUCCESSFUL TEST
00014A 0017 BD F156  A           JSR     OUTCHR      TELL THE WORLD
00015A 001A BD F366  A           JSR     EXKEY       ABORT?
00016A 001D 7E 8189  A           JMP     INIT        NO, CONTINUE ON
00017          0020  A ENDWRM    EQU     *
00019A 8180                      ORG     $8180
00020A 8180 BD F6AC  A DISPLA    JSR     CLRSCR
00021A 8183 CE 0000  A           LDX     #PRESNT
00022A 8186 BD F140  A           JSR     OUT4HS
00023A 8189 CE 0004  A INIT      LDX     #WORM       INITIALIZE WORM LOCATION
00024A 818C DF 00    A           STX     PRESNT
00025A 818E CE 0020  A           LDX     #ENDWRM
00026A 8191 DF 02    A           STX     NEXT
00027A 8193 CE 9000  A LOOP      LDX     #$9000
```

TEST PROGRAM (WORM) (cont'd)

```
00028A 8196 96 00    A        LDAA   PRESNT   DISPLAY LOCATION OF WORM
00029A 8198 BD F122  A        JSR    UNPACK
00030A 819B BD F114  A        JSR    HEXASC
00031A 819E A7 00    A        STAA   0,X
00032A 81A0 17                TBA
00033A 81A1 BD F114  A        JSR    HEXASC
00034A 81A4 A7 01    A        STAA   1,X
00035A 81A6 96 01    A        LDAA   PRESNT+1
00036A 81A8 BD F122  A        JSR    UNPACK
00037A 81AB BD F114  A        JSR    HEXASC
00038A 81AE A7 02    A        STAA   2,X
00039A 81B0 17                TBA
00040A 81B1 BD F114  A        JSR    HEXASC
00041A 81B4 A7 03    A        STAA   3,X
00042A 81B6 C6 1D    A        LDAB   #LENGTH
00043A 81B8 DE 00    A LOOP1  LDX    PRESNT
00044A 81BA A6 00    A        LDAA   0,X
00045A 81BC 08                INX
00046A 81BD DF 00    A        STX    PRESNT
00047A 81BF DE 02    A        LDX    NEXT
00048A 81C1 A7 00    A        STAA   0,X
00049A 81C3 08                INX
00050A 81C4 8C 8000  A        CPX    #$8000
00051A 81C7 26 03 81CC        BNE    ARN1
00052A 81C9 7E 0015  A        JMP    ARN
00053A 81CC DF 02    A ARN1   STX    NEXT
00054A 81CE 5A                DECB
00055A 81CF 26 E7 81B8        BNE    LOOP1
00056A 81D1 DE 00    A        LDX    PRESNT
00057A 81D3 AD 00    A        JSR    0,X
00058A 81D5 20 BC 8193        BRA    LOOP
00059          001D  A LENGTH EQU    $1D
00060          F6AC  A CLRSCR EQU    $F6AC
00061          F140  A OUT4HS EQU    $F140
00062          F122  A UNPACK EQU    $F122
00063          F114  A HEXASC EQU    $F114
00064          F156  A OUTCHR EQU    $F156
00065          F366  A EXKEY  EQU    $F366
00066                         END
TOTAL ERRORS 00000
```

Begin by entering the program into memory. This is done with the M command. First enter the main program starting at 8180 HEX which lies outside the RAM to be tested (note section 2.1). The display should appear as shown below:

```
M   8180  SB  XX  BD  LF
8181  XX  F6  LF
```

Where the XX represents the data at location 8180 HEX. Now the rest of the main program can be entered. Upon completion of this task enter the Worm subroutine starting at ØØØØ HEX in the same manner. If a mistake is made while entering the two programs, there exist a number of ways to correct them in D3BUG2. If the wrong hex number is entered and the line feed has not been depressed the user may continue entering hex characters until the correct two are entered.

Example:

```
8182  XX  AØ  AC  LF
8183  XX
```

Notice AØ is incorrect and is fixed by simply entering the correct two characters AC.

If the wrong entry was made and the line feed key was depressed, this can be corrected by depressing the M key and entering the correction:

Example:

```
8183  XX  C1  LF
8184  XX  M
8183  C1  CE  SB  CE  LF
8184  XX
```

If the wrong entry was made and the carriage return key was depressed, the correction can be made using the M command:

Example:

```
8184   XX  ØF  CR
= M 8184  ØF  ØØ  SB  ØØ  LF
8185   XX
```

After entering the two programs they could be saved on tape using the P command if an MEK68I0 board is present in the system. The programs should be visually verified for correctness.

Now that the programs are in memory, debugging can begin. Starting with the main program, the first instruction to be executed is a call to a subroutine in D3BUG2. Set a breakpoint at 8186 $_{HEX}$, as shown below, in order to observe what the subroutine CLRSCR does.

```
= S  8186   LF
     8186
= G  818Ø   LF
```

Set the breakpoint and execute the program. The CRT display should clear and the following display should appear.

CC	B	A	X	PC	S	RAM	ROM
C4	1Ø	ØF	ØØØØ	8186	81FF	Ø	Ø

Where the PC register holds the address where the breakpoint is encountered and the X register is cleared with the instruction LDX #PRESNT. The next instruction is again a call to a subroutine (OUT4HS) in D3BUG2. Again, use a breakpoint set at INIT ($8189) to observe the results.

```
= D          Clear all breakpoints

= S 8189 LF

8189

= G 818Ø LF
```

Here all breakpoints are cleared and a new one set at 8189 HEX. The program is executed again with the following display appearing on the screen.

```
FFFF
CC    B    A    X     PC    S     RAM  ROM
CØ    10   Ø5   ØØØ2  8189  81FF  Ø    Ø
=
```

A breakpoint has occurred where the contents of PRESNT ($ØØØØ) has been printed on the CRT. The X register now points to Next ($ØØØ2) because OUT4HS increments the index register by two before it returns. The PC register holds the address where the breakpoint was encountered ($8189). This completes the screen initialization and paves the way to check program initialization by single instruction traces.

```
= N
CC    B    A    X     PC    S     RAM  ROM
CØ    1Ø   Ø5   ØØØ4  818C  81FF  Ø    Ø
= N
CC    B    A    X     PC    S     RAM  ROM
CØ    1Ø   Ø5   ØØØ4  818E  81FF  Ø    Ø
= N
```

```
CC   B    A    X      PC     S      RAM   ROM
CØ   1Ø   Ø5   ØØ2Ø   8191   81FF   Ø     Ø
= N
CC   B    A    X      PC     S      RAM   ROM
CØ   1Ø   Ø5   ØØ2Ø   8193   81FF   Ø     Ø
```

Now, to check the results.

```
= M Ø SB ØØ LF
ØØØ1 Ø4 LF
ØØØ2 ØØ LF
ØØØ3 2Ø CR
=
```

The correct starting and ending addresses of Worm have been set-up. Now, to debug Loop. The first two instructions can be traced.

```
= N
CC   B    A    X      PC     S      RAM   ROM
C8   1Ø   Ø5   9ØØØ   8196   81FF   Ø     Ø
= N
CC   B    A    X      PC     S      RAM   ROM
C4   1Ø   ØØ   9ØØØ   8198   81FF   Ø     Ø
```

The index register holds the beginning address of the MEK68R2M's display memory and the A-accumulator holds the most significant byte of the present address of the Worm program. The next two instructions are calls to subroutines (UNPACK and HEXASC) which will unpack the two hex

characters and convert them to ASCII. Set a breakpoint at 819E HEX to stop execution at the correct point.

```
= D
= S 819E
819E
= G 818Ø
```

The following display results:

```
FFFF
CC   B   A   X     PC    ROM   ROM
CØ   ØØ  3Ø  9ØØØ  819E  Ø     Ø
=
```

Where the A-accumulator holds the ASCII equivalent of the most significant hex character and the B-accumulator holds the least significant hex character. Now trace the next two instructions using the multiple trace instruction.

```
= T 2 LF
CC   B   A   X     PC    S     RAM   ROM
CØ   ØØ  3Ø  9ØØØ  81AØ  81FF  Ø     Ø
CC   B   A   X     PC    S     RAM   ROM
C4   ØØ  ØØ  9ØØØ  81A1  81FF  Ø     Ø
=
```

The ASCII character Ø has been placed at 9ØØØ HEX and the contents of B have been placed in A.

```
= M 9ØØØ SB 3Ø CR
=
```

Notice that the FFFF $_{HEX}$ at top of the screen is now ØFFF $_{HEX}$. Now to display the second character that has been placed in the A-accumulator.

```
= D
= S  81A6  LF
81A6
= C        LF
```

With the following display appearing,

```
ØØFF
CC    B    A    X     PC    S     RAM  ROM
CØ    ØØ   3Ø   9ØØØ  81A6  81FF  Ø    Ø
=
```

The hex Ø has been converted to the ASCII equivalent 3Ø $_{HEX}$ and is stored at 9ØØ1 $_{HEX}$.

```
= M   9ØØ1  SB   3Ø   CR
=
```

Trace the next instruction to retrieve the least significant byte of the address which in this case is Ø4.

```
= N
CC    B    A    X     PC    RAM  ROM
CØ    ØØ   Ø4   9ØØØ  81A8  Ø    Ø
```

Now to display it in the same manner as the previous byte.

```
= S   81B6  LF
81B6
= C
```

CC	B	A	X	PC	S	RAM	ROM
CØ	Ø4	34	9ØØØ	81B6	81FF	Ø	Ø

The upper left hand of the CRT should now display:

ØØØ4

Which is the present address of the Worm program. The main program uses this part of the loop to update the present address of the Worm as it (the Worm) moves through memory.

Now trace LOOP 1, which moves the Worm program. First, initialize the count.

= N

CC	B	A	X	PC	S	RAM	ROM
CØ	1D	34	9ØØØ	81B8	81FF	Ø	Ø

The B-accumulator holds the count 1D $_{HEX}$. Now to trace the rest of the loop.

= T 7 LF

CC	B	A	X	PC	S	RAM	ROM
CØ	1D	34	ØØØ4	81BA	81FF	Ø	Ø
CC	B	A	X	PC	S	RAM	ROM
C8	1D	DE	ØØØ4	81BC	81FF	Ø	Ø
CC	B	A	X	PC	S	RAM	ROM
C8	1D	DE	ØØØ5	81BD	81FF	Ø	Ø
CC	B	A	X	PC	S	RAM	ROM
CØ	1D	DE	ØØØ5	81BF	81FF	Ø	Ø

```
CC    B    A    X     PC    S     RAM  ROM
CØ    1D   DE   ØØ2Ø  81C1  81FF  Ø    Ø
CC    B    A    X     PC    S     RAM  ROM
C8    1D   DE   ØØ2Ø  81C3  81FF  Ø    Ø
CC    B    A    X     PC    S     RAM  ROM
C8    1D   DE   ØØ21  81C4  81FF  Ø    Ø
= N
CC    B    A    X     PC    S     RAM  ROM
CA    1D   DE   ØØ21  81C7  81FF  Ø    Ø
= N
CC    B    A    X     PC    S     RAM  ROM
CA    1D   DE   ØØ21  81CC  81FF  Ø    Ø
= N
CC    B    A    X     PC    S     RAM  ROM
CØ    1D   DE   ØØ21  81CE  81FF  Ø    Ø
= N
CC    B    A    X     PC    S     RAM  ROM
CØ    1C   DE   ØØ21  81CF  81FF  Ø    Ø
= N
CC    B    A    X     PC    S     RAM  ROM
CØ    1C   DE   ØØ21  81B8  81FF  Ø    Ø
```

The first byte of the Worm program (DE $_{HEX}$) is transferred from its present starting address ØØØ4 $_{HEX}$ to its new address ØØ2Ø $_{HEX}$. It also checks to see if the top of memory has been reached, which it has not, so on to the next byte.

The count in the B-accumulator is decremented by one and, since the result is not zero, the Loop is repeated. Set a breakpoint at the last instruction of the Loop (81CF HEX) to check that the loop is executed the correct number of times.

```
= D
= S   81CF  LF
81CF
= C
CC    B     A     X      PC     S      RAM   ROM
CØ    1B    Ø2    ØØ22   81CF   81FF   Ø     Ø
= C
CC    B     A     X      PC     S      RAM   ROM
CØ    1A    96    ØØ23   81CF   81FF   Ø     Ø
```

The A-accumulator holds the correct second and third bytes, respectively, and the count in the B-accumulator is decremented by one each time. Continue executing the C command until the B-accumulator holds a zero.

```
= C
CC    B     A     X      PC     S      RAM   ROM
CØ    Ø1    89    ØØ3C   81CF   81FF   Ø     Ø
= C
CC    B     A     X      PC     S      RAM   ROM
C4    ØØ    DE    ØØ3D   81CF   81FF   Ø     Ø
=
```

Notice that the last byte to be moved is DE, which is not the last byte of the Worm program (89 $_{HEX}$). The last byte is actually transferred the step before, which means the Loop count (1D $_{HEX}$) is one too large. This can be fixed by changing the instruction which initializes the count.

```
= M   81B7 SB   1D   1C   CR
= G   818Ø LF
```

Which displays,

```
ØØØ4
CC    B    A    X     PC    S     RAM  ROM
CØ    1B   DE   ØØ21  81CF  81FF  Ø    Ø
```

Repeat the Loop until the B-accumulator contains ØØ.

```
= C
CC    B    A    X     PC    S     RAM  ROM
CØ    1A   Ø2   ØØ22  81CF  81FF  Ø    Ø
= C
.
.
.
CC    B    A    X     PC    S     RAM  ROM
C4    ØØ   89   ØØ3C  81CF  81FF  Ø    Ø
```

The Loop is now executed the correct number of times and the Worm has been transferred to memory locations ØØ2Ø $_{HEX}$ to ØØ3B $_{HEX}$. The last part of Loop executes the worm.

```
= N
CC    B    A    X     PC    S     RAM  ROM
C4    ØØ   89   ØØ3C  81D1  81FF  Ø    Ø
= N
CC    B    A    X     PC    S     RAM  ROM
CØ    ØØ   89   ØØ2Ø  81D3  81FF  Ø    Ø
```

Loop is concluded when the B-accumulator decrements to zero and the index points to the new starting address of the worm program.

```
= N
CC    B    A    X     PC    S     RAM  ROM
CØ    ØØ   89   ØØ2Ø  ØØ2Ø  81FD  Ø    Ø
```

Call the Worm program. Notice that the stack has been decremented by two by the JSR instruciton.

Now, trace the Worm program.

```
= T  7
CC    B    A    X     PC    S     RAM  ROM
CØ    ØØ   89   ØØ3C  ØØ22  81FD  Ø    Ø
CC    B    A    X     PC    S     RAM  ROM
C4    ØØ   ØØ   ØØ3C  ØØ24  81FD  Ø    Ø
CC    B    A    X     PC    S     RAM  ROM
CB    ØØ   ØØ   ØØ3C  ØØ26  81FD  Ø    Ø
CC    B    A    X     PC    S     RAM  ROM
CB    ØØ   ØØ   ØØ3C  ØØ28  81FD  Ø    Ø
```

CC	B	A	X	PC	S	RAM	ROM
C1	ØØ	55	ØØ3C	ØØ2A	81FD	Ø	Ø

CC	B	A	X	PC	S	RAM	ROM
C1	ØØ	55	ØØ3C	ØØ2C	81FD	Ø	Ø

CC	B	A	X	PC	S	RAM	ROM
C4	ØØ	55	ØØ3C	ØØ2E	81FD	Ø	Ø

The Worm tests for the top of memory in the third step by comparing the most significant byte of NEXT with the top page of memory (80 $_{HEX}$). If the top has not been reached a check is made in steps 5 through 7 for enough memory in which to put the next occurrance of the Worm program.

= N

CC	B	A	X	PC	S	RAM	ROM
C4	ØØ	55	ØØ3C	ØØ3Ø	81FD	Ø	Ø

= N

CC	B	A	X	PC	S	RAM	ROM
C4	ØØ	55	ØØ3C	81D5	81FF	Ø	Ø

If there is enough RAM, a normal return from subroutine to the main program occurs.

= N

CC	B	A	X	PC	S	RAM	ROM
C4	ØØ	55	ØØ3C	8193	81FF	Ø	Ø

Loop begins again. If there is not enough RAM, then the ARN routine is executed which prints a "G" on the R2M's screen, checks for an abort

key (any key), and restarts the main program from INIT. To observe a complete run of the Main program and the Worm program, set a breakpoint at INIT and execute from DISPLA. When the breakpoint is encountered, perform a CNTL E to clear the screen and then execute the C command. An incrementing counter should appear in the upper left hand corner of the CRT. This counter will increment up to the top of user memory and stop. In a system which has a MEK68MM32 board the following display results.

```
= M 7FCR
= C G
CC    B    A    X     PC    RAM   ROM
CØ    1Ø   Ø8   8ØØØ  8189  Ø     Ø
=
```

Notice that the index register has reached the top of the memory map and the program printed a "G" and jumped to INTI (8189 $_{HEX}$). Remove the breakpoint and allow the program to run.

```
= D
= G  818Ø
```

A display similar to the following should appear:

```
1234  GG
```

Where the 1234 represents the counter and the G's are the number of successful passes. The program can be aborted by depressing the space bar. You now have a usable RAM test program for checking system memory.

3.3 User Callable Subroutines

There are a number of subroutines in D3BUG2 that could be useful to

the user. A list of these with addresses and a brief description are given at the end of this chapter. This section groups these subroutines and discusses how they relate to each other.

There is a group of routines (see Table 3.1) which can be used to input and output packed and unpacked hex characters. They interact to produce complete addresses and data bytes from inputted ASCII characters which can be used by the calling routine.

TABLE 3.1. SUBROUTINES (PACKED/UNPACKED HEX CHARACTERS)

Subroutine	Address	Subroutine	Address
ADDRES	$F1B5	OUT4HS	$F14Ø
BEGADD	$F1A6	HXBUFØ	$F13D
ENDADD	$F1B7	HEXBF2	$F144
INBYTE	$F1D2	OUT2H	$F14D
INADDR	$F1D9	OUT2HA	$F14F
OUTA	$F12A	UNPACK	$F122
OUTB	$F12D	PACK2B	$F22F
OUTAB	$F139	PACK4B	$F233
OUT2HS	$F147		

Another group of routines handles the input and output of ASCII characters and performs ASCII to hex and hex to ASCII conversion (see Table 3.2). These routines perform the actual display of characters on the MEK68R2M or the RS-232 Terminal.

3.3 User Callable Subroutine (cont'd)

TABLE 3.2. SUBROUTINES (ASCII TO HEX & HEX TO ASCII)

Subroutine	Address
INCHR	$FØE2
OUTCHR	$F156
MESAGE	$F198
ASCHEX	$FØFE
HEXASC	$F114

The last group of routines affect the RAM and ROM pages. With these routines up to 8 pages of RAM and 8 pages of ROM can be accessed, making it possible for programs and related data to be larger than the memory address space of the MC6802 (64K bytes). These routines (see Table 3.3) allow the user to change the page register by suppling a new page number between Ø and 7 in the A-accumulator or by incrementing or decrementing the page number. The user should save all registers before calling these if they are not to be changed.

TABLE 3.3. SUBROUTINES (RAM & ROM PAGES)

Subroutine	Address
CHGRAM	$F592
CNGROM	$F586
INCRAM	$F5B1
INCROM	$F5C2
DECRAM	$F5BA
DECROM	$F5CB

3.3 User Callable Subroutines (cont'd)

Following is an example program which calls a subroutine in another page and returns to the calling routine.

```
PSHB
PSHA
LDAA        PAGERG          Old page number ($8Ø83)
STAA        SAVEPG          Save it
LDAA        #NEWPG          New page number
JSR         CHGRAM          Change RAM page
PULA
PULB                        Restore registers
JSR         ROUTINE         Call subroutine
PSHA
LDAA        SAVEPG          Get old page number
STAA        PAGERG          Change back
PULA
RTS                         Return from subroutine
```

Upon entry all registers are saved in order for the subroutine to pass parameter without inhibitaions. A similar program can be written to call a subroutine in ROM by replacing CNGRAM with CNGROM.

If the subroutine is in the next page then INCRAM can be used instead:

```
Call        PSHB
            PSHA                    Save register
```

```
        JSR     INCRAM      Change to next page
        PULA
        PULB                Restore register
        JSR     ROUTINE
        PSHB
        PSHA
        JSR     DECRAM      Change back to old page
        PULA
        PULB
        RTS                 Return to program
```

This routine is somewhat shorter than the first program and uses only the stack for temporary storage. A routine in ROM can be called in a similar manner by using INCROM and DECROM.

The following is a description of the user callable subroutines with information about the way each affect the MC6802s registers (changes or does not change them), different memory locations, and the stack (the stack depth). Also a description of what each does is given.

Glossary:

A	is the A-accumulator.
B	is the B-accumulator.
CCR	is the Condition Code Register.
X	is the Index Register.
$	indicates this is a hex number.

3.3.1 ADDRES $F1B5

Subroutine: ADDRES.

Address: $F1B5.

Subroutines called: BEGADD, MESAGE, INADD1, OUTS.

Registers affected: A, B, X, CCR.

Memory affected: BEGA ($813A), ENDA ($813C), MEK68R2M display memory.

Stack: 18 bytes.

Description: Returns two addresses stored in BEGA and ENDA, respectively. Address will display prompt and wait for a beginning and then for an ending address to be entered.

3.3.2 ADDXA $F94D

Subroutine: ADDXA.

Address: $F94D.

Subroutines called: None.

Registers affected: A, X, CCR.

Memory affected: XCALC ($8113).

Stack: 2 bytes.

Description: Adds the A-accumulator to the Index register.

3.3.3 ASCHEX $FØFE

Subroutine: ASCHEX.

Address: $FØFE.

Subroutines called: None.

Registers affected: A, CCR.

Memory affected: None.

Stack: 2 bytes.

3.3.3 ASCHEX $FØFE (cont'd)

Description: Converts an ASCII character in the A-accumulator to its hex equivalent. The carry flag is set if the character does not have a hex equivalent. The contents of the A-accumulator is destroyed. The carry flag is cleared if the character has a hex equivalent and the A-accumulator holds the converted character.

3.3.4 BEGADD $F1A6

Subroutine: BEGADD.

Address: $F1A6.

Subroutines called: CRLF, MESAGE, INADD1.

Registers affected: A, B, X, CCR.

Memory affected: BEGA ($813A).

Stack: 16 bytes.

Description: Returns an address in BEGA. BEGADD displays a prompt and waits for a beginning address to be entered.

3.3.5 CLEAR $F699

Subroutine: CLEAR.

Address: $F699.

Subroutines called: None.

Registers affected: A, X, CCR.

Memory affected: MEK68R2M's memory.

Stack: 2 bytes.

Description: Writes an ASCII space ($2Ø) in all locations from $9ØØØ to $9FFF. This clears the R2M's display.

3.3.6 CLRSCR $F6A7

Subroutine: CLRSCR.

Address: $F6A7.

Subroutines called: CLEAR, CRTIZ.

Registers affected: A, B, X, CCR.

Memory affected: R2M's memory, FORMAT ($8115), CTEMP ($81ØD), CLINE ($816Ø), LSCREN ($815E), CURPOS ($8164), STARAD ($8162), LINEAD ($8166), LCOUNT ($815C), and CCOUNT ($815A).

Stack: Four bytes.

Description: Clears the MEK68R2M's display and re-initialize the MC 6845's display registers.

3.3.7 CNGRAM $F592

Subroutine: CNGRAM.

Address: $F592.

Subroutines called: None.

Registers affected: A, B, CCR.

Memory affected: PAGERG ($8Ø83).

Stack: Two bytes.

Description: Changes the RAM page to the page number in the A-accumulator. The page number is limited to Ø through 7.

3.3.8 CNGROM $F586

Subroutine: CNGROM.

Address: $F586.

3.3.8 CNGROM $F586 (cont'd)

Registers affected: A, B, CCR.

Memory affected: PAGERG ($8Ø83).

Stack: Two bytes.

Description: Changes the ROM page to the page number in the A-accumulator. The page number is limited to Ø through 7.

3.3.9 CRLFR2 $F627

Subroutine: CRLFR2.

Address: $F627.

Subroutine called. RTURN1.

Registers affected: A, B, CCR.

Memory affected: CTEMP ($81ØD), CURPOS ($8164), CCOUNT ($815A), CUTEMP ($81ØB), LINEAD ($8166), STARAD ($8162), CRTCAR ($8Ø42), and CRTCDR ($8Ø43).

Stack: 4 bytes.

Descripton: Moves the MEK68R2M's cursor to the next line. If the next line is off the screen CRLF2 will scroll-up one line which pushes the top line off the screen and positions the cursor at the beginning of the bottom line. This bottom line is cleared except for the cursor.

3.3.10 CRLFTR $F61E

Subroutine: CRLFTR.

Address: $F61E.

Subroutines called: OUTCHR.

3.3.10 CRLFTR $F61E (cont'd)

Registers affected: A, B, CCR.

Memory affected: ACIATS ($8ØØ8), ACIA Output Register.

Stack: 4 bytes.

Description: Outputs an ASCII line feed and an ASCII carriage return to the terminal ACIA. The flag TERM ($81ØA) should be set before execution of CRLFTR.

3.3.11 CRTIZ $F6A9

Subroutine: CRTIZ.

Address: $F6A9.

Subroutines called: UDATE1.

Registers affected: A, B, X, CCR.

Memory affected: CTEMP ($81ØD), a temporary register; CLINE ($816Ø), the # of characters per line; LSCREN ($815E), the # of lines on the screen; CURPOS ($8164), the cursor's position; STARAD ($8162), the starting address of the display; LINEAD ($8166), the starting address of the present line; LCOUNT ($815C), the # of lines written on the screen; CCOUNT ($815A), the # of characters written on the present line.

Stack: 4 bytes.

Description: Initializes the MEK68R2M display using the parameter table specified by the address stored in FORMAT ($8115). This table can be one of the six found in D3BUG2 or one supplied by the user (see Table 3.4).

3.3.11 CRTIZ $F6A9 (cont'd)

TABLE 3.4

Label	Address	Format
CRTAB1	$F7Ø8	U.S. 32 x 16
CRTAB2	$F71A	U.S. 64 x 16
CRTAB3	$F72C	U.S. 80 x 20
CRTAB4	$F73E	European 32 x 16
CRTAB5	$F75Ø	European 64 x 16
CRTAB6	$F762	Euorpean 80 x 20

3.3.12 DECRAM $F5BA

Subroutine: DECRAM.

Address: $F5BA.

Subroutines called: GETPG, CNGRAM.

Registers affected: A, B, CCR.

Memory affected: PAGERG ($8Ø83).

Stock: 4 bytes.

Description: Decrements the RAM page register by one. If the result would lie out side the page boundaries Ø to 7 then the Z flag is set.

3.3.13 DECROM $F5CB

Subroutine: DECROM.

Address: $F5CB.

Subroutines called: GETPG, CNGROM.

3.3.13 DECROM $F5CB (cont'd)

Memory affected: PAGERG ($8Ø83), the page register.

Stack: 4 bytes.

Description: Decrements the ROM Page register by one. Only pages between Ø and 7 are allowed. The Z flag is set if decrementing the page register would go out side this limit.

3.3.14 ENDADD $F1B7

Subroutine: ENDADD.

Address: $F1B7.

Subroutines called: MESAGE, INADD1, OUTS.

Registers affected: A, B, X, CCR.

Memory affected: ENDA ($813C), CTEMP ($81ØD), CUTEMP ($81ØB), CURPOS ($8164), CCOUNT ($815A), ROLFLG ($8134), CRFLAG ($8154), HEXBUF ($8118), and XTMP3 ($8111).

Stack: 16 bytes.

Description: Returns an address in ENDA. ENDADD displays a prompt and waits for the ending address to be entered.

3.3.15 HEXASC $F114

Subroutine: HEXASC.

Address: $F114.

Subroutines called: None.

Registers affected: A, CCR.

Memory affected: None.

3.3.15 HEXASC $F114 (cont'd)

Stack: 2 bytes.

Description: Converts an unpacked hex character in "A" to its ASCII equivalent. If "A" does not hold an unpacked character the carry flag is set.

3.3.16 INADDR $F1D9

Subroutine: INADDR.

Address: $F1D9.

Subroutines called: ASCHEX, OUTA, PACKIT, INCHR.

Registers affected: A, B, X, CCR.

Memory affected: CRFLAG, ROLFLG, HEXBUF, FLG24, KEY.

Stack: 14 bytes.

Description: Returns an address in both the index register and HEXBUF. If no address was entered ROLFLG ($8134) is cleared. If a carriage return was pressed CRFLAG ($8154) is set. The terminating key (i.e., line feed) is returned as a code in the B-accumulator (see Table 3.2).

3.3.17 INBYTE $F1D2

Subroutine: INBYTE.

Address: $F1D2.

Subroutines called: ASCHEX, OUTA, PACKIT, INCHR.

Registers affected: A, B, X, CCR.

Memory affected: XTMP3, FLG24, ROLFLG, CRFLAG, KEY, and HEXBUF +2.

Stack: 14 bytes.

3.3.17 INBYTE $F1D2 (cont'd)

Description: Returns a byte in both the A-accumulator and HEXBUF +2. The terminating key (i.e., line feed) is returned as a code in the B-accumulator (see Table 3.5). If no byte is entered, ROLFLG ($8134) is cleared. If a carriage return is depressed, CRFLAG ($8154) is set.

TABLE 3.5

Key	Code	Description
M	Ø	Back up
^	Ø	Back up
CR	1	Carriage return
LF	3	Line feed
O	4	Offset
SP	5	Space bar

3.3.18 INCHR $FØE2

Subroutine: INCHR.

Address: $FØE2.

Subroutines called: None.

Registers affected: A, CCR.

Memory affected: PIA1AC ($8Ø45), ACIATD ($8ØØ9).

Stack: 2 bytes.

Description: Inputs an ASCII character from the console keyboard.

3.3.18 INCHR $FØE2 (cont'd)

The type of console is determined by the flag TERM (see Table 3.6). The most significant bit of the input character is masked.

TABLE 3.6

Term	Console
Ø	MEK68R2M
1	RS-232 Terminal

3.3.19 INCRAM $F5B1

Subroutine: INCRAM.

Address: $F5B1.

Subroutines called: GETPG, CNGRAM.

Registers affected: A, B, CCR.

Memory affected: PAGERG ($8Ø83).

Stack: 4 bytes.

Description: Increments the RAM page register by one, if the result would exceed the page boundary Ø to 7 the Z flag is cleared.

3.3.20 INCROM $F5C2

Subroutine: INCROM.

Address: $F5C2.

Subroutines called: GETPG, CNGROM.

Registers affected: A, B, CCR.

3.3.20 INCROM $F5C2 (cont'd)

Memory affected: PAGERG.

Stack: 4 bytes.

Description: Increments the ROM page by one. The Z-flag is cleared if the page boundary is encountered.

3.3.21 LFEED $F629

Subroutine: LFEED.

Address: $F629.

Subroutines called: None.

Registers affected: A, B, CCR.

Memory affected: CTEMP ($81ØD), CURPOS ($8164), LINEAD ($8166), CUTEMP ($81ØB, $81ØC), LCOUNT ($815C), STARAD ($8162, $8163), CRTCAR ($8Ø42), and CRTCDR ($8Ø43).

Stack: 2 bytes.

Description: Advance the cursor to the next line but does not left justify it (i.e., no carriage return).

3.3.22 MESAGE $F198

Subroutine: MESAGE.

Address: $F198.

Subroutines called: OUTCHR.

Registers affected: B, X, CCR.

Memory affected: CTEMP ($81ØD), CURPOS ($8164), CUTEMP ($81ØB), CCOUNT ($815A), CRTCAR ($8Ø42), and CRTCDR ($8Ø43).

Stack: 9 bytes.

3.3.22 MESAGE $F198 (cont'd)

Description: Displays a string of ASCII characters which is terminated with an ASCII EOT ($Ø4). The Index register points to the beginning of the string upon entry and points to the end of the string upon exit.

3.3.23 OUT2H $F14D

Subroutine: OUT2H.

Address: $F14D.

Subroutines called: UNPACK, OUTA, OUTB.

Registers affected: A, B, X, CCR.

Memory affected: CTEMP ($81ØD), CURPOS ($8164), CUTEMP ($81ØB), CCOUNT ($815A), LINEAD ($8166), STARAD ($8162), CRTCAR ($8Ø42), and CRTCDR ($8Ø43).

Stack: 10 bytes.

Description: Outputs the byte pointed to by the Index register with no following space. The index register is incremented by one.

3.3.24 OUT2HA $F14F

Subroutine: OUT2HA.

Address: $F14F.

Subroutines called: UNPACK, OUTA, OUTB.

Registers affected: A, B, X, CCR.

Memory affected: CTEMP ($81ØD), CURPOS ($8164), CUTEMP ($81ØB), CCOUNT ($815A), LINEAD ($8166), STARAD ($8162), CRTCAR ($8Ø42), and CRTCDR ($8Ø43).

Stack: 10 bytes

3.3.24 OUT2HA $F14F (cont'd)

Description: Displays the byte (2 hex characters) in the A-accumulator with no following space. The Index register is incremented by one.

3.3.25 OUT2HS $F147

Subroutine: OUT2HS.

Address: $F147.

Subroutines called: OUT2H, OUTCHR.

Registers affected: A, B, X, CCR.

Memory affected: See OUT2HA and OUTCHR.

Stack: 12 bytes.

Description: Displays the byte (two hex characters) pointed to by the Index register with a following space. The A-accumulator returns with $2Ø (ASCII space) and the Index register is incremented by one.

3.3.26 OUT4HS $F14Ø

Subroutine: OUT4HS.

Address: $F14Ø.

Subroutines called: OUT2H, OUT2HS.

Registers affected: A, B, X, CCR.

Memory affected: See OUT2H, OUT2HS.

Stack: 12 bytes.

Descritpion: Displays two bytes (4 hex characters) pointed to by the Index Register with a following space. The Index registers is incremented by 2.

3.3.27 OUTA $F12A

Subroutine: OUTA.

Address: $F12A.

Subroutines called: HEXASC, OUTCHR.

Registers affected: CCR.

Memory affected: see OUTCHR.

Stack: 8 bytes.

Description: Displays the unpacked hex character in the A-accumulator. The carry flag is set if the A-accumulator does not hold an unpacked hex character.

3.3.28 OUTAB $F139

Subroutine: OUTAB.

Address: $F139.

Subroutines called: OUTA, OUTB.

Registers affected: CCR.

Memory affected: See OUTA, OUTB.

Stack: 10 bytes.

Description: Displays the unpacked hex characters in the A-accumulator and the B-accumulator.

3.3.29 OUTB $F12D

Subroutine: OUTB.

Address: $F12D.

Subroutines called: HEXASC, OUTCHR.

Registers affected: CCR.

3.3.29 OUTB $F12D (cont'd)

Memory affected: See HEXASC, OUTCHR.

Stack: 8 bytes.

Descritpion: Displays the unpacked hex character in the B-accumulator. The carry flag is set if the B-accumulator does not contain an unpacked character.

3.3.30 OUTCHR $F156

Subroutine: OUTCHR.

Address: $F156.

Subroutines called: UPDATE, CRLF.

Registers affected: A, B, CCR.

Memory affected: CTEMP ($81ØD), ACIATD ($8ØØ9), CURPOS ($8164), CUTEMP ($81ØB), CCOUNT ($815A), see CRLF, UPDATE.

Stack: 6 bytes.

Description: Displays the ASCII character in the A-accumulator. The console display is determined by the flag term.

3.3.31 OUTS $F149

Subroutine: OUTS.

Address: $F149.

Subroutines called: OUTCHR.

Registers affected: A, B, CCR.

Memory affected: See OUTCHR.

Stack: 6 bytes.

Description: Display a space on the console display.

3.3.32 PACK2B $F22F

Subroutine: PACK2B.

Address: $F22F.

Subroutines called: ROLENT.

Registers affected: A, B, CCR.

Memory affected: FLG24 ($8135), ROLFLG ($8134), see ROLENT, HEXBUF +2 ($811A).

Stack: 5 bytes.

Description: Packs the unpacked hex character in the A-accumulator into HEXBUF +2 ($811A).

3.3.33 PACK4B $F233

Subroutine: PACK4B.

Address: $F233.

Subroutines called: ROLENT.

Registers affected: A, B, CCR.

Memory affected: FLG24 ($8135), ROLFLG ($8134), HEXBUF ($8118, $8119), see ROLENT.

Stack: 5 bytes.

Description: Packs the unpacked hex character in the A-accumulator into HEXBUF; HEXBUF +1.

3.3.34 RMBKPT $FDFD

Subroutine: RMBKPT.

Address: $FDFD.

Subroutine called: None.

3.3.34 RMBKPT $FDFD (cont'd)

Registers affected: A, B, X, CCR.

Memory affected: POINTR ($813E), any place where a SWI has been set by the GO command.

Stack: 2 bytes.

Description: Removes SWI breakpoints from a program using the breakpoint table ($8143 to $8163) for refrence addresses. RMBKPT checks for a SWI op-code at each address on the table, if one is found the original op-code which was saved earlier on the table is re-inserted. If a SWI op-code is not found the original op-code is not re-inserted.

3.3.35 ROLENT $FB44

Subroutine: ROLENT.

Address: $FB44.

Subroutines called: None.

Registers affected: A, B, CCR.

Memory affected: ROLFLG ($8134), HEXBUF ($8118, $8119), and HEXBUF +2 ($811A).

Stack: 2 bytes.

Description: Assembles unpacked hex characters into one or two byte packed hex numbers. The flag FLG24 determines whether to pack into one byte or two (see Table 3.7). The result of the one byte operation is stored in HEXBUF +2, while the two byte operation stores its result in HEXBUF ($8118)

3.3.35 ROLENT $FB44 (cont'd)

and HEXBUF +1 ($8119). Another flag, ROLFLG, is used to add leading zeros before the first hex character (refer to Table 3.7).

TABLE 3.7. FLAG OPERATION

Flag	# Byte	Result
FLG24	Ø	4 hex characters are to be packed.
	1	2 hex characters are to be packed.
ROLFLG	Ø	Add leading zeros.
	1	Do not add leading zeros.

3.3.36 RTURN1 $F6ØB

Subroutine: RTURN1.

Address: $F6ØB.

Subroutines called: UPDATE.

Registers affected: A, B, CCR.

Memory affected: CTEMP ($81ØD), LINEAD ($8166), CURPOS ($8164), CCOUNT ($8154), see UPDATE.

Stack: 2 bytes.

Description: Returns the MEK68R2M's cursors to the beginning of the current line.

3.3.37 SCROLU $F672

Subroutine: SCROLU.

3.3.37 SCROLU $F672 (cont'd)

Address: $F672.

Subroutines called: UPDATE.

Registers affected: A, B, X, CCR.

Memory affected: LCOUNT ($815C), STARAD ($8162, $8163), see UPDATE.

Stack: 2 bytes.

Description: Moves the display up one line by changing the starting address of the display memory of MEK68R2M to the starting address of the second line of the display. In other words the display window is moved down one line to expose a previously undisplayed line at the bottom of the screen.

3.3.38 SFTA4R $F125

Subroutine: SFTA4R.

Address: $F125.

Subroutines called: None.

Registers affected: A, CCR.

Memory affected: None.

Stack: 2 bytes.

Description: Shifts the A-accumulator four bits to the right.

3.3.39 TERMIN $FØ11

Subroutine: TERMIN.

Address: $FØ11.

Subroutines called: None.

Registers affected: A, CCR.

3.3.39 TERMIN $FØ11 (cont'd)

Memory affected: IOPIAD ($8ØØ4), IOPIAC ($8ØØ5), and TERM ($81ØA).

Stack: 2 bytes.

Description: Tests for a RS-232 terminal attached to the MEK68IO board. Bit 1 of a PIA register ($8ØØ4) determines whether the terminal is there or not (see Table 3.8). The flag TERM is set if bit 1 is high and cleared if bit 1 is low.

TABLE 3.8. FLAG OPERATION

Bit 1	Term	Result
Ø	Ø	No Terminal
1	1	Terminal is main console

3.3.40 UNPACK $F122

Subroutine: UNPACK.

Address: $F122.

Subroutines called: None.

Registers affected: A, B, CCR.

Memory affected: None.

Stack: 2 bytes.

Description: Unpacks two hex characters in the A-accumulator into the A-accumulator (Most Significant nybble) and the B-accumulator (Least Significant nybble).

3.3.41 UPDATE $F682

Subroutine UPDATE.

3.3.41 UPDATE $F682 (cont'd)

Address: $F682.

Subroutines called: None.

Registers affected: A, B, X, CCR.

Memory affected: CRTCAR ($8Ø42), CRTCDR ($8Ø43).

Stack: 2 bytes.

Description: Updates the display registers in the MC6845 (see MEK68R2M data sheet) using the display parameters stored at STARAD ($8162, $8163) and CURPOS ($8164, $8165). STARAD holds the starting address of the screen display. CURPOS is the present location of the cursor. By changing STARAD different parts of the display memory can be displayed on the CRT. CURPOS allows the user to move the cursor through the display. By loading these two and calling update, the cursor can be easily changed without affecting the other display parameters of the MC6845.

3.4 Jump Table

D3BUG2 contains a jump table to which other programs may jump to execute certain routines in the ROM. There are five jumps on the table and are listed below.

TABLE 3.9. D3BUG2 JUMP TABLE

Address	Jump	Jump Address
$FØØ2	JMP ENT1	$FØ37
$FØØ5	JMP CONTRL	$FØA5
$FØØ8	JMP EXGET	$F37B
$FØØB	JMP OUTCHR	$F156
$FØØE	JMP EXKEY	$F363

3.4 Jump Table (cont'd)

ENT1 is the restart location of D3BUG2. The display is re-initialized and control is given to D3BUG2.

CONTROL is the beginning of D3BUG2's command routine. The display is not re-initialized and control is given to D3BUG2.

EXGET allows the user to enter data from the Keyboard. One byte (2 hex characters) can be entered if the INIT2 ($8137) is cleared. The byte is returned in the A-accumulator. Two bytes (4 hex characters) can be entered into the Index register if the flag INT2 is set.

OUTCHR outputs an ASCII character to the system console. The flag TERM ($81ØA) selects the type of console in use. If cleared the MEK68R2M is used for the console. IF set the RS-232 serial port on the MEK68I0 ($8ØØ8, $8ØØ9) is used for the console (see the MEK68I0 manual for interfacing information).

EXKEY provides a means of detecting the depressing of a key on the console's keyboard. To use EXKEY for this function the flag ETRFG ($8158) should be cleared, otherwise control is given to D3BUG2 without testing for a key. Depressing a key will give control to D3BUG2. If no key has been depressed, then EXKEY will return to the calling program.

3.5 Interrupts

All the interrupts of the MC6802 are available to the user. Many of the interrupts are also used by the monitor, but flags are set when this is the case. This enables the monitor to detect user interrupts and trans-

3.5 Interrupts

fers control to user interrupt handlers. Six memory locations are set aside for the user's interrupt routine addresses. They are UNMIV (NMI VECTOR, $81Ø4 and $81Ø5), UIRQV (IRQ VECTOR, $81Ø6, $81Ø7), and USWIV (SWI VECTOR, $81Ø8 and $81Ø9). Each of the three types of interrupts are discussed separately.

3.5.1 Non-Maskable Interrupt (NMI)

The Non-Maskable Interrupt is the highest priority interrupt in the MC6802. As its name states no other interrupts (IRQ, SWI) can mask it, which means it cannot be overridden except by external hardware. When this interrupt occurs, a vector is fetched which points to the beginning of the interrupt handler program. The NMI interrupt is used by D3BUG to sense a key depressed on the MEK6802D3's keypad or to interrupt the processor in order that an instruction can be traced. If neither of these have caused the NMI interrupt, than D3BUG2 assumes it was user induced and fetchs the address stored at UNMIV ($81Ø4, $81Ø5) and jumps to the user's interrupt handler. The user must put the interrupt handler's address at UNMIV before attempting to use the NMI interrupt.

3.5.2 IRQ Interrupt

The IRQ Interrupt is a maskable interrupt. By setting the interrupt mask bit in the MC6802's condition code register, the IRQ interrupt can be enabled or disabled under software control. Upon reset, D3BUG2 disables the IRQ interrupt, and the user must enable it when it is needed. The address of interrupt handler program must be placed at UIRQV ($81Ø6,

3.5.2 IRQ Interrupt (cont'd)

$81Ø7) before using IRQ interrupt.

3.5.3 SWI Interrupt

The Software Interrupt is actually an MC6802 instruction that is placed in a program to cause an induced interrupt much like the NMI and IRQ interrupts. When a SWI instruction ($3F) is encountered in a program the MC 6802's registers are pushed onto the stack and the interrupt handler vector is fetched causing control to be given to to the handler. D3BUG2 uses the SWI interrupt to perform its breakpoint function, but has the ability to distinguish between its own SWI's and the user's. The user can place the starting address of his SWI handler routine at USWIV ($81Ø8, $81Ø9) and set his own SWI's in his program independently of D3BUG2's.

The user has full use of the MC6802's interrupts by placing interrupt handler addresses at the proper memory locations and pulling the correct inputs to MC6802 low or setting SWI instructions in his programs. Programs which depend on these interrupts can be de-bugged with ease using D3BUG2.

3.6 Interfacing a RS-232 Terminal

D3BUG2 is designed to output to either a MEK68R2M board or the RS-232 serial port ($8ØØ8, $8ØØ9) on the MEK68IO board. The flag TERM ($81ØA) is used to determine which type of display is in use. D3BUG2 looks at jumpers on both the MEK68IO and the MEK68R2M to determine which way to set TERM. If both a RS-232 terminal and a MEK68R2M board are in the system, the RS-232 terminal is used for the display.

3.7 D3 Keypad Operation With D3BUG2

D3BUG2 allows the use of the D3BUG keypad monitor stored in the MC6846. D3BUG is invoked simply by pressing the EX key on the MEK6802D3 keypad. The LED display lights up with the = prompt and all the commands of D3BUG can be used. D3BUG2 can be re-entered by depressing a key on the console's ASCII keyboard or by depressing the RS key (Reset) on the MEK6802D3's keypad.

3.8 User Defined Functions

D3BUG2 allows the user to define up to three functions, which may be invoked by depressing the shift key and either the 1, 2, or 3 key. The address of these user's functions are stored in the last three optional registers and may be changed by using the Register display command. The user may end his function with a RTS instruction, which returns to the monitor. In this way the user may implement his own commands to the monitor.

Example: Changing the user defined functions.

Displayed Registers.

.
.
.

ROMPG = Ø

ØØØØ	Reg	=	ØØØØ		
1234	Reg	=	ØØØ2	Ø	Change user function.
1234	Reg	=	ØØØ2	1Ø	Address 1 to location zero.

3.9 Optional Display Registers

D3BUG2 allows the user to display four additional registers in addition

3.9 Optional Display Registers (cont'd)

to those of the MC6802 if the 16 x 64 or the 20 x 80 display format is selected on the MEK68R2M board or a RS-232 Terminal is attached to the I/O board. The optional registers may be displayed and changed through the use of the register display command in any display format.

The user enters the address of the memory location to be displayed when the following display appears on the CRT:

ØØØØ Reg = ØØØF 4343

The above display can be interpreted in the following manner: The first hex number ØØØØ is actually the contents of the 16 bit memory location ØØØF. The hex number 4343 is the new 16 bit memory location entered by the user and replaces the ØØØF. All four optional registers may be displayed and changed in this manner.

Note: See register display command for more information regarding the optional registers.

3.10 Program Flow

The D3BUG2 monitor program interfaces the MEK68R2M board with the debug routines in D3BUG. D3BUG2 also expands the capabilities of D3BUG with added features like multiple instruction trace, offset load of cassette programs and display optional registers during tracing. D3BUG2 begins operation when power is applied and a MEK68R2M board or a RS-232 terminal is attached to a MEK68IO board in the system. If neither is in the system, depress the EX key on the MEK6802D3 and the = prompt will appear on the LED display.

3.10 Program Flow (cont'd)

The flow charts of the commands and subroutines which make up D3BUG2 can be found in Appendix 3.

APPENDIX 1

PIN DESCRIPTION

APPENDIX 1. (MEK68R2/MEK68R2M) SIGNAL LIST PIN

Pin	Signal	Pin	Signal
1	GND	40	A10
4	+5V	41	A11
6	OPEN	42	A12
9	VMA	43	A13
11	R/$\overline{W}$	44	A14
12	GND	45	A15
13	$\overline{RESET}$	46	GND
15	$\overline{IRQ}$	47	GND
19	GND	48	DØ
27	I/O1	49	D1
28*	I/O2	50	D2
29*	$\overline{ACT}$	51	D3
30	AØ	52	D4
31	A1	53	D5
32	A2	54	D6
33	A3	55	D7
34	A4	56	+12V
35	A5	57	GND
36	A6	58	-12V
37	A7	59	+5V
38	A8	60	GND
39	A9		

* Pins not used on MEK68R2M

APPENDIX 1. (cont'd)

U40 PIN ASSIGNMENTS - TTL VIDEO AND LIGHT PEN CONNECTOR

Pin	Function	Pin	Function
1	CA2 I/O	9	N/C
2	L.P. STROBE	10	N/C
3	L.P. CONTROL	11	N/C
4	N/C	12	N/C
5	N/C	13	N/C
6	N/C	14	V. SYNC
7	N/C	15	H. SYNC
8	GND	16	TTL VIDEO

APPENDIX 2

SOFTWARE LISTING, D3BUG2 1.0

```
00001                           NAM     D3BUG2
00002                           TTL     - MOTOROLA INC., "MAKING IT HAPPEN IN MEMOR
00003                           OPT     CREF,O,LLEN=80


00005                   ************************************************************
00006                   *       REV 1.0
00007                   *       COPYRIGHT (C) 1978 BY MOTOROLA INC.
00008                   *
00009                   *       MEMORY SYSTEMS  "MAKING IT HAPPEN"
00010                   *       AUSTIN, TEXAS
00011                   *       MICROCOMPUTER CAPITAL OF THE WORLD!
00012                   *                               5/16/79
00013                   *   D3BUG2 COMMAND SUMMARY
00014                   *
00015                   *       L   LOAD FROM TAPE
00016                   *       M   MEMORY CHANGE; O OFFSET SUBCOMMAND
00017                   *       O   OFFSET LOAD
00018                   *       P   PUNCH TO TAPE
00019                   *       E   EXAMINE MEMORY BLOCK
00020                   *       R   DISPLAY CONTENTS OF TARGET STACK
00021                   *   CC    B    A    X    P    S RAM ROM REG1 REG2 REG3 REG4
00022                   *       B   PRINT OUT ALL BREAKPOINTS
00023                   *       C   CONTINUE EXECUTION FROM CURRENT LOCATION
00024                   *       N   NEXT INSTRUCTION
00025                   *       T   TRACE 'N' INSTRUCTIONS
00026                   *       G   GO TO LOCATION 'N'
00027                   *       D   DELETE ALL BREAKPOINTS
00028                   *       U   RESET BREAKPOINT WITH ADDRESS 'N'
00029                   *       V   VERIFY TAPE
00030                   *       S   SET A BREAKPOINT WITH ADDRESS 'N'
00031                   *       SHIFT '!'  USER DEFINED ENTRY,OPTION REG 2
00032                   *       SHIFT '"'  USER DEFINED ENTRY,OPTION REG 3
00033                   *       SHIFT '#'  USER DEFINED ENTRY,OPTION REG 4
00034                   *
00035                   *   D3BUG2  CONTROL FUNCTIONS
00036                   *
00037                   * CTL E   CLEAR SCREEN - HOME CURSOR
00038                   *
```

```
00040                    *  USER CALLABLE SUBROUTINES IN D3BUG2
00041                    *
00042                    * ADDRES $F1B5 RETURNS TWO ADDRESSES IN BEGA AND ENDA
00043                    * ADDXA  $F94D ADDS THE A-ACC. TO THE INDEX
00044                    * ASCHEX $F0FE CONVERTS ASCII TO HEX
00045                    * INADDR $F1D9 BUILD 16 BIT ADDR IN X REG
00046                    * INBYTE $F1D2 BUILDS 8 BIT HEX VALUE IN A REG
00047                    * CLEAR  $F699 STORE ASCII SPACE IN DISPLAY RAM
00048                    * CLRSCR $F6A7 CLEARS DISPLAY RAM REINITS CRTC
00049                    * CNGRAM $F592 CHANGES THE RAM PAGE
00050                    * CNGROM $F586 CHANGES THE ROM PAGE
00051                    * CRLFR2 $F627 CARRIAGE RETURN LINE FEED ON THE R2M
00052                    * CRLFTR $F61E CARRIAGE RETURN LINE FEED ON THE TERMINAL
00053                    * CRTIZ  $F6A9 INIT CRTC WITH FORMAT= ADDR OF CRT TABLE
00054                    * BEGADD $F1A6 BUILDS BEGIN. 16 BIT ADDR IN BEGA
00055                    * DECRAM $F5BA DECREMENT THE RAM PAGE BY ONE
00056                    * DECROM $F5CB DECREMENT THE ROM PAGE BY ONE
00057                    * ENDADD $F1B7 BUILDS BEGIN-END 16 BIT ADDR
00058                    * HEXASC $F114 CONVERTS HEX TO ASCII
00059                    * INCHR  $F0E2 INPUTS ONE CHAR FROM KEYBD
00060                    * INCRAM $F5B1 INCREMENTS THE RAM PAGE BY ONE
00061                    * INCROM $F5C2 INCREMENTS THE ROM PAGE BY ONE
00062                    * LFEED  $F629 LINE FEED ON THE R2M
00063                    * OUTS   $F149 OUTPUT SPACE
00064                    * OUT2H  $F14D OUTPUT 2 HEX CHAR POINTED BY X
00065                    * OUT2HA $F14F OUTPUT 2 HEX CHAR IN A REG
00066                    * OUT2HS $F147 OUTPUT 2 HEX CHAR + SPACE
00067                    * OUT4HS $F140 OUTPUT 4 HEX CHAR + SPACE
00068                    * OUTA   $F12A DISPLAYS THE UNPACKED HEX CHAR. IN A
00069                    * OUTAB  $F139 DISPLAYS THE UNPACKED HEX CHAR IN A AND B
00070                    * OUTB   $F12D DISPLAYS THE UNPACKED HEX CHAR IN B
00071                    * OUTCHR $F156 DISPLAYS THE ASCII CHAR. IN A
00072                    * MESAGE $F198 OUTPUT DATA POINTED BY X
00073                    * PACK2B $F22F PACKS THE HEX CHAR. IN A INTO HEXBUF+2
00074                    * PACK4B $F233 PACKS THE HEX CHAR. IN A INTO HEXBUF
00075                    * RMBKPT $FDFD REMOVES ANY SET BREAKPOINTS
00076                    * RTURN1 $F60B CARRIAGE RETURN ON THE R2M
00077                    * SCROLU $F672 SCROLL ONE LINE OF DISPLAY SCREEN
00078                    * SFTA4R $F125 SHIFTS THE A-ACC. 4 BITS RIGHT
00079                    * TERMIN $F011 DETECTS AN RS232 TERMINAL PORT
00080                    * UNPACK $F122 UNPACK A INTO A(MSN) AND B(LSN)
00081                    * UPDATE $F682 USED TO UPDATE 6845 CRTC
00082                    *
00083                    * I/O EQUATES
00084                    *
00085  8004  A IOPIAD EQU    $8004    THE PIA ON THE I/O BOARD
00086  8005  A IOPIAC EQU    $8005    IT'S COMMAND REG.
00087  8008  A ACIATS EQU    $8008    TERMINAL RS232 PORT
00088  8009  A ACIATD EQU    $8009
00089  800A  A ACIAS  EQU    $800A    K.C. CASSETTE
00090  800B  A ACIAD  EQU    $800B
00091  8042  A CRTCAR EQU    $8042    CRTC ADDR REG
00092  8043  A CRTCDR EQU    $8043    CRTC DATA REG
00093  8044  A PIA1AD EQU    $8044    R2 KEYBOARD PORT
00094  8045  A PIA1AC EQU    $8045
00095  8046  A PIA1BD EQU    $8046    R2 OPTIONS
00096  8047  A PIA1BC EQU    $8047
```

```
00098                    *
00099                    * SYSTEM EQUATES FOR D3BUG2
00100                    *
00101           F94D   A ADDXA   EQU     $F94D     ADD A TO X
00102           813A   A BEGA    EQU     $813A     BEGINNING ADDR. STORAGE
00103           813C   A ENDA    EQU     $813C     END ADDRESS STORAGE
00104A 8168              ENDOPT  ORG     $8168
00105A 8168     0008   A BEGOPT RMB     8         OPTIONAL REGS. ADDR. TABLE
00106           8170   A ENDOPT EQU     *
00107           815A   A CCOUNT EQU     $815A     CHARACTER REGISTER
00108           815C   A LCOUNT EQU     $815C     LINE REGISTER
00109           815E   A LSCREN EQU     $815E     LINES PER SCREEN-1
00110           8160   A CLINE  EQU     $8160     NUM CHAR/LINE
00111           8162   A STARAD EQU     $8162     STARTING ADDR OF DISPLAY RAM
00112           8164   A CURPOS EQU     $8164     CURRENT ADDR OF CURSOR
00113           8166   A LINEAD EQU     $8166     LINE  ADDRESS
00114           810A   A TERM   EQU     $810A     RS232 TERMINAL FLAG
00115           810B   A CUTEMP EQU     $810B     XTMP2
00116           810D   A CTEMP  EQU     $810D
00117           810F   A STEMP  EQU     $810F
00118           8111   A TEMPX  EQU     $8111     XTMP3
00119           8111   A XTMP3  EQU     TEMPX
00120           8154   A CRFLAG EQU     $8154     CARRIAGE RETURN FLAG
00121           8156   A DISFLG EQU     $8156     CONTROL FLAG FOR REGISTER DISPLAY
00122           F8AF   A ENNMI  EQU     $F8AF     ENABLES THE FRONT PANEL KEYBOARD
00123           8130   A EXFLG  EQU     $8130     EXTERNAL ROM PRESENCE FLAG
00124           8135   A FLG24  EQU     $8135
00125           8115   A FORMAT EQU     $8115
00126           8118   A HEXBUF EQU     $8118
00127           8132   A INIT   EQU     $8132
00128           8137   A INIT2  EQU     $8137
00129           8117   A KEY    EQU     $8117
00130           9000   A MEMSTR EQU     $9000     BEGIN ADDR DISPLAY MEMORY
00131           A000   A MEMEND EQU     $A000     END ADDR+1  DISPLAY MEMORY
00132           810F   A NMBTRC EQU     STEMP     NUMBER OF TRACES TO EXECUTE
00133           8083   A PAGERG EQU     $8083     PAGE REGISTER
00134           8134   A ROLFLG EQU     $8134     FLAG TO CONTROL ROLENT
00135           FB44   A ROLENT EQU     $FB44     HEX PACKING ROUTINE IN D3BUG
00136           817F   A SP     EQU     $817F     TOP OF MONITER STACK
00137                    *
00138                    * USER'S REGISTERS STORAGE LOCATIONS
00139                    *
00140           8124   A UCC    EQU     $8124
00141           8125   A UB     EQU     $8125
00142           8126   A UA     EQU     $8126
00143           8127   A UX     EQU     $8127
00144           8129   A UPC    EQU     $8129
00145           812B   A USP    EQU     $812B
00146           8104   A UNMIV  EQU     $8104     USER'S NMI VECTOR
00147           8106   A UIRQV  EQU     $8106     USER'S IRQ VECTOR
00148           8108   A USWIV  EQU     $8108     USER'S SWI VECTOR
```

```
00150                          *
00151                          * INIT CRT, SET SCREEN DIMEN, CLEAR SCREEN
00152                          *
00153A F000                            ORG     $F000
00154A F000    55      A               FCB     $55         ROM FLAG
00155A F001    AA      A               FCB     $AA
00156A F002 7E F037    A               JMP     ENT1        START OF D3BUG2
00157A F005 7E F0A5    A               JMP     CONTRL      REENTRY POINT OF D3BUG2
00158A F008 7E F37B    A               JMP     EXGET       GET EITHER AN ADDR. OR A BYTE
00159A F00B 7E F156    A               JMP     OUTCHR      OUTPUT A CHARACTER
00160A F00E 7E F363    A               JMP     EXKEY       TEST FOR A KEY
00161                          *
00162                          * INTERROGATE HARDWARE FOR RS232 TERMINAL
00163                          *
00164A F011 7F 810A    A TERMIN CLR     TERM        SET UP TERMINAL FLAG
00165A F014 7F 8005    A        CLR     IOPIAC      CLEAR THE I/O PIA
00166A F017 86 AA      A        LDAA    #$AA
00167A F019 B7 8004    A        STAA    IOPIAD      CHECK FOR A PIA
00168A F01C B1 8004    A        CMPA    IOPIAD
00169A F01F 26 14 F035         BNE     ENDTER      NOT THERE CONTINUE ON
00170A F021 7F 8004    A        CLR     IOPIAD      MAKE THE PORT ALL INPUTS
00171A F024 86 04      A        LDAA    #4
00172A F026 B7 8005    A        STAA    IOPIAC      CHANGE TO THE DATA REGISTER
00173A F029 44                 LSRA                LOOK FOR TERMINAL FLAG AT IOPIAD
00174A F02A B4 8004    A        ANDA    IOPIAD      TEST FOR RS232 OPTION
00175A F02D 27 06 F035         BEQ     ENDTER      NOT IN RS232 MODE
00176A F02F 86 08      A        LDAA    #8          HASH OFFSET FOR INIT. SCREEN SIZE
00177A F031 7C 810A    A        INC     TERM        YES, TALK TO AN EXTERNAL TERMINAL
00178A F034 39                 RTS                 SKIP THE CRT INITIALIZATION
00179A F035 4F           ENDTER CLRA                INIT. THE R2 BOARD
00180A F036 39                 RTS
00181A F037 CE 81FF    A ENT1   LDX     #$81FF
00182A F03A FF 812B    A        STX     USP
00183A F03D 8E 817F    A        LDS     #SP
00184A F040 CE F42B    A        LDX     #REGRMB     INITIALIZE THE USER SWI VECTOR
00185A F043 FF 8108    A        STX     USWIV
00186A F046 BD F8AF    A        JSR     ENNMI       ENABLES THE FRONT PANEL KEYBOARD
00187A F049 BD FCE5    A        JSR     CLR         CLEAR OUT THE BREAKPOINT TABLE
00188A F04C 8D C3 F011         BSR     TERMIN      CHECK FOR TERMINAL MODE
00189A F04E 26 2A F07A         BNE     US1         SKIP THE R2 INIT.
00190          F050    A CRTINT EQU    *
00191A F050 BD F699    A        JSR     CLEAR       CLEAR DISPLAY MEMORY
00192A F053 7F 8045    A        CLR     PIA1AC      CLEAR A SIDE PIA
00193A F056 7F 8047    A        CLR     PIA1BC      CLEAR B SIDE
00194A F059 7F 8044    A        CLR     PIA1AD      SET PA0-PA7 AS INPUTS
00195A F05C 7F 8046    A        CLR     PIA1BD      SET KEYBD DDR AS INPUTS
00196A F05F 86 04      A        LDAA    #04
00197A F061 B7 8047    A        STAA    PIA1BC      SET CONTROL REG FOR DDR ACCESS
00198A F064 B7 8045    A        STAA    PIA1AC      SET CONTROL REG FOR DDR ACCESS
00199A F067 B6 8046    A        LDAA    PIA1BD      GET R2 DISPLAY FORMAT CODE
00200A F06A BD F125    A        JSR     SFTA4R
00201A F06D 84 0C      A        ANDA    #$C
00202A F06F 81 0C      A        CMPA    #$C
00203A F071 27 12 F085         BEQ     USERS
00204A F073 F6 8044    A        LDAB    PIA1AD      CHECK R2 OPTION REG. FOR EUROPEAN
00205A F076 2A 02 F07A         BPL     US1         NOT SET
00206A F078 8B 02      A        ADDA    #2          POINT TO EUROPEAN HASH TABLE
```

```
00208                          *
00209                          * CALCULATE DISPLAY FORMAT ADDRESS
00210                          *
00211
00212              F07A   A US1     EQU     *
00213A F07A CE F774   A           LDX     #CRTADD   STARTING ADDR. OF FORMAT TABLE
00214A F07D BD F94D   A           JSR     ADDXA     CALCULATE THE HASH OFFSET
00215A F080 EE 00     A           LDX     0,X       GET ADDRESS OF SCREEN FORMAT TABLE
00216A F082 FF 8115   A US2     STX     FORMAT    SAVE FOR LATER
00217A F085 BD F6A9   A USERS   JSR     CRTIZ     INIT CRTC
00218
00219                          *
00220                          *  INIT PIA KEYBD,ACIA K.C. CASSETTE/ TERM
00221                          *
00222
00223              F088   A ENT     EQU     *
00224A F088 7F 8044   A           CLR     PIA1AD    SET KEYBD DDR AS INPUTS
00225A F08B 86 04     A           LDAA    #$04      SET KEYBD FOR NEGATIVE STROBE
00226A F08D B7 8045   A           STAA    PIA1AC
00227A F090 4A                    DECA              RESET CODE FOR ACIA (3)
00228A F091 B7 800A   A           STAA    ACIAS     RESET CASSETTE ACIA
00229A F094 B7 8008   A           STAA    ACIATS    RESET  TERM ACIA $8008
00230A F097 86 15     A           LDAA    #$15      SET UP ACIA FOR DATA TRANSFER
00231A F099 B7 8008   A           STAA    ACIATS    JMP  $6000  [7E 6000]
00232              F09C   A ENT2    EQU     *
00233A F09C BD F619   A           JSR     CRLF      MAKE THE PROMPT LEFT JUSITFIED
00234A F09F CE F780   A           LDX     #MCL2     PRINT HEADER
00235A F0A2 BD F198   A           JSR     MESAGE    PRINT DATA STRING
```

```
00237                    ************************************************************
00238                    *
00239                    * MAIN COMMAND/CONTROL LOOP
00240                    *
00241                    ************************************************************
00242           F0A5   A CONTRL EQU     *
00243                    *
00244                    * RESTORE STACK POINTER REGISTER
00245                    *
00246A F0A5 8E 817F   A          LDS     #SP        SP WAS INITIALIZED EARLIER
00247                    *
00248                    *
00249A F0A8 7F 8137   A          CLR     INIT2      INITIALIZATION FLAG
00250A F0AB 7F 812F   A          CLR     NFLAG      CLEAR THE TRACE FLAG
00251A F0AE 7F 8158   A          CLR     XTRFG      CLEAR THE D3BUG2 TRACE FLAG
00252A F0B1 B6 8088   A          LDAA    $8088      CLEAR PIA INTERRUPT FLAG
00253A F0B4 86 01     A          LDAA    #1
00254A F0B6 B7 8130   A          STAA    EXFLG      SET THE EXTERNAL ROM FLAG
00255A F0B9 BD F619   A          JSR     CRLF       NEW LINE
00256A F0BC 86 3D     A          LDAA    #'=
00257A F0BE BD F156   A          JSR     OUTCHR     PRINT THE PROMPT "="
00258                    *
00259           F0C1   A CTRL1   EQU     *
00260A F0C1 8D 1F F0E2           BSR     INCHR      GET AN ASCII CHARACTER
00261A F0C3 CE F6CF   A          LDX     #FCTABL    X:= ADDRESS OF JUMP TABLE
00262           F0C6   A NXTCHR  EQU     *
00263A F0C6 A1 00     A          CMPA    0,X        DOES INPUT CHAR MATCH?
00264A F0C8 27 0A F0D4           BEQ     GOODCH     YES, GOTO APPROPRIATE ROUTINE
00265A F0CA 08                   INX                ELSE, UPDATE INDEX INTO TABLE
00266A F0CB 08                   INX
00267A F0CC 08                   INX
00268A F0CD 8C F708   A          CPX     #FCTBEN    END OF TABLE REACHED?
00269A F0D0 26 F4 F0C6           BNE     NXTCHR     NO, TRY NEXT CHAR
00270A F0D2 20 D1 F0A5           BRA     CONTRL
00271
00272                    *
00273                    *   EXECUTE THE COMMAND ROUTINE
00274                    *
00275
00276           F0D4   A GOODCH  EQU     *
00277A F0D4 81 20     A          CMPA    #$20       TEST FOR COMMAND CHAR.
00278A F0D6 2D 04 F0DC           BLT     CH1        IT IS A CONTROL KEY DO NOT PRINT
00279A F0D8 8D 7C F156           BSR     OUTCHR     PRINT COMMAND
00280A F0DA 8D 6D F149           BSR     OUTS       OUTPUT SPACE AFTER COMMAND
00281A F0DC EE 01     A CH1      LDX     1,X        GET ADDRESS FROM JUMP TABLE
00282A F0DE AD 00     A          JSR     0,X        GOTO APPROPRIATE ROUTINE
00283A F0E0 20 C3 F0A5           BRA     CONTRL
```

```
00285                    ************************************************************
00286                    *
00287                    * KEYBOARD INPUT ROUTINE
00288                    *
00289                    ************************************************************
00290                    *
00291A F0E2 7D 810A  A INCHR  TST    TERM      IN TERMINAL MODE?
00292A F0E5 27 0B F0F2        BEQ    INCHR1    NO, INPUT FROM KEYBOARD
00293A F0E7 B6 8008  A INCHLP LDAA   ACIATS    READ ACIA COMMAND REGISTER
00294A F0EA 47                ASRA
00295A F0EB 24 FA F0E7        BCC    INCHLP
00296A F0ED B6 8009  A        LDAA   ACIATD    GET INPUTTED DATA BYTE
00297A F0F0 20 09 F0FB        BRA    ENDINC
00298A F0F2 B6 8045  A INCHR1 LDAA   PIA1AC    INPUT FROM THE PIA @ 8045
00299A F0F5 48                ASLA             CHECK FOR INPUT
00300A F0F6 24 EA F0E2        BCC    INCHR     NO DATA TRY AGAIN
00301A F0F8 B6 8044  A        LDAA   PIA1AD    INPUT CHAR.
00302A F0FB 84 7F    A ENDINC ANDA   #$7F      STRIP OFF UNUSED BIT
00303A F0FD 39                RTS
00304                    ************************************************************
00305                    *
00306                    * ASCHEX CONVERTS ASCII UPPER CASE AND NUMBERS TO HEX
00307                    *
00308                    ************************************************************
00309                    *
00310A F0FE 80 30    A ASCHEX SUBA   #$30
00311A F100 2B 10 F112        BMI    ENDASC
00312A F102 81 09    A        CMPA   #9        IS IT A NUMBER?
00313A F104 2F 0A F110        BLE    RETASC    IT IS RETURN WITH NUMBER
00314A F106 81 11    A        CMPA   #$11
00315A F108 2B 08 F112        BMI    ENDASC    NOT A HEX NUMBER
00316A F10A 81 16    A        CMPA   #$16
00317A F10C 2E 04 F112        BGT    ENDASC    NOT A HEX NUMBER
00318A F10E 80 07    A        SUBA   #7        CONVERT  TO HEX
00319A F110 0C           RETASC CLC
00320A F111 39                RTS
00321A F112 0D           ENDASC SEC              NON-HEX
00322A F113 39                RTS
00323                    ************************************************************
00324                    *
00325                    * HEX TO ASCII CONVERSION (UNPACKED HEX)
00326                    *
00327                    ************************************************************
00328                    *
00329A F114 81 0F    A HEXASC CMPA   #$F       A NUMBER?
00330A F116 2E FA F112        BGT    ENDASC
00331A F118 81 09    A        CMPA   #$9
00332A F11A 2E 03 F11F        BGT    ALPHA     NO, MUST BE AN ALPHA
00333A F11C 8B 30    A        ADDA   #$30      CONVERT
00334A F11E 39                RTS
00335A F11F 8B 37    A ALPHA  ADDA   #$37      CONVERT TO ASCII
00336A F121 39                RTS
```

```
00338                          *
00339                          * UNPACK HEX NUMBERS
00340                          *
00341A F122 16                 UNPACK TAB
00342A F123 C4 0F     A                ANDB   #$F        PUT LSN IN B
00343A F125 44                 SFTA4R LSRA
00344A F126 44                        LSRA
00345A F127 44                        LSRA
00346A F128 44                        LSRA
00347A F129 39                        RTS
00348                          ** OUTPUT ACCUMULATORS WITHOUT CHANGING THEM
00349                          *
00350A F12A 36                 OUTA   PSHA              SAVE IT FOR RETURN
00351A F12B 20 02 F12F                BRA    OUTARN
00352                          *
00353                          *
00354                          *
00355A F12D 36                 OUTB   PSHA              SAVE IT
00356A F12E 17                        TBA
00357A F12F 37                 OUTARN PSHB
00358A F130 8D E2 F114                BSR    HEXASC    CONVERT
00359A F132 25 02 F136                BCS    NOUT      NOT HEX
00360A F134 8D 20 F156                BSR    OUTCHR
00361A F136 33                 NOUT   PULB
00362A F137 32                        PULA
00363A F138 39                        RTS
00364                          *
00365A F139 8D EF F12A OUTAB   BSR    OUTA      OUTPUT A ACC.
00366A F13B 20 F0 F12D                BRA    OUTB      OUTPUT B ACC.
00367                          *
00368                          * OUTPUT PACKED HEX
00369                          *
00370A F13D CE 8118   A HXBUF0 LDX    #HEXBUF   OUTPUT 2 BYTE BUFFER
00371A F140 8D 0B F14D OUT4HS  BSR    OUT2H
00372A F142 20 03 F147                BRA    OUT2HS
00373A F144 CE 811A   A HEXBF2 LDX    #HEXBUF+2 OUTPUT 1 BYTE BUFFER
00374A F147 8D 04 F14D OUT2HS  BSR    OUT2H
00375A F149 86 20     A OUTS   LDAA   #$20      A SPACE
00376A F14B 20 09 F156                BRA    OUTCHR
00377                          *
00378                          * UNPACK AND OUTPUT BOTH NIBBLES IN ASCII
00379                          *
00380A F14D A6 00     A OUT2H  LDAA   0,X       GET PACKED BYTES
00381A F14F 8D D1 F122 OUT2HA  BSR    UNPACK    UNPACK INTO A AND B ACC.
00382A F151 8D D7 F12A                BSR    OUTA      OUTPUT BOTH ACCS.
00383A F153 08                        INX              NEXT BYTE
00384A F154 20 D7 F12D                BRA    OUTB
```

```
00386                    *
00387                    * DISPLAY AN ASCII CHAR.
00388                    *          JMP $6020
00389A F156 7D 810A  A OUTCHR TST    TERM      CHECK FOR RS232 MODE
00390A F159 27 0B F166         BEQ    OUTCR1    NO, CONTINUE WITH R2 DISPLAY
00391A F15B F6 8008  A ARNOUT LDAB   ACIATS    GET STATUS OF ACIA
00392A F15E 57                 ASRB
00393A F15F 57                 ASRB
00394A F160 24 F9 F15B         BCC    ARNOUT    WAIT TILL TRANS. REG. EMPTY
00395A F162 B7 8009  A         STAA   ACIATD    OUTPUT
00396A F165 39                 RTS              JMP $6020 7E 6020
00397
00398                    *
00399                    * OUTPUT TO THE R2M
00400                    *                       JMP $6020 7F 6020
00401
00402A F166 FF 810D  A OUTCR1 STX    CTEMP     SAVE REGS.
00403A F169 F6 8164  A         LDAB   CURPOS    GET CURSOR POSITION
00404A F16C C4 0F    A         ANDB   #$F       MASK TO 4K
00405A F16E CA 90    A         ORAB   #$90
00406A F170 F7 810B  A         STAB   CUTEMP
00407A F173 F6 8165  A         LDAB   CURPOS+1
00408A F176 F7 810C  A         STAB   CUTEMP+1
00409A F179 FE 810B  A         LDX    CUTEMP    GET CURSOR POSITION
00410A F17C A7 00    A         STAA   0,X       WRITE CHAR.
00411A F17E FE 8164  A         LDX    CURPOS    GET CURSOR POSITION
00412A F181 08                 INX              ADVANCE CURSOR
00413A F182 FF 8164  A         STX    CURPOS
00414A F185 7C 815A  A         INC    CCOUNT
00415A F188 B6 815A  A         LDAA   CCOUNT    #OF CHARS. ON THIS LINE
00416A F18B B1 8160  A         CMPA   CLINE     END OF LINE
00417A F18E 2B 05 F195         BMI    UPD       OUTPUT CR/LF AND RETURN
00418A F190 FE 810D  A         LDX    CTEMP
00419A F193 8D 1D F1B2         BSR    CRLFD
00420A F195 7E F682  A UPD     JMP    UPDATE    UPDATE CURSOR ON SCREEN AND RETURN
```

```
00422                    ********************************************************
00423                    *
00424                    * OUTPUT AN ASCII STRING
00425                    *
00426                    ********************************************************
00427A F198 36           MESAGE PSHA
00428A F199 A6 00     A  LOOPME LDAA   0,X       GET CHARACTER FROM STRING
00429A F19B 81 04     A         CMPA   #4        END OF STRING ?
00430A F19D 27 05 F1A4          BEQ    ENDMES    NO, CONTINUE
00431A F19F 8D B5 F156  MESGE1  BSR    OUTCHR    OUTPUT WHATS IN A
00432A F1A1 08                  INX
00433A F1A2 20 F5 F199          BRA    LOOPME    CONT. UNTIL END OF MESSAGE
00434A F1A4 32           ENDMES PULA
00435A F1A5 39           RETENT RTS
00436                    *
00437                    * ASK FOR BEGINNING AND ENDING ADDRESS
00438                    *
00439A F1A6 8D 0A F1B2  BEGADD  BSR    CRLFD     PRINT CR/LF
00440A F1A8 CE F7C3   A         LDX    #MCL4     BEGINNING ADDR. MESSAGE
00441A F1AB 8D EB F198          BSR    MESAGE    OUTPUT MESSAGE
00442A F1AD 8D 19 F1C8          BSR    INADD1    GET ADDR. IN X
00443A F1AF FF 813A   A         STX    BEGA
00444A F1B2 7E F619   A  CRLFD  JMP    CRLF      OUTPUT A CR/LF AND RETURN
00445A F1B5 8D EF F1A6  ADDRES  BSR    BEGADD
00446A F1B7 CE F7CC   A  ENDADD LDX    #MCL5     ENDING ADDR. MESSAGE
00447A F1BA 8D DC F198          BSR    MESAGE
00448A F1BC 8D 0A F1C8          BSR    INADD1    ENDING ADDR. IS RETRIEVE
00449A F1BE FF 813C   A         STX    ENDA      SAVE IT
00450A F1C1 7E F149   A         JMP    OUTS      MOVE OVER ONE SPACE
```

```
00452                    ************************************************************
00453                    *
00454                    *INPUT 2/4 HEX ASCII BYTES AND PACK INTO 1 OR 2 BYTES.
00455                    *TERMINATED BY EITHER A CR OF A \. IF NO CHAR. ARE
00456                    *INPUTTED IT ABORTS TO MONITOR.
00457                    *
00458                    ************************************************************
00459                    *
00460A F1C4 8D 0C F1D2 INBYT1 BSR    INBYTE   GET THE BYTE FIRST
00461A F1C6 20 02 F1CA        BRA    TSTENT   CHECK IF ANY DATA WAS ENTERED
00462A F1C8 8D 0F F1D9 INADD1 BSR    INADDR   GET THE ADDR. FIRST
00463A F1CA 7D 8134  A TSTENT TST    ROLFLG   CHECK FOR ENTERED DATA
00464A F1CD 26 D6 F1A5        BNE    RETENT   DATA WAS ENTERED
00465A F1CF 7E F0A5  A        JMP    CONTRL   DATA WAS NOT ENTERED
00466                    *
00467A F1D2 86 01    A INBYTE LDAA   #1       PACK A BYTE
00468A F1D4 FF 8111  A        STX    XTMP3    SAVE THE X REG.
00469A F1D7 20 01 F1DA        BRA    CLEARF
00470A F1D9 4F           INADDR CLRA             PACK 2 BYTES
00471A F1DA B7 8135  A CLEARF STAA   FLG24
00472A F1DD 7F 8134  A        CLR    ROLFLG   FIRST TIME THROUGH
00473A F1E0 7F 8154  A        CLR    CRFLAG   CLEAR THE CARRIAGE RETURN FLAG
00474A F1E3 BD F0E2  A LOOPAD JSR    INCHR    INPUT A CHAR.
00475A F1E6 CE F223  A        LDX    #COMMD
00476A F1E9 A1 00    A COMLOP CMPA   0,X
00477A F1EB 26 07 F1F4        BNE    ARNCOM
00478A F1ED E6 01    A        LDAB   1,X      GET DECODE
00479A F1EF F7 8117  A        STAB   KEY      SAVE IT
00480A F1F2 20 13 F207        BRA    ENDAD
00481A F1F4 08           ARNCOM INX
00482A F1F5 08                  INX
00483A F1F6 8C F22F  A        CPX    #ENDCOM  END OF COMMAND TABLE?
00484A F1F9 26 EE F1E9        BNE    COMLOP
00485A F1FB BD F0FE  A        JSR    ASCHEX   CONVERT TO HEX
00486A F1FE 25 E3 F1E3        BCS    LOOPAD   NOT A HEX CHAR. IGNORE
00487A F200 BD F12A  A ALTIN  JSR    OUTA     OUTPUT THE HEX KEY IN A
00488A F203 8D 35 F23A        BSR    PACKIT   PACK THE BYTE
00489A F205 20 DC F1E3        BRA    LOOPAD   CONTINUE INPUT
00490A F207 81 0D    A ENDAD  CMPA   #$D
00491A F209 26 03 F20E        BNE    ENDLF
00492A F20B 7C 8154  A        INC    CRFLAG
00493A F20E 7D 8134  A ENDLF  TST    ROLFLG
00494A F211 27 08 F21B        BEQ    RETAD    NO ADDR. WAS ENTER RETURN
00495A F213 7D 8135  A OBTN   TST    FLG24    WHICH HOLDS ADDR.
00496A F216 27 07 F21F        BEQ    OBTNX    ITS A 2 BYTE WORD
00497A F218 B6 811A  A        LDAA   HEXBUF+2 GET BYTE
00498A F21B FE 8111  A RETAD  LDX    XTMP3    RETRIEVE
00499A F21E 39                RTS
00500A F21F FE 8118  A OBTNX  LDX    HEXBUF   GET PACKED ADDR. / WORD
00501A F222 39                RTS
```

```
00503                    *
00504                    * COMMAND CONVERSION TABLE
00505                    *
00506
00507           F223   A COMMD  EQU     *
00508A F223     4D     A         FCB     'M        BACKUP
00509A F224     00     A         FCB     0
00510A F225     5E     A         FCB     '^        BACKUP
00511A F226     00     A         FCB     0
00512A F227     0D     A         FCB     $D        CR
00513A F228     01     A         FCB     1
00514A F229     0A     A         FCB     $A        LF
00515A F22A     03     A         FCB     3
00516A F22B     4F     A         FCB     'O        OFFSET
00517A F22C     04     A         FCB     4
00518A F22D     20     A         FCB     '
00519A F22E     05     A         FCB     5
00520           F22F   A ENDCOM EQU     *
00521                    *
00522                    * PACK ROUTINE THAT PACKS TWO OR FOUR BYTES
00523                    *
00524A F22F C6 01     A PACK2B LDAB    #1
00525A F231 20 01 F234    BRA     PACKHX
00526A F233 5F            PACK4B CLRB
00527A F234 F7 8135   A PACKHX STAB    FLG24
00528A F237 7F 8134   A         CLR     ROLFLG
00529A F23A 37            PACKIT PSHB              SAVE B REG
00530A F23B BD FB44   A         JSR     ROLENT    PACK A INTO HEXBUF:+1 FOR 4 BYTE
00531                    *                         PACK A INTO HEXBUF+2 FOR 2 BYTE
00532A F23E 33                    PULB
00533A F23F 39                    RTS
```

```
00535                    ************************************************************
00536                    *
00537                    *
00538                    *  LOAD FROM TAPE AT EITHER 300 OR 1200 BAUD
00539                    *
00540                    ************************************************************


00542           F240   A LOAD   EQU    *
00543A F240 4F                   CLRA             NOT THE VERIFY ROUTINE
00544A F241 20 02 F245           BRA    BEGLD
00545A F243 86 01      A VERFY   LDAA   #1       THIS IS THE VERIFY ROUTINE
00546A F245 B7 8132    A BEGLD   STAA   VERIFY
00547A F248 8D 20 F26A           BSR    BAUD      DETERMINE WHICH FORMAT IS DESIRED
00548A F24A 26 03 F24F           BNE    LV1200    DO A 1200 BAUD FORMATTED TAPE
00549A F24C 7C 8131    A         INC    S3FLAG    DO A  300 BAUD FORMATED TAPE
00550A F24F 86 01      A LV1200  LDAA   #1        SET UP A FLAG IN THE TAPE PROGRAM
00551A F251 BD FEC0    A         JSR    GO1       GO LOAD THE TAPE
00552A F254 7D 8133    A         TST    ERROR     CHECK FOR A TAPE ERROR
00553A F257 27 05 F25E           BEQ    ENDLD     NO ERROR
00554A F259 CE F7E8    A LDERR   LDX    #ERRMSG   PRINT CHKSUM ? FOR CHECK SUM ERROR
00555A F25C 20 03 F261           BRA    ENDLD1
00556A F25E CE F7C3    A ENDLD   LDX    #MCL4     PRINT "BEG ADDR ="
00557A F261 BD F198    A ENDLD1  JSR    MESAGE
00558A F264 CE 8129    A         LDX    #UPC      GET BEGING ADDR.
00559A F267 7E F140    A         JMP    OUT4HS    PRINT ON SCREEN
00560           8131   A S3FLAG  EQU    $8131
00561           8132   A VERIFY  EQU    $8132
00562           8133   A ERROR   EQU    $8133
00563           FEC0   A GO1     EQU    $FEC0
00564                    *
00565                    * SUBROUTINE TO DETERMINE THE FORMAT OF THE TAPE
00566                    *
00567           F26A   A BAUD    EQU    *
00568A F26A CE F7EF    A         LDX    #PLMSG    OUTPUT THE PROMPT
00569A F26D BD F198    A         JSR    MESAGE
00570A F270 7F 8131    A         CLR    S3FLAG
00571A F273 BD F1D2    A         JSR    INBYTE    GET THE ANSWER
00572A F276 36                   PSHA
00573A F277 BD F149    A         JSR    OUTS      MOVE OVER ONE SPACE
00574A F27A 32                   PULA
00575A F27B 7D 8154    A         TST    CRFLAG    TEST FOR A 'CR
00576A F27E 27 02 F282           BEQ    ARBAUD    ABORT!
00577A F280 31                   INS              ABORT ROUTINE
00578A F281 31                   INS
00579A F282 81 03      A ARBAUD  CMPA   #3        TEST FOR 300 BAUD
00580A F284 39                   RTS
```

```
00582                    ************************************************************
00583                    *
00584                    * PUNCH EITHER 300 OR 1200 BAUD FORMATED TAPES
00585                    *
00586                    ************************************************************
00587          F285  A PUNCH  EQU    *
00588A F285 8D E3 F26A         BSR    BAUD
00589A F287 26 03 F28C         BNE    PNCH12
00590A F289 7C 8131  A         INC    S3FLAG   SET FOR A 300 BAUD TAPE
00591A F28C BD F1B5  A PNCH12  JSR    ADDRES   ASK FOR BEGINNING AND ENDING ADDR.
00592A F28F 7E FFA7  A         JMP    PUNCHA   GO TO PUNCH ROUTINE IN D3BUG
00593          FFA7  A PUNCHA EQU    $FFA7
00594                    ********************************************
00595                    *
00596                    *   OFFSET LOAD ROUTINE
00597                    *    ENTER THE NEW STARTING ADDR.
00598                    *
00599                    ********************************************
00600A F292 7F 8132  A OFLOAD CLR    VERIFY   WANT TO LOAD A TAPE
00601A F295 8D D3 F26A         BSR    BAUD     DETERMINE THE BAUD RATE FORMAT
00602A F297 26 03 F29C         BNE    OF1200   DO A 1200 BAUD FORMATED TAPE
00603A F299 7C 8131  A         INC    S3FLAG   DO A 300 BAUD FORMATTED TAPE
00604A F29C BD F1A6  A OF1200  JSR    BEGADD   GET OFFSET ADDRESS
00605A F29F 86 11    A         LDAA   #%00010001
00606A F2A1 B7 8138  A         STAA   LDFLAG   SET LOAD FLAG
00607A F2A4 B7 800A  A         STAA   ACIAS
00608          8138  A LDFLAG EQU    $8138
00609A F2A7 7D 8131  A         TST    S3FLAG
00610A F2AA 26 1C F2C8         BNE    OFF300
00611A F2AC 8D 53 F301 LOOP12  BSR    XINCHR
00612A F2AE C1 96    A         CMPB   #$96     LOOK FOR SYNC BYTE
00613A F2B0 26 FA F2AC         BNE    LOOP12
00614A F2B2 4F         LOP12A  CLRA           START CHECKSUM
00615A F2B3 8D 4C F301         BSR    XINCHR   GET # OF BYTES
00616A F2B5 27 4D F304         BEQ    QUIT     END OF FILE
00617A F2B7 8D 25 F2DE         BSR    MPIN     LOAD A FILE
00618A F2B9 8D 46 F301         BSR    XINCHR   GET CHECKSUM
00619A F2BB 4C                 INCA
00620A F2BC 27 02 F2C0         BEQ    ARNERR
00621A F2BE 20 99 F259         BRA    LDERR    CHECKSUM ERROR
00622A F2C0 8D 3F F301 ARNERR  BSR    XINCHR   START AGAIN
00623A F2C2 C1 96    A         CMPB   #$96
00624A F2C4 27 EC F2B2         BEQ    LOP12A   CONTINUE
00625A F2C6 20 91 F259         BRA    LDERR    SOMETHING WRONG
00626A F2C8 8D 37 F301 OFF300  BSR    XINCHR
00627A F2CA C1 42    A         CMPB   #'B      LOOK FOR BEGINNING CHAR
00628A F2CC 27 09 F2D7         BEQ    BLKRD    READ A BLOCK OF DATA
00629A F2CE C1 47    A         CMPB   #'G      LOOK FOR END OF FILE
00630A F2D0 26 F6 F2C8         BNE    OFF300   KEEP LOOKING
00631A F2D2 7F 8131  A         CLR    S3FLAG   CLEAN UP FLAGS
00632A F2D5 20 2D F304         BRA    QUIT
00633A F2D7 8D 28 F301 BLKRD   BSR    XINCHR   GET # OF BYTES
00634A F2D9 5C                 INCB            ADJUST COUNT
00635A F2DA 8D 02 F2DE         BSR    MPIN     LOAD IN A BLOCK OF DATA
00636A F2DC 20 EA F2C8         BRA    OFF300   TRY FOR ANOTHER BLOCK
```

```
00638                    **********************************
00639                    *
00640                    * MODIFIED PIN
00641                    *
00642                    **********************************
00643A F2DE CE 8139  A MPIN   LDX    #LDTBLE
00644A F2E1 E7 00    A        STAB   0,X       SAVE # OF BYTES
00645A F2E3 8D 1C F301        BSR    XINCHR
00646A F2E5 7D 8137  A        TST    INIT2
00647A F2E8 26 03 F2ED        BNE    NXTADD    SAVE ORIGINAL MSB ADDR
00648A F2EA F7 813C  A        STAB   ENDA      SAVE ORIGINAL MSB ADDR
00649A F2ED 8D 12 F301 NXTADD BSR    XINCHR    GET LSB OF ORIGINAL ADDR
00650A F2EF 7D 8137  A        TST    INIT2
00651A F2F2 26 06 F2FA        BNE    NXTAD1
00652A F2F4 F7 813D  A        STAB   ENDA+1
00653A F2F7 7C 8137  A        INC    INIT2
00654A F2FA BD FF18  A NXTAD1 JSR    PIN1      GOTO UPPER TWO K PROGRAM
00655A F2FD FF 813A  A        STX    BEGA      UPDATE BLOCK ADDR.
00656A F300 39                RTS
00657          FF18  A PIN1   EQU    $FF18
00658          8139  A LDTBLE EQU    $8139
00659A F301 7E FF48  A XINCHR JMP    $FF48     INPUT A CHAR. FROM CASSETTE
00660                    *
00661                    * QUIT THE LOAD ROUTINE
00662                    *
00663A F304 CE F7F6  A QUIT   LDX    #ADMESG   PRINT "ADDR. ON TAPE ="
00664A F307 BD F198  A        JSR    MESAGE
00665A F30A CE 813C  A        LDX    #ENDA     GET ORIGINAL ADDR. OF TAPE FILE
00666A F30D 7E F140  A        JMP    OUT4HS    PRINT ON SCREEN
```

```
00668                    ************************************************************
00669                    *
00670                    * CHANGE MEMORY (M AAAA DD NN)
00671                    *
00672                    ************************************************************
00673           F310   A CHANGE EQU    *
00674A F310 7F 8132   A         CLR    INIT
00675A F313 7F 8154   A         CLR    CRFLAG
00676A F316 7F 8137   A         CLR    INIT2     CLEAR OFSET FLAG
00677A F319 BD F1D9   A         JSR    INADDR    GET MEMORY ADDR.
00678A F31C B7 812E   A         STAA   MEMFLG    SET FLAG WITH A NON-ZERO VALUE
00679A F31F FF 8111   A         STX    XTMP3
00680A F322 C1 04     A         CMPB   #$4       LOOK FOR OFFSET COMMAND
00681A F324 26 11 F337          BNE    MEM       CONTINUE NORMAL OPERATION
00682A F326 7C 8132   A         INC    INIT      PREVENT DOUBLE PRINTING AFTER PUSH
00683A F329 7C 8137   A         INC    INIT2     DO AN OFFSET CALCULATION
00684A F32C BD F149   A MEM1    JSR    OUTS      SPACE
00685A F32F 86 4F     A         LDAA   #'O
00686A F331 BD F156   A         JSR    OUTCHR
00687A F334 BD F149   A         JSR    OUTS
00688A F337 BD FADC   A MEM     JSR    MEMCH     GO TO UPPER 2K PROGRAM
00689A F33A B6 8117   A         LDAA   KEY       GET DECODED KEY
00690A F33D 81 05     A         CMPA   #5        LOOK FOR "\"
00691A F33F 27 16 F357          BEQ    SLASH
00692A F341 81 04     A         CMPA   #4        LOOK FOR OFFSET COMMAND
00693A F343 27 E7 F32C          BEQ    MEM1
00694A F345 81 01     A         CMPA   #1
00695A F347 27 16 F35F          BEQ    ENDCHG    A CARRAGE RETURN WAS ENTERED
00696A F349 BD F619   A         JSR    CRLF      NEW LINE
00697A F34C BD F13D   A         JSR    HXBUF0    OUTPUT ADDR. IN HEXBUF
00698A F34F BD F144   A         JSR    HEXBF2    OUTPUT ADDR. IN HEXBUF+2
00699A F352 FE 8111   A         LDX    XTMP3     RETRIEVE X REG.
00700A F355 20 E0 F337          BRA    MEM       NEXT ENTRY
00701A F357 BD F149   A SLASH   JSR    OUTS
00702A F35A BD F147   A         JSR    OUT2HS    OUTPUT DATA BYTE
00703A F35D 20 D8 F337          BRA    MEM       GO WAIT FOR ANY CHANGE
00704A F35F 7F 812E   A ENDCHG  CLR    MEMFLG
00705A F362 39                  RTS              RETURN TO MONITOR
00706           F363   A EXKEY   EQU    *
00707A F363 7D 8158   A         TST    XTRFG     TRACE?
00708A F366 26 51 F3B9          BNE    RENTRA    A TRACE HAS OCCURRED.
00709A F368 BD F011   A         JSR    TERMIN
00710A F36B 26 07 F374          BNE    CKTERM    SEE IF A KEY HAS BEEN PRESSED
00711A F36D B6 8045   A         LDAA   PIA1AC    SCAN KEYBOARD
00712A F370 48                  ASLA
00713A F371 25 58 F3CB          BCS    CONTL
00714A F373 39                  RTS              NO KEY PENDING
00715A F374 B6 8008   A CKTERM  LDAA   ACIATS    GET CONTROL REGS.
00716A F377 47                  ASRA
00717A F378 25 51 F3CB          BCS    CONTL     A KEY, GIVE CONTROL TO D3BUG2
00718A F37A 39                  RTS              CONTINUE FRONT PANEL OPERATIONS
```

```
00720                          *
00721                          * EXGET, IS USED TO GET EITHER AN ADDR. OR A DATA BYTE
00722                          *
00723              F37B  A EXGET  EQU    *
00724A F37B 7D 8137  A         TST    INIT2    GET AN ADDR.
00725A F37E 26 17 F397         BNE    ADDRIN   YES
00726A F380 BD F1D2  A BYTEIN  JSR    INBYTE   GET ONE BYTE
00727A F383 7D 8134  A         TST    ROLFLG
00728A F386 27 0B F393         BEQ    EXARN
00729A F388 7D 812E  A         TST    MEMFLG   IS THIS FROM CHANGE?
00730A F38B 27 06 F393         BEQ    EXARN    NO.
00731A F38D A7 00    A         STAA   0,X      SAVE THE CHANGE
00732A F38F A1 00    A         CMPA   0,X      CHECK FOR CORRECTNESS
00733A F391 26 09 F39C         BNE    BADBYT   IT'S NOT
00734A F393 B6 8117  A EXARN   LDAA   KEY      RETRIEVE DECODED KEY
00735A F396 39                 RTS
00736A F397 BD F1D9  A ADDRIN  JSR    INADDR   GET ADDR FROM KEYBOARD
00737A F39A 20 F7 F393         BRA    EXARN
00738A F39C 86 3F    A BADBYT  LDAA   #'?      BAD BYTE PROMPT
00739A F39E BD F156  A         JSR    OUTCHR
00740A F3A1 20 D8 F37B         BRA    EXGET
00741              812E  A MEMFLG EQU    $812E
00742              FADC  A MEMCH  EQU    $FADC
00743                          *
00744                          * TRACE ROUTINE FOR 1 OR MORE TRACES
00745                          *
00746              F3A3  A TRACE  EQU    *
00747A F3A3 BD F1C4  A         JSR    INBYT1   GET # OF BYTES TO TRACE
00748A F3A6 B6 811A  A         LDAA   HEXBUF+2 GET # OF TRACES
00749A F3A9 26 02 F3AD         BNE    XTRACE   MULTI. TRACE
00750A F3AB 86 01    A TRACE1  LDAA   #1       ONE INSTRUCTION TO TRACE
00751A F3AD B7 810F  A XTRACE  STAA   NMBTRC   SAVE THE # OF TRACES
00752A F3B0 7C 8158  A         INC    XTRFG    SET THE TRACE FLAG
00753A F3B3 B6 8088  A TRACLP  LDAA   $8088     RESET PIA INTERRUPT FLAG
00754A F3B6 7E FCD5  A         JMP    TRACE2   GO TRACE AN INSTRUCTION
00755A F3B9 2B 7B F436 RENTRA  BMI    GOTO1    CONTINUE EXECUTION
00756A F3BB BD F011  A         JSR    TERMIN   REINITIALIZE RS232 FLAG IF NEEDED
00757A F3BE 86 01    A         LDAA   #1
00758A F3C0 B7 8130  A         STAA   EXFLG
00759              812F  A NFLAG  EQU    $812F
00760A F3C3 BD F44F  A         JSR    REGDIS   REENTRY POINT FOR TRACE
00761A F3C6 7A 810F  A         DEC    NMBTRC   ONE LESS
00762A F3C9 26 E8 F3B3         BNE    TRACLP   TRACE ONE MORE
00763A F3CB 7E F0A5  A CONTL   JMP    CONTRL   STACK IS DESTROYED BY TRACE SO JMP
00764              8158  A XTRFG  EQU    $8158
00765              FCD5  A TRACE2 EQU    $FCD5    TRACE ROUTINE IN TOP TWO K
```

```
00767                    ********************************************************
00768                    *
00769                    *    EXAMINE MEMORY BLOCK ROUTINE
00770                    *
00771                    *    PERIOD PRINTED IF NOT ASCII
00772                    *     PRINTS LINES AND WAITS FOR ANY KEYBOARD CHAR
00773                    *     TO CONTINUE OR CR TERMINATES ROUTINE
00774                    ********************************************************
00775           F3CE   A MEMBLK EQU    *
00776A F3CE BD F1A6   A         JSR    BEGADD   GET STARTING ADDR.
00777A F3D1 F6 815E   A         LDAB   LSCREN   OUTPUT NUMBER OF LINES
00778A F3D4 F7 8111   A         STAB   RLINES
00779A F3D7 CE 813A   A STRTEX  LDX    #BEGA    GET THE ADDR. OF THE ADDR.
00780A F3DA BD F140   A         JSR    OUT4HS   PRINT ADDR. ON R2
00781A F3DD 8D 26 F405          BSR    ESETUP   SETUP # OF CHARS TO PRINT
00782A F3DF 37          EHOUT   PSHB
00783A F3E0 BD F147   A         JSR    OUT2HS   OUTPUT HEX BYTES
00784A F3E3 33                  PULB
00785A F3E4 5A                  DECB            ONE LESS CHAR. TO PRINT
00786A F3E5 26 F8 F3DF          BNE    EHOUT    ENDOF OUTPUT?
00787A F3E7 8D 1C F405          BSR    ESETUP   GET READY TO OUTPUT ASCII
00788A F3E9 BD F416   A EAOUT   JSR    OUTAS    OUTPUT ASCII EQUIVALENT
00789A F3EC 5A                  DECB            ONE LESS CHAR.
00790A F3ED 26 FA F3E9          BNE    EAOUT
00791A F3EF FF 813A   A         STX    BEGA     START NEW LINE
00792A F3F2 BD F619   A         JSR    CRLF     NEWLINE
00793A F3F5 7A 8111   A         DEC    RLINES   ONE LESS LINE
00794A F3F8 26 DD F3D7          BNE    STRTEX   CONT. UNTIL ALL LINES OUTPUTED
00795A F3FA 7C 8111   A         INC    RLINES   PREPARE FOR A NEW LINE
00796A F3FD BD F0E2   A         JSR    INCHR    GET COMMAND KEY
00797A F400 81 0D     A         CMPA   #$D      CARRIAGE RETURN?
00798A F402 26 D3 F3D7          BNE    STRTEX   NO, OUTPUT ANOTHER LINE
00799A F404 39                  RTS
```

```
00801                          *
00802                          * SUBROUTINE TO SETUP # OF MEMORY LOCATIONS TO DISPLAY
00803                          *
00804A F405 F6 8160  A ESETUP LDAB   CLINE    NUMBER OF CHARS PER LINE
00805A F408 54                 LSRB
00806A F409 54                 LSRB          DIVIDE BY 4 FOR # TO DISPLAY
00807A F40A C1 14    A         CMPB   #$14     IS IT 80 CHAR/LINE
00808A F40C 26 02 F410         BNE    DEC2     NO, ONLY SUBTRACT TWO CHARS.
00809A F40E C0 02    A         SUBB   #2       SUBTRACT 2 CHARS. TO DISPLAY 16 CH
00810A F410 C0 02    A DEC2    SUBB   #2       SUBTRACT 2 CHARS. FROM DISPLAY
00811A F412 FE 813A  A         LDX    BEGA     GET START OF BLOCK
00812A F415 39                 RTS
00813                          *
00814                          * OUTPUT ASCII REPRESENTATION OF MEMORY LOCATION
00815                          *
00816A F416 37         OUTAS   PSHB
00817A F417 A6 00    A         LDAA   0,X      GET MEMORY BYTE
00818A F419 2B 08 F423         BMI    ENDAS    NOT ASCII
00819A F41B 81 7A    A         CMPA   #$7A
00820A F41D 2E 04 F423         BGT    ENDAS    DEFAULT TO '.'
00821A F41F 81 20    A         CMPA   #$20
00822A F421 2C 02 F425         BGE    ENDOUT   ITS A CONTROL CHAR.
00823A F423 86 2E    A ENDAS   LDAA   #'.      NONASCII CHAAR.
00824A F425 BD F156  A ENDOUT  JSR    OUTCHR
00825A F428 33                 PULB
00826A F429 08                 INX             ONE LESS CHARACTER
00827A F42A 39                 RTS
00828          8111  A RLINES  EQU    $8111
00829                          ********************************************************
00830                          *
00831                          * GO TO REQUESTED
00832                          *
00833                          ********************************************************


00835A F42B BD FDFD  A REGRMB  JSR    RMBKPT
00836A F42E 20 9B F3CB         BRA    CONTL
00837          FDFD  A RMBKPT  EQU    $FDFD    REMOVE BREAKPOINTS
00838          F430  A GOTO    EQU    *
00839A F430 BD F1C8  A         JSR    INADD1   GET STARTING ADDR.
00840A F433 54                 LSRB            CHECK FOR TYPE OF COMMAND
00841A F434 24 95 F3CB         BCC    CONTL    AN ILLEGAL COMMAND
00842A F436 7F 8136  A GOTO1   CLR    FNCFLG   NORMAL GOTO
00843A F439 86 01    A         LDAA   #1
00844A F43B B7 810F  A         STAA   NMBTRC   SETUP FOR RETURN TO EXKEY
00845A F43E B7 8158  A         STAA   XTRFG
00846A F441 7E FC64  A         JMP    GO       INSTALL ANY BREAKPOINTS AND EXECUT
00847          8136  A FNCFLG  EQU    $8136    THEY MULTIPLEX ITS USE
00848          FC64  A GO      EQU    $FC64
00849A F444 86 80    A CONT    LDAA   #$80     CONTINUE EXECUTION IN THE USER'S
00850A F446 B7 8158  A         STAA   XTRFG    PROGRAM INSTEAD OF DISPLAYING HIS
00851A F449 7F 8134  A         CLR    ROLFLG
00852A F44C 7E F3AB  A         JMP    TRACE1
```

```
00854                    ************************************************************
00855                    *
00856                    *  R COMMAND
00857                    *
00858                    * PRINT STACK CONTENTS
00859                    *
00860                    ************************************************************


00862A F44F 8D 7B F4CC REGDIS BSR     CRLF1     START NEW LINE
00863A F451 CE F78B  A         LDX     #MCL3     DISPLAY HEADER
00864A F454 BD F198  A         JSR     MESAGE
00865A F457 B6 8160  A         LDAA    CLINE
00866A F45A 81 20    A         CMPA    #32       32 CHAR. PER LINE SCREEN
00867A F45C 27 06 F464         BEQ     SKPMSG    YES, DON'T PRINT OPTIONAL REGISTER
00868A F45E CE F7AB  A         LDX     #MCL35
00869A F461 BD F198  A         JSR     MESAGE    PRINT HEADER
00870A F464 7F 8156  A SKPMSG  CLR     DISFLG
00871A F467 8D 0F F478 SKPMG1  BSR     SINGLE    PRINT REGISTERS HORIZONTALLY
00872A F469 8D 30 F49B         BSR     DOUBLE
00873A F46B BD F52F  A         JSR     PAGE
00874A F46E 8D 6B F4DB         BSR     OPTION
00875A F470 39                 RTS
00876          F471  A PSTAK1  EQU     *
00877A F471 8D DC F44F         BSR     REGDIS    DISPLAY REGISTERS HORIZONTALLY
00878                    *
00879                    * NOW PRINT THEM VERTICALLY AND ALLOW CHANGES
00880                    *
00881A F473 7C 8156  A CHNGRG  INC     DISFLG    VERTICAL PRINT FLAG
00882A F476 20 EF F467         BRA     SKPMG1
00883                    *
00884                    * SUBROUTINE TO PRINT THE SINGLE BYTE REGS.
00885                    *
00886          F478  A SINGLE  EQU     *
00887A F478 8D 52 F4CC         BSR     CRLF1     GOTO NEXT LINE
00888A F47A CE 8124  A         LDX     #UCC      START OF REGS.
00889A F47D BD F147  A LOOP1B  JSR     OUT2HS    PRINT THE SINGLE BYTE REG.
00890A F480 7D 8156  A         TST     DISFLG    HORIZONTAL OR VERTICAL PRINT
00891A F483 27 10 F495         BEQ     NEXT1     DO A HORIZONTAL PRINT
00892A F485 BD F1D2  A         JSR     INBYTE    GET ANY CHANGE
00893A F488 7D 8134  A         TST     ROLFLG    BYTE ENTERED?
00894A F48B 27 04 F491         BEQ     NXTREG    NO, GO TO THE NEXT REGISTER
00895A F48D 09                 DEX               BACKUP
00896A F48E A7 00    A         STAA    0,X       STORE CHANGE
00897A F490 08                 INX               NEXT REG.
00898A F491 8D 3B F4CE NXTREG  BSR     TSTCR     CHECK FOR USER TERMINATION
00899A F493 8D 37 F4CC         BSR     CRLF1     ADVANCE TO NEXT LINE
00900A F495 8C 8127  A NEXT1   CPX     #UCC+3    END OF SINGLE BYTE REGS.?
00901A F498 26 E3 F47D         BNE     LOOP1B    NO,CONTINUE
00902A F49A 39                 RTS               FINISHED
```

```
00904                    *
00905                    * SUBROUTINE TO DISPLAY DOUBLE BYTE REGISTERS
00906                    *
00907A F49B CE 8127  A DOUBLE LDX    #UX       START OF USER DOUBLE BYTE REGS.
00908A F49E BD F140  A LOOP2B JSR    OUT4HS    OUTPUT DOUBLE BYTE REG.
00909A F4A1 7D 8156  A        TST    DISFLG    HORIZONTAL OF VERTICAL PRINT?
00910A F4A4 27 20 F4C6        BEQ    NEXT2     HORIZONTAL PRINT
00911A F4A6 FF 8111  A        STX    XTMP3     SAVE
00912A F4A9 BD F1D9  A        JSR    INADDR    GET ANY CHANGE
00913A F4AC FE 8111  A        LDX    XTMP3
00914A F4AF 7D 8134  A        TST    ROLFLG    WAS A CHANGE MADE
00915A F4B2 27 0E F4C2        BEQ    NEXTRG    NO, DON'T CHANGE IT
00916A F4B4 09                DEX
00917A F4B5 09                DEX              BACKUP
00918A F4B6 B6 8118  A        LDAA   HEXBUF    GET MSB OF CHANGE
00919A F4B9 A7 00    A        STAA   0,X
00920A F4BB 08                INX
00921A F4BC B6 8119  A        LDAA   HEXBUF+1  GET LSB
00922A F4BF A7 00    A        STAA   0,X
00923A F4C1 08                INX
00924A F4C2 8D 0A F4CE NEXTRG BSR    TSTCR     USER TERMINATION?
00925A F4C4 8D 06 F4CC        BSR    CRLF1
00926A F4C6 8C 812D  A NEXT2  CPX    #UX+6     LAST REG?
00927A F4C9 26 D3 F49E        BNE    LOOP2B    NO CONTINUE
00928A F4CB 39                RTS
00929A F4CC 20 5C F52A CRLF1  BRA    CRLF2
00930                    *
00931                    * SUBROUTINE TO TEST FOR CARRIAGE RETURN
00932                    *
00933A F4CE 7D 8154  A TSTCR  TST    CRFLAG    A "CR"?
00934A F4D1 27 04 F4D7        BEQ    ENDCR     NO, CONTINUE
00935A F4D3 31                INS
00936A F4D4 31                INS
00937A F4D5 31                INS
00938A F4D6 31                INS              RETURN TO MONITOR
00939A F4D7 39          ENDCR RTS
00940A F4D8 7E F149  A OUTS1  JMP    OUTS
```

```
00942                          *
00943                          * DISPLAY OPTIONAL REGISTERS
00944                          *
00945A F4DB 7D 8156  A OPTION  TST     DISFLG    ALLOW CHANGING OF OPTIONAL REGS.
00946A F4DE 26 08 F4E8         BNE     CONTOP    IN THE 32x16 DISPLAY
00947A F4E0 B6 8160  A         LDAA    CLINE
00948A F4E3 81 20    A         CMPA    #32       32 CHAR. DISP.
00949A F4E5 2E 01 F4E8         BGT     CONTOP    NO, CONTINUE
00950A F4E7 39                 RTS
00951A F4E8 CE 8168  A CONTOP  LDX     #BEGOPT   BEGINNING OF TABLE
00952A F4EB FF 8111  A OPLOOP  STX     XTMP3     SAVE IT
00953A F4EE EE 00    A         LDX     0,X
00954A F4F0 8D E6 F4D8         BSR     OUTS1     A SPACE FIRST
00955A F4F2 BD F140  A         JSR     OUT4HS    PRINT VALUE
00956A F4F5 7D 8156  A         TST     DISFLG    HORIZONTAL OR VERTICAL PRINT?
00957A F4F8 27 25 F51F         BEQ     NXTOPT    HORIZONTAL
00958A F4FA CE F7E3  A         LDX     #OPTMSG   PRINT 'REG. = '
00959A F4FD BD F198  A         JSR     MESAGE
00960A F500 FE 8111  A         LDX     XTMP3     PRINT REG. ADDR.
00961A F503 BD F140  A         JSR     OUT4HS
00962A F506 BD F1D9  A         JSR     INADDR    GET ANY CHANGES
00963A F509 7D 8134  A         TST     ROLFLG
00964A F50C 27 0D F51B         BEQ     OPNXT     NO CHANGE
00965A F50E FE 8111  A         LDX     XTMP3
00966A F511 B6 8118  A         LDAA    HEXBUF    GET MSB OF CHANGE
00967A F514 A7 00    A         STAA    0,X
00968A F516 B6 8119  A         LDAA    HEXBUF+1  GET LSB
00969A F519 A7 01    A         STAA    1,X
00970A F51B 8D B1 F4CE OPNXT   BSR     TSTCR     CHECK FOR END OF CHANGES
00971A F51D 8D 0B F52A         BSR     CRLF2
00972A F51F FE 8111  A NXTOPT  LDX     XTMP3
00973A F522 08                 INX
00974A F523 08                 INX               NEXT REG.
00975A F524 8C 8170  A         CPX     #ENDOPT   END OF TABLE
00976A F527 26 C2 F4EB         BNE     OPLOOP    NO, CONTINUE
00977A F529 39                 RTS
00978A F52A 7E F619  A CRLF2   JMP     CRLF      START NEW LINE
00979A F52D 20 A9 F4D8 OUTS2   BRA     OUTS1
```

```
00981                    *
00982                    * DISPLAY THE PAGE REGISTER
00983                    *
00984A F52F 8D 49 F57A PAGE   BSR    GETPG
00985A F531 37               PSHB
00986A F532 36               PSHA
00987A F533 7D 8156  A       TST    DISFLG   HORIZONTAL OR VERTICAL DISPLAY
00988A F536 27 06 F53E       BEQ    HORIZN   HORIZONTAL
00989A F538 CE F7D5  A       LDX    #RAMMSG  PRINT RAM PAGE =
00990A F53B BD F198  A       JSR    MESAGE
00991A F53E 8D ED F52D HORIZN BSR   OUTS2
00992A F540 32               PULA
00993A F541 BD F12A  A       JSR    OUTA     PRINT RAM PAGE
00994A F544 8D E7 F52D       BSR    OUTS2
00995A F546 8D E5 F52D       BSR    OUTS2
00996A F548 7D 8156  A       TST    DISFLG
00997A F54B 27 13 F560       BEQ    ROM
00998A F54D 8D 4C F59B       BSR    TSTBAD   TAKE ONLY LEGAL PAGE NUMBERS (0-7)
00999A F54F 27 02 F553       BEQ    NXTRM
01000A F551 8D 3F F592       BSR    CNGRAM   CHANGE THE RAM PAGE REG.
01001A F553 33           NXTRM PULB          PREPARE FOR A RETURN
01002A F554 BD F4CE  A       JSR    TSTCR
01003A F557 37               PSHB           NO RETURN OCCURRED
01004A F558 8D D0 F52A       BSR    CRLF2
01005A F55A CE F7DC  A       LDX    #ROMMSG
01006A F55D BD F198  A       JSR    MESAGE
01007A F560 8D CB F52D ROM    BSR    OUTS2
01008A F562 33               PULB
01009A F563 BD F12D  A       JSR    OUTB     PRINT ROM PAGE
01010A F566 8D C5 F52D       BSR    OUTS2
01011A F568 7D 8156  A       TST    DISFLG
01012A F56B 27 18 F585       BEQ    ENDPAG
01013A F56D 8D BE F52D       BSR    OUTS2
01014A F56F 8D 2A F59B       BSR    TSTBAD
01015A F571 27 02 F575       BEQ    NXTR
01016A F573 8D 11 F586       BSR    CNGROM   CHANGE ROM PAGE
01017A F575 BD F4CE  A NXTR   JSR    TSTCR
01018A F578 20 B0 F52A       BRA    CRLF2
```

```
01020                          *
01021                          * SUBROUTINE TO GET AND SEPARATE THE RAM AND ROM PAGES
01022                          *
01023A F57A B6 8083  A GETPG   LDAA   PAGERG   GET PAGE REG.
01024A F57D 16                 TAB
01025A F57E C4 07    A         ANDB   #7       SEPARATE THE ROM PAGE
01026A F580 44                 LSRA
01027A F581 44                 LSRA
01028A F582 44                 LSRA
01029A F583 84 07    A         ANDA   #7       SEPARATE THE RAM PAGE
01030A F585 39         ENDPAG  RTS
01031                          *
01032                          * SUBROUTINE TO CHANGE THE ROM PAGE REG.
01033                          *
01034A F586 84 07    A CNGROM  ANDA   #7       CLR ALL BUT LOWER 3 BITS
01035A F588 C6 38    A         LDAB   #$38     PREPARE TO CHANGE THE PAGE REG.
01036                          *
01037                          * SUBROUTINE TO CHANGE PAGE REG.
01038                          *
01039A F58A F4 8083  A CHNGPG  ANDB   PAGERG   CLR ROM PAGE
01040A F58D 1B                 ABA             ADD TOGETHER
01041A F58E B7 8083  A         STAA   PAGERG   UPDATE PAGEREG.
01042A F591 39                 RTS             FINISHED
01043                          *
01044                          * SUBROUTINE TO CHANGE THE RAM PAGE
01045                          *
01046A F592 84 07    A CNGRAM  ANDA   #7       CLR ALL BUT LOWER 3 BITS
01047A F594 48                 ASLA
01048A F595 48                 ASLA
01049A F596 48                 ASLA            CORRECT POSITION IN THE PAGE REG.
01050A F597 C6 07    A         LDAB   #7       DO NOT CHANGE THE ROM PAGE
01051A F599 20 EF F58A         BRA    CHNGPG   GO CHANGE THE PAGE REGISTER
01052                          *
01053                          * SUBROUTINE TO CHECK PAGE ENTRY
01054                          *
01055A F59B BD F1D2  A TSTBAD  JSR    INBYTE
01056A F59E 7D 8134  A         TST    ROLFLG
01057A F5A1 27 1E F5C1         BEQ    RETBAD
01058A F5A3 4D                 TSTA
01059A F5A4 2B 04 F5AA         BMI    BAD
01060A F5A6 81 08    A         CMPA   #8
01061A F5A8 2D 17 F5C1         BLT    RETBAD
01062A F5AA 86 3F    A BAD     LDAA   #'?
01063A F5AC BD F156  A         JSR    OUTCHR
01064A F5AF 20 EA F59B         BRA    TSTBAD
```

```
01066                          *
01067                          * SUBROUTINE TO INC OR DEC TO THE NEXT RAM PAGE
01068                          * Z=0 BEYOND THE PAGE LIMITS
01069                          *
01070A F5B1 8D C7 F57A INCRAM  BSR    GETPG     WHATS IN THE RAM AND ROM PAGE
01071A F5B3 4C                 INCA              INC THE RAM PAGE
01072A F5B4 81 07     A        CMPA   #7        IS IT AN ILLEGAL PAGE #
01073A F5B6 2E 09 F5C1         BGT    BADRAM    YES, THE Z-FLAG = 0
01074A F5B8 20 05 F5BF         BRA    CNGRM1
01075A F5BA 8D BE F57A DECRAM  BSR    GETPG
01076A F5BC 4A                 DECA
01077A F5BD 2B 02 F5C1         BMI    BADRAM
01078A F5BF 8D D1 F592 CNGRM1  BSR    CNGRAM    GO CHANGE THE RAM PAGE
01079          F5C1   A RETBAD EQU    *
01080A F5C1 39                 BADRAM RTS               Z=0 IF BAD PAGE #
01081                          *
01082                          * SUBROUTINE TO INC OR DEC THE ROM PAGE
01083                          * THE Z-FLAG = 0 IF YOU GO BEYOND THE PAGE LIMITS
01084                          *
01085A F5C2 8D B6 F57A INCROM  BSR    GETPG
01086A F5C4 5C                 INCB
01087A F5C5 C1 07     A        CMPB   #7
01088A F5C7 2E 0B F5D4         BGT    BADROM
01089A F5C9 20 05 F5D0         BRA    CNGRM2
01090A F5CB 8D AD F57A DECROM  BSR    GETPG
01091A F5CD 5A                 DECB
01092A F5CE 2B 04 F5D4         BMI    BADROM    PAGE LESS THAN ZERO
01093A F5D0 17                 CNGRM2 TBA               PLACE NEW ROM PAGE IN 'A
01094A F5D1 8D B3 F586         BSR    CNGROM    CHANGE THE ROM PAGE
01095A F5D3 5F                 CLRB             SET Z=1
01096A F5D4 39                 BADROM RTS
```

```
01098                          ****************************************
01099                          *
01100                          * BREAKPOINTS
01101                          *
01102                          ****************************************
01103                          *
01104A F5D5 BD F1C8  A SETBRK  JSR     INADD1
01105A F5D8 BD FD45  A         JSR     FSET34    D3BUG ROUTINE TO ADD A BREAKPOINT
01106A F5DB 20 06 F5E3         BRA     PNTBRK    DISPLAY BREAKPOINTS
01107A F5DD BD F1C8  A RSTBRK  JSR     INADD1
01108A F5E0 BD FDB1  A         JSR     FCLR34    ROUTINE TO DELETE A BREAKPOINT
01109A F5E3 8D 34 F619 PNTBRK  BSR     CRLF      NEW LINE
01110A F5E5 F6 8142  A         LDAB    NUMBER    GET NUMBER OF BREAKPOINTS
01111A F5E8 27 11 F5FB         BEQ     ENDDSP    NONE ARE PRESENT RETURN
01112A F5EA CE 8143  A         LDX     #BOF      LOAD 1ST ADDR.
01113A F5ED 37               BRKLOP PSHB
01114A F5EE BD F140  A         JSR     OUT4HS    PRINT ADDR.
01115A F5F1 BD F149  A         JSR     OUTS
01116A F5F4 BD F149  A         JSR     OUTS
01117A F5F7 33                 PULB
01118A F5F8 5A                 DECB
01119A F5F9 26 F2 F5ED         BNE     BRKLOP    NEXT ADDR.
01120A F5FB 39               ENDDSP RTS               ******RETURN*****
01121           8143  A BOF    EQU     $8143
01122           FDB1  A FCLR34 EQU     $FDB1
01123           FD45  A FSET34 EQU     $FD45
01124           8142  A NUMBER EQU     $8142
01125           FCFE  A ACTION EQU     $FCFE
01126           FCE5  A CLR    EQU     $FCE5
01127                          *
01128                          ****************************************
01129                          *
01130                          * USER PROGRAMS
01131                          *
01132                          ****************************************
01133           F5FC  A USEPG1 EQU     *
01134A F5FC FE 816A  A         LDX     BEGOPT+2 USES THE 2ND OPTIONAL REG.
01135A F5FF 6E 00    A         JMP     0,X
01136           F601  A USEPG2 EQU     *
01137A F601 FE 816C  A         LDX     BEGOPT+4 USES THE 3RD OPTIONAL REG.
01138A F604 6E 00    A         JMP     0,X
01139           F606  A USEPG3 EQU     *
01140A F606 FE 816E  A         LDX     BEGOPT+6 USES THE 4TH OPTIONAL REG.
01141A F609 6E 00    A         JMP     0,X
```

```
01143                          ********** CR **********
01144                          * CARRIAGE RETURN
01145                          *
01146A F60B FF 810D  A RTURN1  STX    CTEMP    SAVE REGS.
01147A F60E FE 8166  A         LDX    LINEAD   GET START ADR THIS LINE
01148A F611 FF 8164  A         STX    CURPOS   SAVE IT
01149A F614 7F 815A  A         CLR    CCOUNT   CLEAR CHAR/LINE COUNTER
01150A F617 20 69 F682         BRA    UPDATE   UPDATE CRTC

01152                          ********** CR/LF **********
01153          F619  A PCRLF   EQU    *        CR/LF COME HERE
01154A F619 7D 810A  A CRLF    TST    TERM     LOOK FOR TERMINAL MODE
01155A F61C 27 09 F627         BEQ    CRLFR2   DO A R2 CRLF
01156A F61E 86 0A    A         LDAA   #$A      LINE FEED
01157A F620 8D 02 F624         BSR    OUTC1
01158A F622 86 0D    A         LDAA   #$D      CARRIAGE RETURN
01159A F624 7E F156  A OUTC1   JMP    OUTCHR
01160A F627 8D E2 F60B CRLFR2  BSR    RTURN1   DO CR
01161                          * SUBR TO DO LINE FEED
01162A F629 FF 810D  A LFEED   STX    CTEMP    SAVE REGS.
01163A F62C CE 8162  A         LDX    #STARAD  POINT TO DISPLAY VARIABLE
01164A F62F A6 03    A         LDAA   3,X      GET CURPOS LSB
01165A F631 BB 8160  A         ADDA   CLINE    ADD CHAR.LINE
01166A F634 A7 03    A         STAA   3,X      SAVE IT
01167A F636 24 02 F63A         BCC    LFEED1   CHECK CARRY
01168A F638 6C 02    A         INC    2,X      ADD CARRY TO MSB


01170                          * ROUTINE TO ADVANCE LINE START ADR
01171          F63A  A LFEED1  EQU    *
01172A F63A A6 05    A         LDAA   5,X      GET LINEAD LSB
01173A F63C BB 8160  A         ADDA   CLINE    ADD CHAR/LINE
01174A F63F A7 05    A         STAA   5,X      SAVE IT
01175A F641 B7 810C  A         STAA   CUTEMP+1 SAVE IT IN TEMP
01176A F644 A6 04    A         LDAA   4,X      GET LINEAD MSB
01177A F646 89 00    A         ADCA   #$00     ADD CARRY
01178A F648 A7 04    A         STAA   4,X      UPDATE LINEAD WITHOUT MASKING OFF
01179                          * ROUTINE TO CLEAR NEXT LINE
01180A F64A 84 0F    A         ANDA   #$0F     MASK TO 4K
01181A F64C 8A 90    A         ORAA   #$90
01182A F64E B7 810B  A         STAA   CUTEMP   SAVE IT IN TEMP
01183A F651 FE 810B  A LFEED3  LDX    CUTEMP   GET START ADR NEXT LINE
01184A F654 86 20    A         LDAA   #$20     GET A SPACE
01185A F656 F6 8160  A         LDAB   CLINE    GET CHAR.LINE
01186A F659 A7 00    A LFEED4  STAA   0,X      STORE A SPACE
01187A F65B 08                 INX             NEXT LOCATION
01188A F65C 8C A000  A         CPX    #$A000   LAST DISPLAY LOCATION?
01189A F65F 26 03 F664         BNE    LFEED5   NO,CONTINUE
01190A F661 CE 9000  A         LDX    #$9000   REPOINT TO BEGINNING
01191          F664  A LFEED5  EQU    *
01192A F664 5A                 DECB            COUNT LOCATIONS
01193A F665 26 F2 F659         BNE    LFEED4   LAST LOCATIONS
```

```
01195                          * ROUTINE TO CHECK FOR LAST LINE ON SCREEN
01196A F667 CE 815C  A         LDX    #LCOUNT  POINT TO DISPLAY VARIABLE
01197A F66A A6 00    A         LDAA   0,X      GET LINES THIS SCREEN
01198A F66C 6C 00    A         INC    0,X
01199A F66E A1 02    A         CMPA   2,X      LAST LINE?
01200A F670 26 10 F682         BNE    UPDATE
01201                          *
01202                          * SUBROUTINE TO SCROLL UP
01203                          * A & X REGS. ARE DESTROYED
01204                          *
01205A F672 CE 815C  A SCROLU  LDX    #LCOUNT  POINT TO DISPLAY VARIABLE
01206A F675 6A 00    A         DEC    0,X
01207A F677 A6 07    A         LDAA   7,X      GET STARTING ADDR. LSB
01208A F679 BB 8160  A         ADDA   CLINE    ADD CHAR/LINE
01209A F67C A7 07    A         STAA   7,X
01210A F67E 24 02 F682         BCC    UPDATE
01211A F680 6C 06    A         INC    6,X
01212                          ********** UPDATE **********
01213                          * UPDATE CRTC
01214                          *
01215                          *
01216                          ************************************


01218          F682  A UPDATE  EQU    *
01219A F682 C6 0C    A         LDAB   #$0C     POINT TO CRTC START ADR
01220A F684 CE 8162  A         LDX    #STARAD  POINT TO DISPLAY VAR
01221          F687  A UDATE1  EQU    *
01222A F687 A6 00    A         LDAA   0,X      GET DATA
01223A F689 F7 8042  A         STAB   CRTCAR   SELECT REGISTER
01224A F68C B7 8043  A         STAA   CRTCDR   PROGRAM VALUE
01225A F68F 08                 INX            NEXT VALUE
01226A F690 5C                 INCB           NEXT REGISTER
01227A F691 C1 10    A         CMPB   #$10     LAST REG?
01228A F693 26 F2 F687         BNE    UDATE1   NEXT VALUES
01229A F695 FE 810D  A         LDX    CTEMP    RESTORE THE X REG.
01230A F698 39                 RTS
```

```
01232                    ************************************************************
01233                    *
01234                    *      SUBROUTINE TO CLEAR DISPLAY RAM
01235                    *
01236                    ************************************************************


01238           F699   A CLEAR  EQU     *
01239A F699 CE 9000   A         LDX     #MEMSTR  GET BEG. ADDR OF DISPLAY MEMORY
01240A F69C 86 20     A         LDAA    #$20     ASCII SPACE
01241           F69E   A CLEAR1 EQU     *
01242A F69E A7 00     A         STAA    0,X      STORE SPACE IN RAM
01243A F6A0 08                  INX
01244A F6A1 8C A000   A         CPX     #MEMEND  LAST LOCATION?
01245A F6A4 26 F8 F69E          BNE     CLEAR1   CONTINUE
01246A F6A6 39                  RTS


01248                    ************************************************************
01249                    *
01250                    *          SUBROUTINE TO CLEAR DISPLAY SCREEN
01251                    *
01252                    *
01253                    ************************************************************
01254                    *
01255                    *
01256           F6A7   A CLRSCR EQU     *
01257A F6A7 8D F0 F699          BSR     CLEAR    CLEAR DISPLAY RAM


01259                    ************************************************************
01260                    *             CRTC INITIALIZATION ROUTINE
01261                    *
01262                    ************************************************************


01264           F6A9   A CRTIZ  EQU     *
01265A F6A9 5F                  CLRB
01266A F6AA FE 8115   A         LDX     FORMAT   GET CRT INIT TABLE ADDRESS
01267A F6AD FF 810D   A         STX     CTEMP    SAVE THE XREG.
01268A F6B0 8D D5 F687          BSR     UDATE1
01269A F6B2 A6 10     A         LDAA    $10,X    LOAD CONTENTS OF POINTER
01270                    *         TO GET CHAR/LINE INFO
01271A F6B4 B7 8160   A         STAA    CLINE    SET CLINE LSCREN ACCORDINGLY
01272A F6B7 A6 11     A         LDAA    $11,X
01273A F6B9 B7 815E   A         STAA    LSCREN
01274A F6BC CE 9000   A         LDX     #MEMSTR  INITIAL CURSOR POS
01275A F6BF FF 8164   A         STX     CURPOS
01276A F6C2 FF 8162   A         STX     STARAD   INITIAL START ADDR
01277A F6C5 FF 8166   A         STX     LINEAD   STARTING ADR THIS LINE
01278A F6C8 7F 815C   A         CLR     LCOUNT
01279A F6CB 7F 815A   A         CLR     CCOUNT   CLEAR CCOUNT & LCOUNT
01280A F6CE 39                  RTS
```

```
01282                    *
01283                    * JUMP TABLE TO ROUTINES PERFORMING CRTBUG FCTN'S
01284                    *
01285           F6CF   A FCTABL EQU    *
01286A F6CF     42     A        FCC    /B/       "B" - PRINT ALL BREAKS
01287A F6D0     F5E3   A        FDB    PNTBRK
01288A F6D2     43     A        FCC    /C/       "C" - CONTINUE
01289A F6D3     F444   A        FDB    CONT
01290A F6D5     44     A        FCC    /D/       "D" - DELETE ALL BREAKS
01291A F6D6     FCE5   A        FDB    CLR
01292A F6D8     45     A        FCC    /E/       'E' - EXAMINE MEMORY BLOCK
01293A F6D9     F3CE   A        FDB    MEMBLK
01294A F6DB     47     A        FCC    /G/       "G" - GO TO ENTERED ADDRESS
01295A F6DC     F430   A        FDB    GOTO
01296A F6DE     4C     A        FCC    /L/       "L" - LOAD
01297A F6DF     F240   A        FDB    LOAD
01298A F6E1     4D     A        FCC    /M/       "M" - MEMORY CHANGE
01299A F6E2     F310   A        FDB    CHANGE
01300A F6E4     4E     A        FCC    /N/       "N" - NEXT (TRACE 1 INSTR)
01301A F6E5     F3AB   A        FDB    TRACE1
01302A F6E7     4F     A        FCC    /O/       "O"- OFFSET LOAD
01303A F6E8     F292   A        FDB    OFLOAD
01304A F6EA     50     A        FCC    /P/       "P" - PUNCH
01305A F6EB     F285   A        FDB    PUNCH
01306A F6ED     52     A        FCC    /R/       "R" - PRINT STACK
01307A F6EE     F471   A        FDB    PSTAK1
01308A F6F0     54     A        FCC    /T/       "T" - TRACE N INSTRUCTIONS
01309A F6F1     F3A3   A        FDB    TRACE
01310A F6F3     55     A        FCC    /U/       "U" - RESET A BREAKPOINT
01311A F6F4     F5DD   A        FDB    RSTBRK
01312A F6F6     56     A        FCC    /V/       "V" - VERIFY A TAPE
01313A F6F7     F243   A        FDB    VERFY
01314A F6F9     53     A        FCC    /S/       "S" - SET A BREAKPOINT
01315A F6FA     F5D5   A        FDB    SETBRK
01316A F6FC     21     A        FCC    /!/       "SHIFT 1" EXECUTES PROGRAM AT ADDR
01317                      *                      IN USER1 ADDR VECTOR $816A
01318A F6FD     F5FC   A        FDB    USEPG1
01319A F6FF     22     A        FCC    /"/       "SHIFT 2" EXECUTES PROGRAM AT ADDR
01320                      *                      IN USER2 ADDR VECTOR $816C
01321A F700     F601   A        FDB    USEPG2
01322A F702     23     A        FCC    /#/       "SHIFT 3" EXECUTES PROGRAM AT ADDR
01323                      *                      IN USER3 ADDR VECTOR $816E
01324A F703     F606   A        FDB    USEPG3
01325A F705     05     A        FCB    5         CONTROL 'E' ERASE SCREEN
01326A F706     F6A7   A        FDB    CLRSCR
01327           F708   A FCTBEN EQU    *
```

```
01329                    ************************************************************
01330                    *
01331                    *   CRT INITIALIZATION FOR 32 X 16 SCREEN SIZE
01332                    *
01333                    *                FOR U S TELEVISION
01334                    *
01335                    ************************************************************
01336                    *
01337           F708   A CRTAB1 EQU    *
01338A F708    31     A        FCB    49,32,39,06 HORIZ. ,DISP.,SYNC POS,SYNC WID
01339A F70C    15     A        FCB    21,08,16,18 VERT. ,ADJUST ,DISPLAYED,SYNC P
01340A F710    00     A        FCB    00,11    INTERLACE, MAX SCAN LINE ADDR
01341A F712    00     A        FCB    00,09    CURSOR SIZE & BLINK
01342A F714    90     A        FCB    144,0    START ADDRESS
01343A F716    90     A        FCB    144,0    CURSOR ADDRESS
01344A F718    20     A        FCB    32,15    CHAR,LINES-1


01346                    ************************************************************
01347                    *
01348                    *    CRT INITIALIZATION TABLE FOR 64 X 16 SCREEN SIZE
01349                    *
01350                    *                FOR U S TELEVISION
01351                    *
01352                    ************************************************************
01353           F71A   A CRTAB2 EQU    *
01354A F71A    63     A        FCB    99,64,79,06 HORIZ,DISP,SYNC POS,SYNC WIDTH
01355A F71E    14     A        FCB    20,09,16,18 VERT.,ADJUST,DISPLAYED,SYNC POS
01356A F722    00     A        FCB    0,11     INTERLACE,MAX SCAN LINE ADDR
01357A F724    00     A        FCB    00,09    CURSOR SIZE & BLINK
01358A F726    90     A        FCB    144,00   START ADDRESS
01359A F728    90     A        FCB    144,00   CURSOR ADDRESS
01360A F72A    40     A        FCB    64,15    CHAR,LINES-1
```

```
01362                          ************************************************************
01363                          *
01364                          *   CRT INITIALIZATION FOR 80 X 20 SCREEN SIZE
01365                          *
01366                          *                FOR U S TELEVISION
01367                          *
01368                          ************************************************************
01369           F72C   A CRTAB3 EQU     *
01370A F72C     63     A        FCB     99,80,86,06 HORIZ,DISP,SYNC POS,SYNC WIDTH
01371A F730     14     A        FCB     20,09,20,20 VERT, ADJUST,DISPLAYED,SYNC POS
01372A F734     00     A        FCB     00,11     INTERLACE,MAX SCAN LINE ADDR
01373A F736     00     A        FCB     00,09     CURSOR SIZE & BLINK
01374A F738     90     A        FCB     144,00    START ADDRESS
01375A F73A     90     A        FCB     144,00    CURSOR ADDRESS
01376A F73C     50     A        FCB     80,19     CHAR,LINES-1


01378                          ************************************************************
01379                          *
01380                          *   CRT INITIALIZATION FOR 32 X 16 SCREEN SIZE
01381                          *
01382                          *                FOR EUROPEAN TELEVISION
01383                          *
01384                          ************************************************************
01385           F73E   A CRTAB4 EQU     *
01386A F73E     31     A        FCB     49,32,38,06 HORIZ,DISP,SYNC POS,SYNC WIDTH
01387A F742     19     A        FCB     25,01,16,20 VERT, ADJUST,DISPLAYED,SYNC POS
01388A F746     00     A        FCB     00,11     INTERLACE,MAX SCAN ADDR
01389A F748     00     A        FCB     00,09     CURSOR SIZE,BLINK
01390A F74A     90     A        FCB     144,00    START ADDRESS
01391A F74C     90     A        FCB     144,00    CURSOR ADDRESS
01392A F74E     20     A        FCB     32,15     CHAR,LINES-1


01394                          ************************************************************
01395                          *
01396                          *   CRT INITIALIZATION FOR 64 X 16 SCREEN SIZE
01397                          *
01398                          *                FOR EUROPEAN TELEVISION
01399                          *
01400                          ************************************************************
01401           F750   A CRTAB5 EQU     *
01402A F750     63     A        FCB     99,64,76,06 HORIZ,DISP,SYNC POS,SYNC WIDTH
01403A F754     19     A        FCB     25,01,16,20 VERT, ADJUST,DISPLAYED,SYNC POS
01404A F758     00     A        FCB     00,11     INTERLACE,MAX SCAN LINE ADDR
01405A F75A     00     A        FCB     00,09     CURSOR SIZE,BLINK
01406A F75C     90     A        FCB     144,00    START ADDRESS
01407A F75E     90     A        FCB     144,00    CURSOR ADDRESS
01408A F760     40     A        FCB     64,15     CHAR,LINES-1
```

```
01410                    ************************************************************
01411                    *
01412                    *   CRT INITIALIZATION FOR 80 X 20 SCREEN SIZE
01413                    *
01414                    *                FOR EUROPEAN TELEVISION
01415                    *
01416                    ************************************************************
01417                    *
01418                    *
01419           F762   A CRTAB6 EQU     *
01420A F762     63     A        FCB     99,80,86,06 HORIZ,DISP,SYNC POS,SYNC WIDTH
01421A F766     19     A        FCB     25,01,20,20 VERT, ADJUST,DISPLAYED,SYNC POS
01422A F76A     00     A        FCB     00,11     INTERLACE,MAX SCAN LINE ADDR
01423A F76C     00     A        FCB     00,09     CURSOR SIZE,BLINK
01424A F76E     90     A        FCB     144,00    START ADDRESS
01425A F770     90     A        FCB     144,00    CURSOR ADDRESS
01426A F772     50     A        FCB     80,19     CHAR,LINES-1


01428                    ************************************************************
01429                    *
01430                    *         CRT ADDRESS TABLE
01431                    *
01432                    ************************************************************



01434           F774   A CRTADD EQU     *
01435A F774     F708   A        FDB     CRTAB1    U.S. 32 X 16
01436A F776     F73E   A        FDB     CRTAB4    EUROPE
01437A F778     F71A   A        FDB     CRTAB2    U.S. 64 X 16
01438A F77A     F750   A        FDB     CRTAB5    EUROPE
01439A F77C     F72C   A        FDB     CRTAB3    U.S. 80 X 20
01440A F77E     F762   A        FDB     CRTAB6    EUROPE
```

```
01442                    * CONSTANT DATA
01443                    *
01444A F780     44     A MCL2   FCC    /D3BUG2 1.0/
01445A F78A     04     A        FCB    4
01446A F78B     43     A MCL3   FCC    /CC B  A   X    PC   S   RAM ROM/
01447A F7AA     04     A        FCB    4
01448A F7AB     20     A MCL35  FCC    / REG1  REG2  REG3  REG4/
01449A F7C2     04     A        FCB    4
01450A F7C3     42     A MCL4   FCC    /BEG ADR=/
01451A F7CB     04     A        FCB    4
01452A F7CC     45     A MCL5   FCC    /END ADR=/
01453A F7D4     04     A        FCB    4
01454A F7D5     52     A RAMMSG FCC    /RAMPG=/
01455A F7DB     04     A        FCB    4
01456A F7DC     52     A ROMMSG FCC    /ROMPG=/
01457A F7E2     04     A        FCB    4
01458A F7E3     52     A OPTMSG FCC    /REG=/
01459A F7E7     04     A        FCB    4
01460A F7E8     43     A ERRMSG FCC    /Cksum?/
01461A F7EE     04     A        FCB    4
01462A F7EF     33     A PLMSG  FCC    /3or12?/
01463A F7F5     04     A        FCB    4
01464A F7F6     54     A ADMESG FCC    /True Adr=/
01465A F7FF     04     A        FCB    4
```

```
01467                                   END
TOTAL ERRORS 00000


800B ACIAD  00090*
800A ACIAS  00089*00228 00607
8009 ACIATD 00088*00296 00395
8008 ACIATS 00087*00229 00231 00293 00391 00715
FCFE ACTION 01125*
F1B5 ADDRES 00445*00591
F397 ADDRIN 00725 00736*
F94D ADDXA  00101*00214
F7F6 ADMESG 00663 01464*
F11F ALPHA  00332 00335*
F200 ALTIN  00487*
F282 ARBAUD 00576 00579*
F1F4 ARNCOM 00477 00481*
F2C0 ARNERR 00620 00622*
F15B ARNOUT 00391*00394
F0FE ASCHEX 00310*00485
F5AA BAD    01059 01062*
F39C BADBYT 00733 00738*
F5C1 BADRAM 01073 01077 01080*
F5D4 BADROM 01088 01092 01096*
F26A BAUD   00547 00567*00588 00601
813A BEGA   00102*00443 00655 00779 00791 00811
F1A6 BEGADD 00439*00445 00604 00776
F245 BEGLD  00544 00546*
8168 BEGOPT 00105*00951 01134 01137 01140
F2D7 BLKRD  00628 00633*
8143 BOF    01112 01121*
F5ED BRKLOP 01113*01119
F380 BYTEIN 00726*
815A CCOUNT 00107*00414 00415 01149 01279
F0DC CH1    00278 00281*
F310 CHANGE 00673*01299
F58A CHNGPG 01039*01051
F473 CHNGRG 00881*
F374 CKTERM 00710 00715*
F699 CLEAR  00191 01238*01257
F69E CLEAR1 01241*01245
F1DA CLEARF 00469 00471*
8160 CLINE  00110*00416 00804 00865 00947 01165 01173 01185 01208 01271
FCE5 CLR    00187 01126*01291
F6A7 CLRSCR 01256*01326
F592 CNGRAM 01000 01046*01078
F5BF CNGRM1 01074 01078*
F5D0 CNGRM2 01089 01093*
F586 CNGROM 01016 01034*01094
F1E9 COMLOP 00476*00484
F223 COMMD  00475 00507*
F444 CONT   00849*01289
F3CB CONTL  00713 00717 00763*00836 00841
F4E8 CONTOP 00946 00949 00951*
F0A5 CONTRL 00157 00242*00270 00283 00465 00763
8154 CRFLAG 00120*00473 00492 00575 00675 00933
```

```
F619 CRLF   00233 00255 00444 00696 00792 00978 01109 01154*
F4CC CRLF1  00862 00887 00899 00925 00929*
F52A CRLF2  00929 00971 00978*01004 01018
F1B2 CRLFD  00419 00439 00444*
F627 CRLFR2 01155 01160*
F708 CRTAB1 01337*01435
F71A CRTAB2 01353*01437
F72C CRTAB3 01369*01439
F73E CRTAB4 01385*01436
F750 CRTAB5 01401*01438
F762 CRTAB6 01419*01440
F774 CRTADD 00213 01434*
8042 CRTCAR 00091*01223
8043 CRTCDR 00092*01224
F050 CRTINT 00190*
F6A9 CRTIZ  00217 01264*
810D CTEMP  00116*00402 00418 01146 01162 01229 01267
F0C1 CTRL1  00259*
8164 CURPOS 00112*00403 00407 00411 00413 01148 01275
810B CUTEMP 00115*00406 00408 00409 01175 01182 01183
F410 DEC2   00808 00810*
F5BA DECRAM 01075*
F5CB DECROM 01090*
8156 DISFLG 00121*00870 00881 00890 00909 00945 00956 00987 00996 01011
F49B DOUBLE 00872 00907*
F3E9 EAOUT  00788*00790
F3DF EHOUT  00782*00786
813C ENDA   00103*00449 00648 00652 00665
F207 ENDAD  00480 00490*
F1B7 ENDADD 00446*
F423 ENDAS  00818 00820 00823*
F112 ENDASC 00311 00315 00317 00321*00330
F35F ENDCHG 00695 00704*
F22F ENDCOM 00483 00520*
F4D7 ENDCR  00934 00939*
F5FB ENDDSP 01111 01120*
F0FB ENDINC 00297 00302*
F25E ENDLD  00553 00556*
F261 ENDLD1 00555 00557*
F20E ENDLF  00491 00493*
F1A4 ENDMES 00430 00434*
8170 ENDOPT 00106*00975
F425 ENDOUT 00822 00824*
F585 ENDPAG 01012 01030*
F035 ENDTER 00169 00175 00179*
F8AF ENNMI  00122*00186
F088 ENT    00223*
F037 ENT1   00156 00181*
F09C ENT2   00232*
F7E8 ERRMSG 00554 01460*
8133 ERROR  00552 00562*
F405 ESETUP 00781 00787 00804*
F393 EXARN  00728 00730 00734*00737
8130 EXFLG  00123*00254 00758
F37B EXGET  00158 00723*00740
F363 EXKEY  00160 00706*
FDB1 FCLR34 01108 01122*
F6CF FCTABL 00261 01285*
```

```
F708 FCTBEN 00268 01327*
8135 FLG24  00124*00471 00495 00527
8136 FNCFLG 00842 00847*
8115 FORMAT 00125*00216 01266
FD45 FSET34 01105 01123*
F57A GETPG  00984 01023*01070 01075 01085 01090
FC64 GO     00846 00848*
FEC0 GO1    00551 00563*
F0D4 GOODCH 00264 00276*
F430 GOTO   00838*01295
F436 GOTO1  00755 00842*
F114 HEXASC 00329*00358
F144 HEXBF2 00373*00698
8118 HEXBUF 00126*00370 00373 00497 00500 00748 00918 00921 00966 00968
F53E HORIZN 00988 00991*
F13D HXBUF0 00370*00697
F1C8 INADD1 00442 00448 00462*00839 01104 01107
F1D9 INADDR 00462 00470*00677 00736 00912 00962
F1C4 INBYT1 00460*00747
F1D2 INBYTE 00460 00467*00571 00726 00892 01055
F0E7 INCHLP 00293*00295
F0E2 INCHR  00260 00291*00300 00474 00796
F0F2 INCHR1 00292 00298*
F5B1 INCRAM 01070*
F5C2 INCROM 01085*
8132 INIT   00127*00674 00682
8137 INIT2  00128*00249 00646 00650 00653 00676 00683 00724
8005 IOPIAC 00086*00165 00172
8004 IOPIAD 00085*00167 00168 00170 00174
8117 KEY    00129*00479 00689 00734
815C LCOUNT 00108*01196 01205 01278
F259 LDERR  00554*00621 00625
8138 LDFLAG 00606 00608*
8139 LDTBLE 00643 00658*
F629 LFEED  01162*
F63A LFEED1 01167 01171*
F651 LFEED3 01183*
F659 LFEED4 01186*01193
F664 LFEED5 01189 01191*
8166 LINEAD 00113*01147 01277
F240 LOAD   00542*01297
F2AC LOOP12 00611*00613
F47D LOOP1B 00889*00901
F49E LOOP2B 00908*00927
F1E3 LOOPAD 00474*00486 00489
F199 LOOPME 00428*00433
F2B2 LOP12A 00614*00624
815E LSCREN 00109*00777 01273
F24F LV1200 00548 00550*
F780 MCL2   00234 01444*
F78B MCL3   00863 01446*
F7AB MCL35  00868 01448*
F7C3 MCL4   00440 00556 01450*
F7CC MCL5   00446 01452*
F337 MEM    00681 00688*00700 00703
F32C MEM1   00684*00693
F3CE MEMBLK 00775*01293
FADC MEMCH  00688 00742*
```

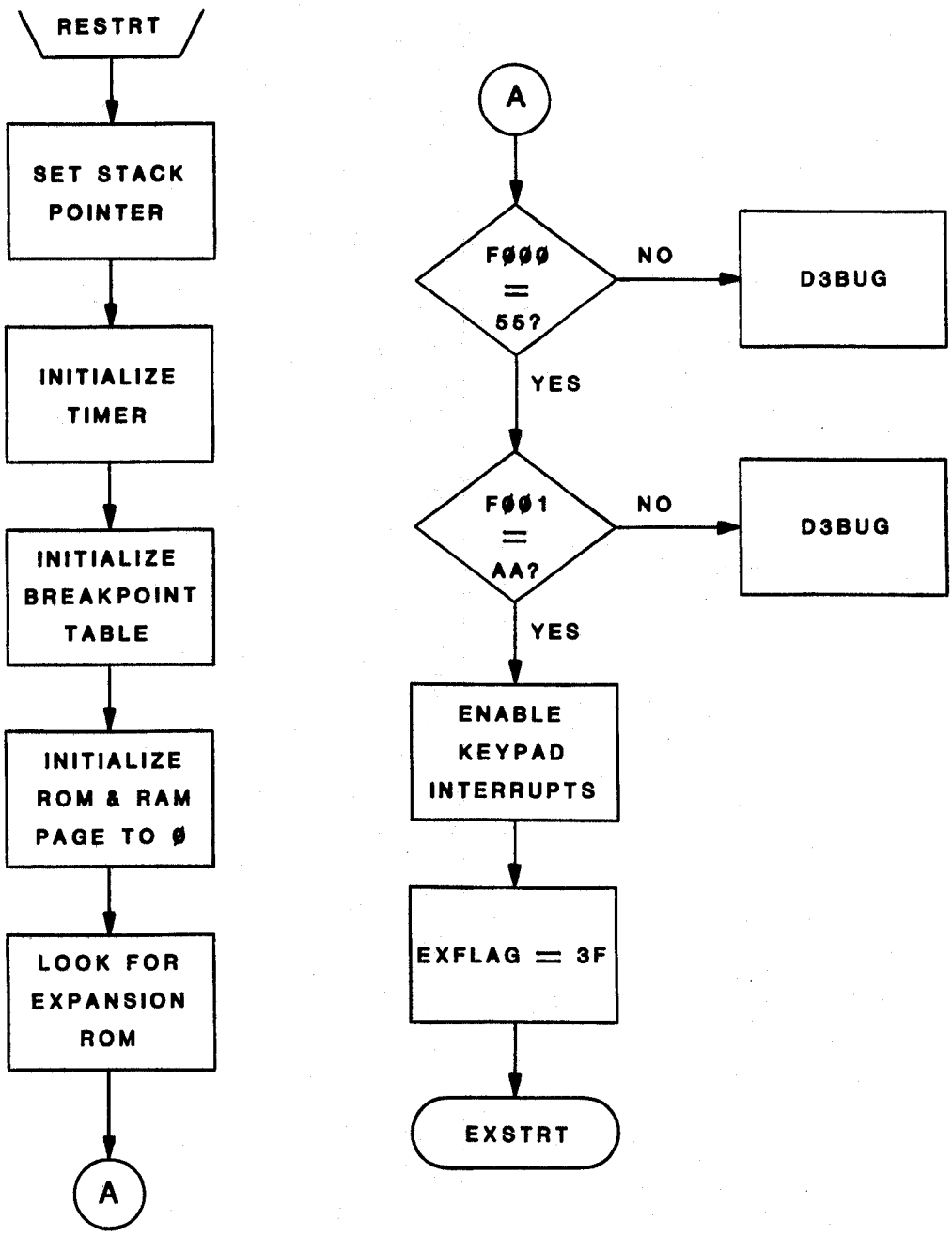
```
A000 MEMEND 00131*01244
812E MEMFLG 00678 00704 00729 00741*
9000 MEMSTR 00130*01239 01274
F198 MESAGE 00235 00427*00441 00447 00557 00569 00664 00864 00869 00959
            00990 01006
F19F MESGE1 00431*
F2DE MPIN   00617 00635 00643*
F495 NEXT1  00891 00900*
F4C6 NEXT2  00910 00926*
F4C2 NEXTRG 00915 00924*
812F NFLAG  00250 00759*
810F NMBTRC 00132*00751 00761 00844
F136 NOUT   00359 00361*
8142 NUMBER 01110 01124*
F2FA NXTAD1 00651 00654*
F2ED NXTADD 00647 00649*
F0C6 NXTCHR 00262*00269
F51F NXTOPT 00957 00972*
F575 NXTR   01015 01017*
F491 NXTREG 00894 00898*
F553 NXTRM  00999 01001*
F213 OBTN   00495*
F21F OBTNX  00496 00500*
F29C OF1200 00602 00604*
F2C8 OFF300 00610 00626*00630 00636
F292 OFLOAD 00600*01303
F4EB OPLOOP 00952*00976
F51B OPNXT  00964 00970*
F4DB OPTION 00874 00945*
F7E3 OPTMSG 00958 01458*
F14D OUT2H  00371 00374 00380*
F14F OUT2HA 00381*
F147 OUT2HS 00372 00374*00702 00783 00889
F140 OUT4HS 00371*00559 00666 00780 00908 00955 00961 01114
F12A OUTA   00350*00365 00382 00487 00993
F139 OUTAB  00365*
F12F OUTARN 00351 00357*
F416 OUTAS  00788 00816*
F12D OUTB   00355*00366 00384 01009
F624 OUTC1  01157 01159*
F156 OUTCHR 00159 00257 00279 00360 00376 00389*00431 00686 00739 00824
            01063 01159
F166 OUTCR1 00390 00402*
F149 OUTS   00280 00375*00450 00573 00684 00687 00701 00940 01115 01116
F4D8 OUTS1  00940*00954 00979
F52D OUTS2  00979*00991 00994 00995 01007 01010 01013
F22F PACK2B 00524*
F233 PACK4B 00526*
F234 PACKHX 00525 00527*
F23A PACKIT 00488 00529*
F52F PAGE   00873 00984*
8083 PAGERG 00133*01023 01039 01041
F619 PCRLF  01153*
8045 PIA1AC 00094*00192 00198 00226 00298 00711
8044 PIA1AD 00093*00194 00204 00224 00301
8047 PIA1BC 00096*00193 00197
8046 PIA1BD 00095*00195 00199
FF18 PIN1   00654 00657*
```

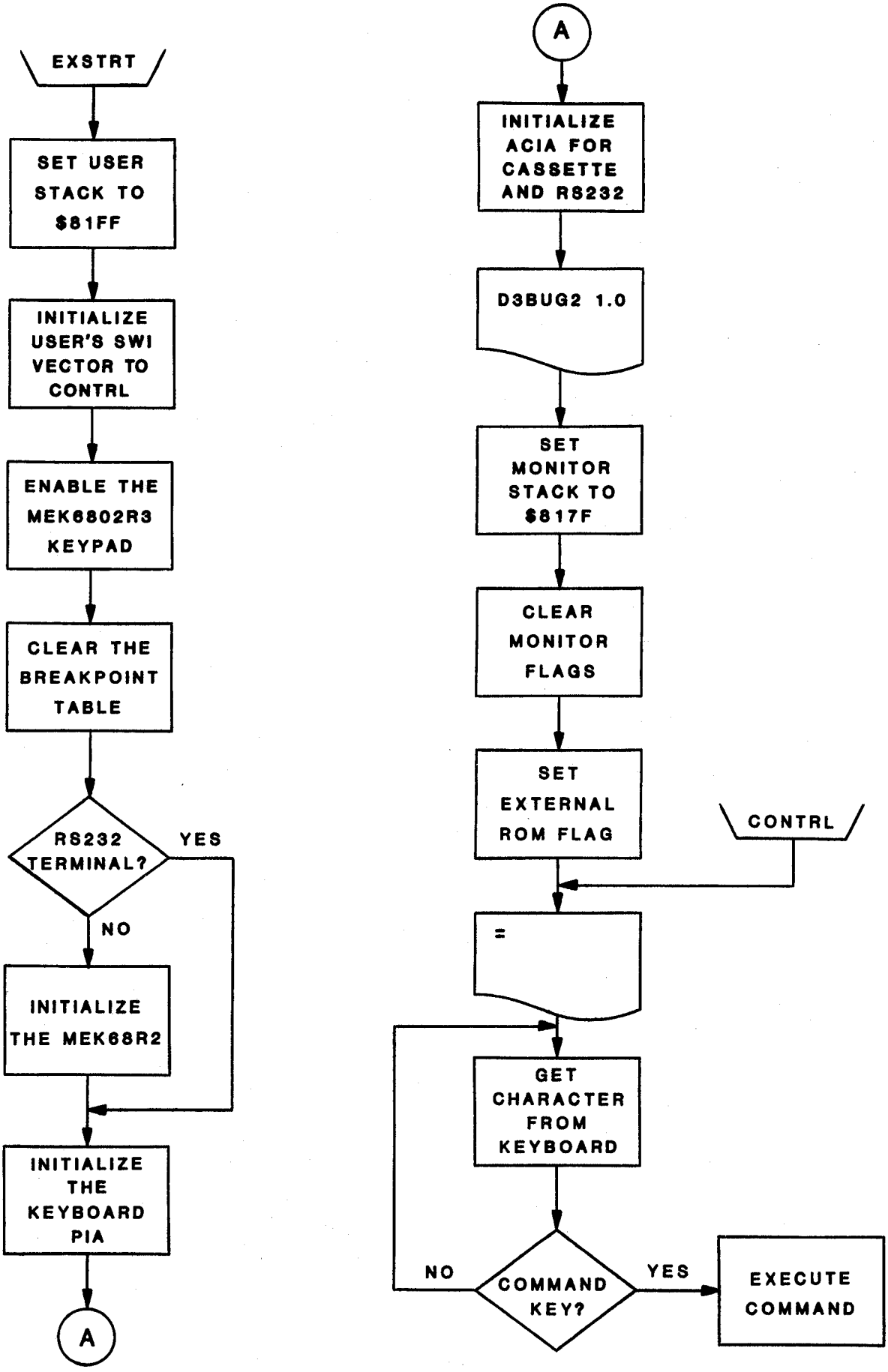
```
F7EF PLMSG  00568 01462*
F28C PNCH12 00589 00591*
F5E3 PNTBRK 01106 01109*01287
F471 PSTAK1 00876*01307
F285 PUNCH  00587*01305
FFA7 PUNCHA 00592 00593*
F304 QUIT   00616 00632 00663*
F7D5 RAMMSG 00989 01454*
F44F REGDIS 00760 00862*00877
F42B REGRMB 00184 00835*
F3B9 RENTRA 00708 00755*
F21B RETAD  00494 00498*
F110 RETASC 00313 00319*
F5C1 RETBAD 01057 01061 01079*
F1A5 RETENT 00435*00464
8111 RLINES 00778 00793 00795 00828*
FDFD RMBKPT 00835 00837*
FB44 ROLENT 00135*00530
8134 ROLFLG 00134*00463 00472 00493 00528 00727 00851 00893 00914 00963
            01056
F560 ROM    00997 01007*
F7DC ROMMSG 01005 01456*
F5DD RSTBRK 01107*01311
F60B RTURN1 01146*01160
8131 S3FLAG 00549 00560*00570 00590 00603 00609 00631
F672 SCROLU 01205*
F5D5 SETBRK 01104*01315
F125 SFTA4R 00200 00343*
F478 SINGLE 00871 00886*
F467 SKPMG1 00871*00882
F464 SKPMSG 00867 00870*
F357 SLASH  00691 00701*
817F SP     00136*00183 00246
8162 STARAD 00111*01163 01220 01276
810F STEMP  00117*00132
F3D7 STRTEX 00779*00794 00798
8111 TEMPX  00118*00119
810A TERM   00114*00164 00177 00291 00389 01154
F011 TERMIN 00164*00188 00709 00756
F3A3 TRACE  00746*01309
F3AB TRACE1 00750*00852 01301
FCD5 TRACE2 00754 00765*
F3B3 TRACLP 00753*00762
F59B TSTBAD 00998 01014 01055*01064
F4CE TSTCR  00898 00924 00933*00970 01002 01017
F1CA TSTENT 00461 00463*
8126 UA     00142*
8125 UB     00141*
8124 UCC    00140*00888 00900
F687 UDATE1 01221*01228 01268
8106 UIRQV  00147*
8104 UNMIV  00146*
F122 UNPACK 00341*00381
8129 UPC    00144*00558
F195 UPD    00417 00420*
F682 UPDATE 00420 01150 01200 01210 01218*
F07A US1    00189 00205 00212*
F082 US2    00216*
```

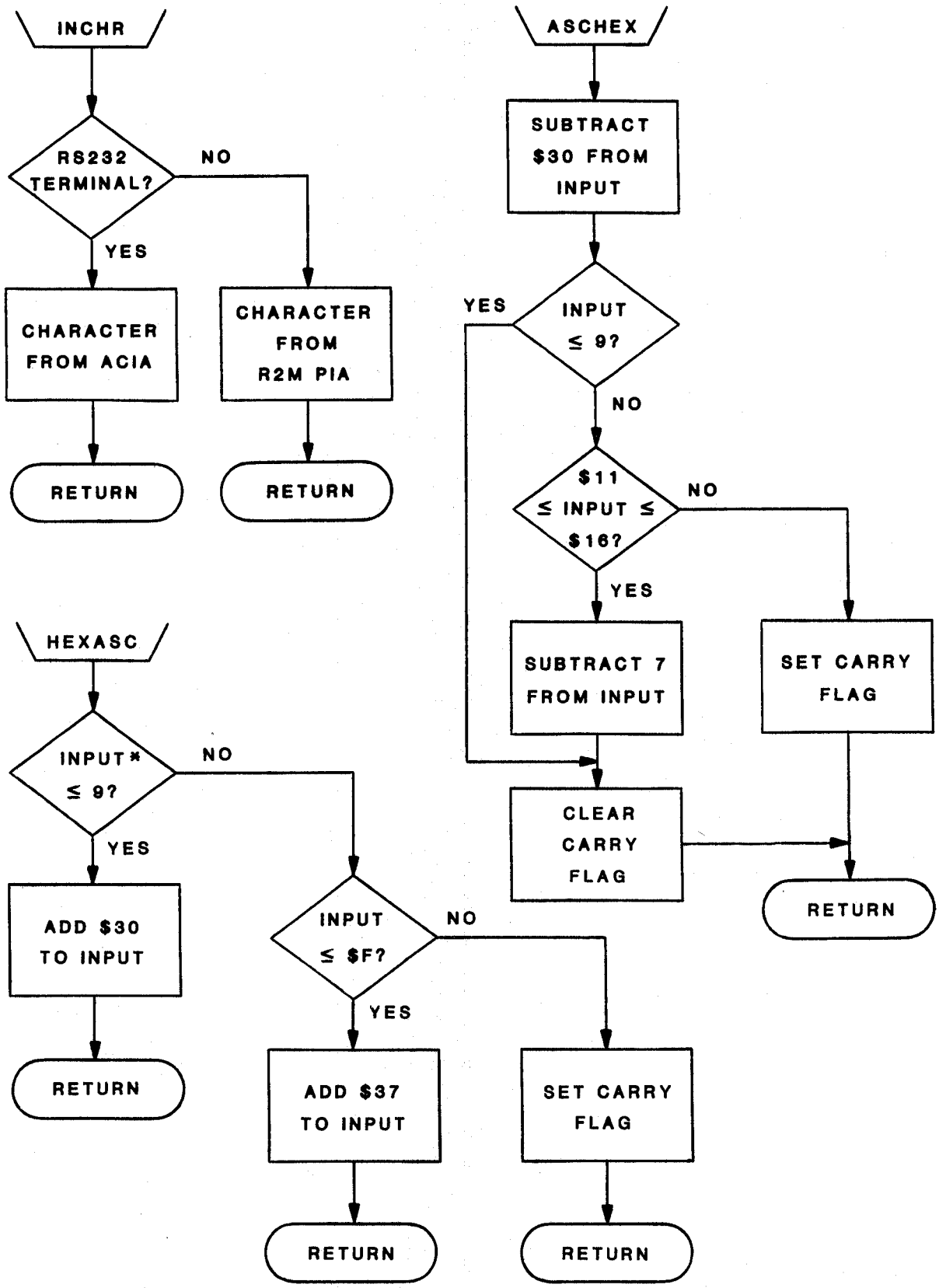
```
F5FC USEPG1 01133*01318
F601 USEPG2 01136*01321
F606 USEPG3 01139*01324
F085 USERS  00203 00217*
812B USP    00145*00182
8108 USWIV  00148*00185
8127 UX     00143*00907 00926
F243 VERFY  00545*01313
8132 VERIFY 00546 00561*00600
F301 XINCHR 00611 00615 00618 00622 00626 00633 00645 00649 00659*
8111 XTMP3  00119*00468 00498 00679 00699 00911 00913 00952 00960 00965
            00972
F3AD XTRACE 00749 00751*
8158 XTRFG  00251 00707 00752 00764*00845 00850
```

APPENDIX 3

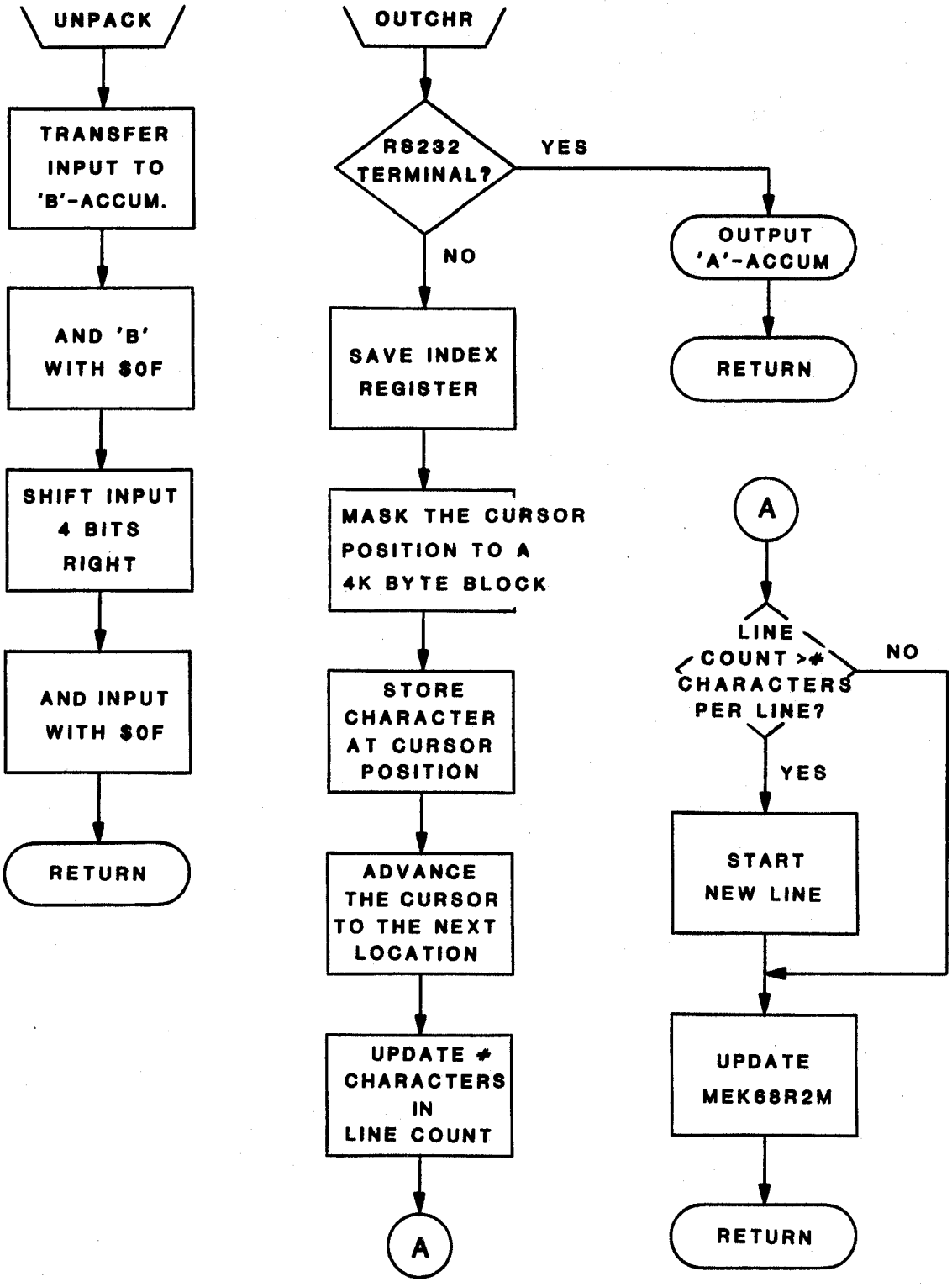
FLOW CHARTS, D3BUG2 1.0

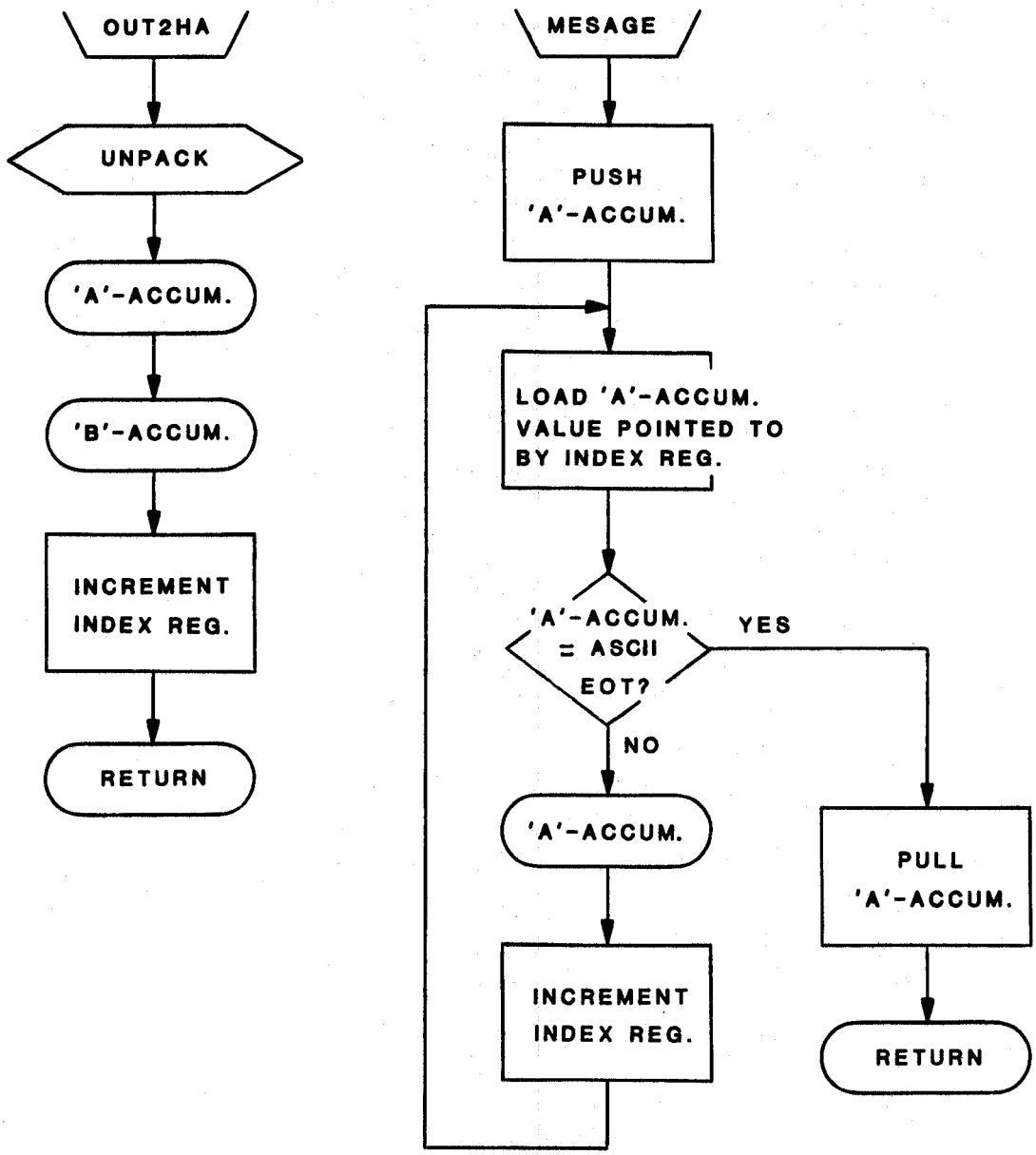

RESTRT
SET STACK POINTER
INITIALIZE TIMER
INITIALIZE BREAKPOINT TABLE
INITIALIZE ROM & RAM PAGE TO Ø
LOOK FOR EXPANSION ROM
A
A
FØØØ = 55?
NO
D3BUG
YES
FØØ1 = AA?
NO
D3BUG
YES
ENABLE KEYPAD INTERRUPTS
EXFLAG = 3F
EXSTRT

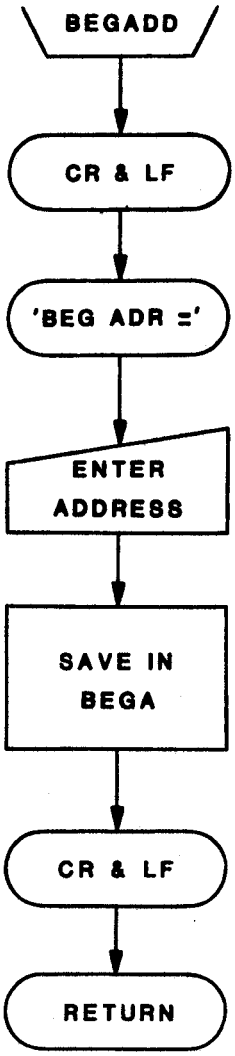

EXSTRT
SET USER STACK TO $81FF
INITIALIZE USER'S SWI VECTOR TO CONTRL
ENABLE THE MEK6802R3 KEYPAD
CLEAR THE BREAKPOINT TABLE
RS232 TERMINAL?
YES
NO
INITIALIZE THE MEK68R2
INITIALIZE THE KEYBOARD PIA
A
A
INITIALIZE ACIA FOR CASSETTE AND RS232
D3BUG2 1.0
SET MONITOR STACK TO $817F
CLEAR MONITOR FLAGS
SET EXTERNAL ROM FLAG
CONTRL
=
GET CHARACTER FROM KEYBOARD
COMMAND KEY?
NO
YES
EXECUTE COMMAND

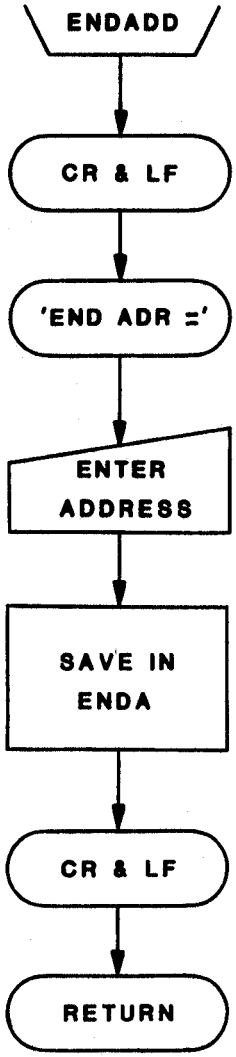


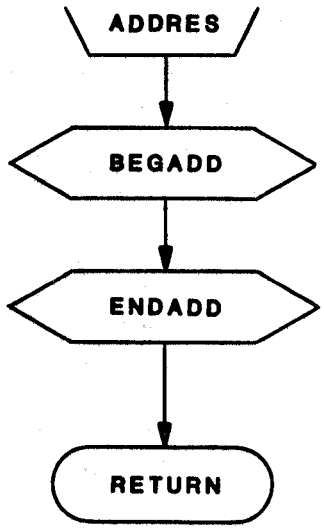
* INPUT IS VALUE IN THE 'A' - ACCUMULATOR

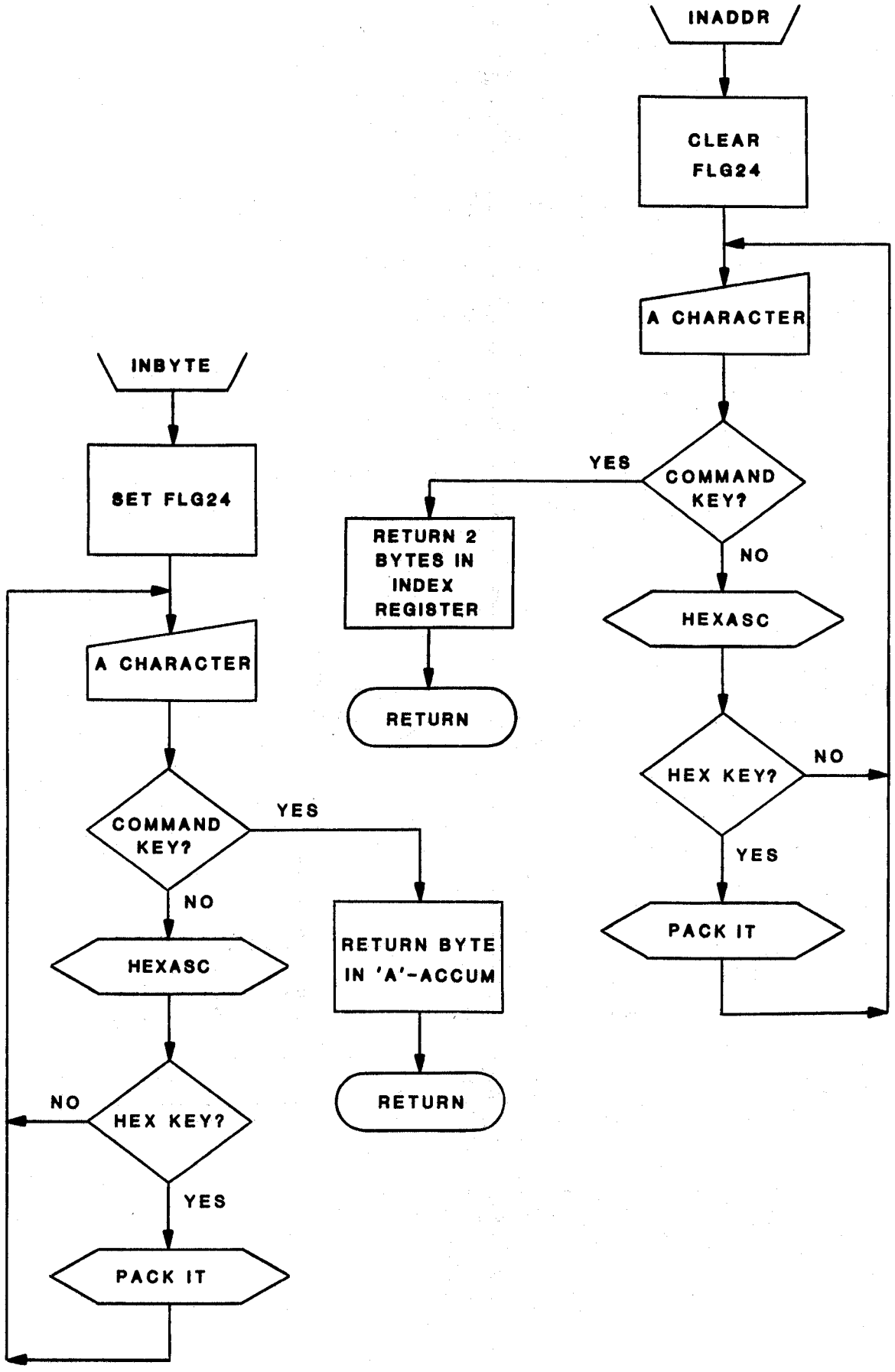

UNPACK
TRANSFER INPUT TO 'B'-ACCUM.
AND 'B' WITH $0F
SHIFT INPUT 4 BITS RIGHT
AND INPUT WITH $0F
RETURN
OUTCHR
RS232 TERMINAL?
YES
NO
OUTPUT 'A'-ACCUM
RETURN
SAVE INDEX REGISTER
MASK THE CURSOR POSITION TO A 4K BYTE BLOCK
STORE CHARACTER AT CURSOR POSITION
ADVANCE THE CURSOR TO THE NEXT LOCATION
UPDATE # CHARACTERS IN LINE COUNT
A
A
LINE COUNT >= # CHARACTERS PER LINE?
NO
YES
START NEW LINE
UPDATE MEK68R2M
RETURN

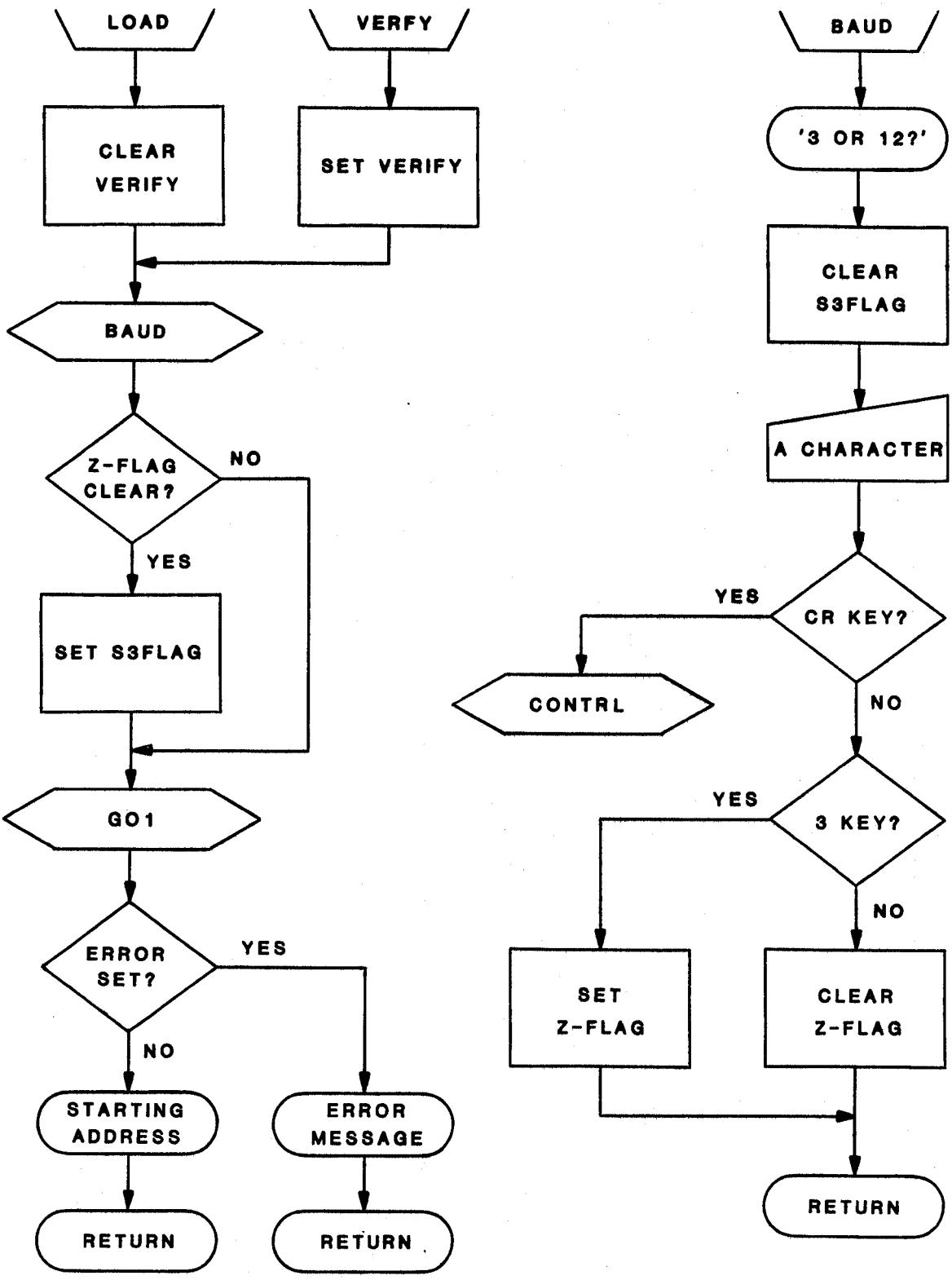

OUT2HA
UNPACK
'A'-ACCUM.
'B'-ACCUM.
INCREMENT INDEX REG.
RETURN
MESAGE
PUSH 'A'-ACCUM.
LOAD 'A'-ACCUM. VALUE POINTED TO BY INDEX REG.
'A'-ACCUM. = ASCII EOT?
YES
NO
'A'-ACCUM.
INCREMENT INDEX REG.
PULL 'A'-ACCUM.
RETURN

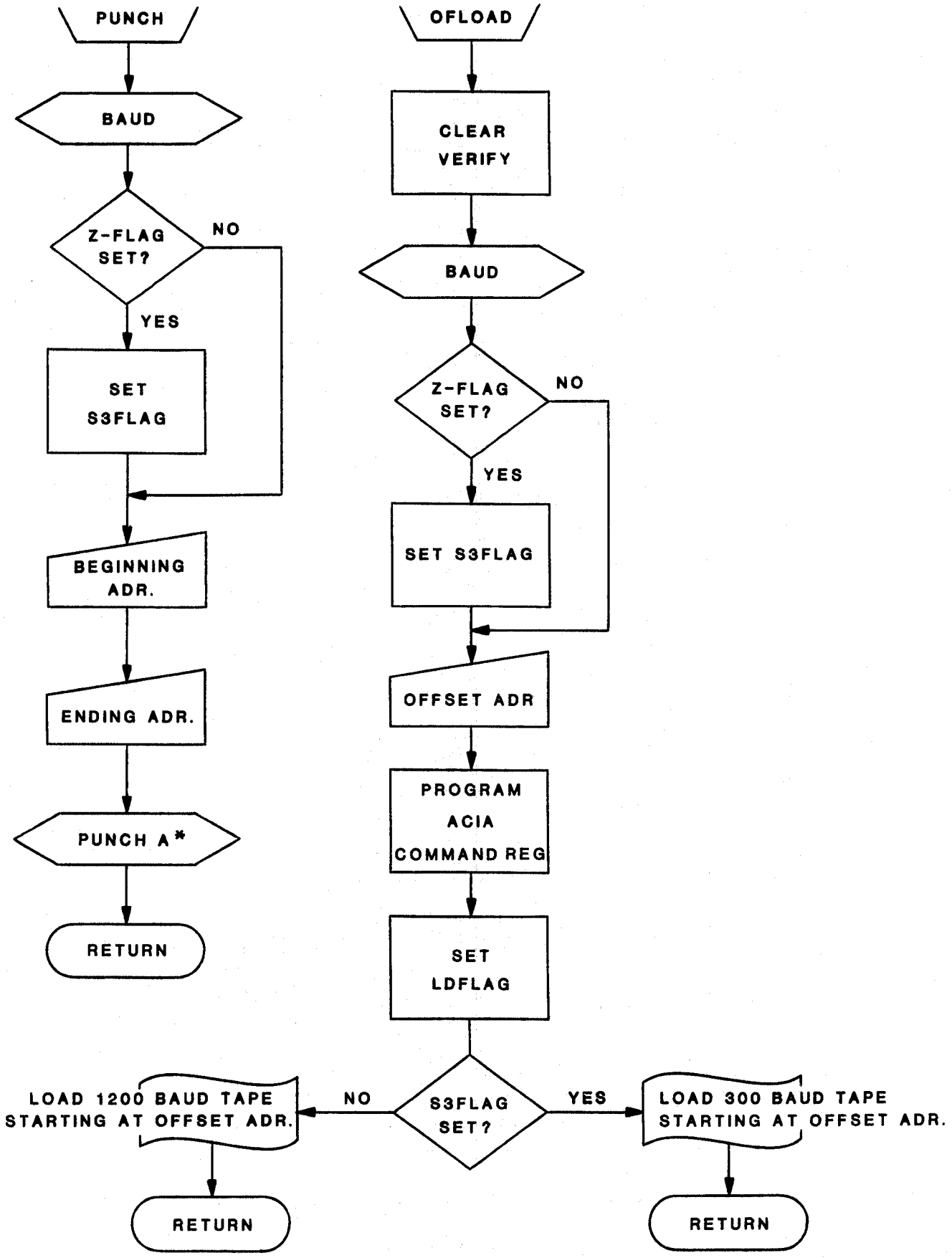

BEGADD
CR & LF
'BEG ADR ='
ENTER ADDRESS
SAVE IN BEGA
CR & LF
RETURN


ENDADD
CR & LF
'END ADR ='
ENTER ADDRESS
SAVE IN ENDA
CR & LF
RETURN

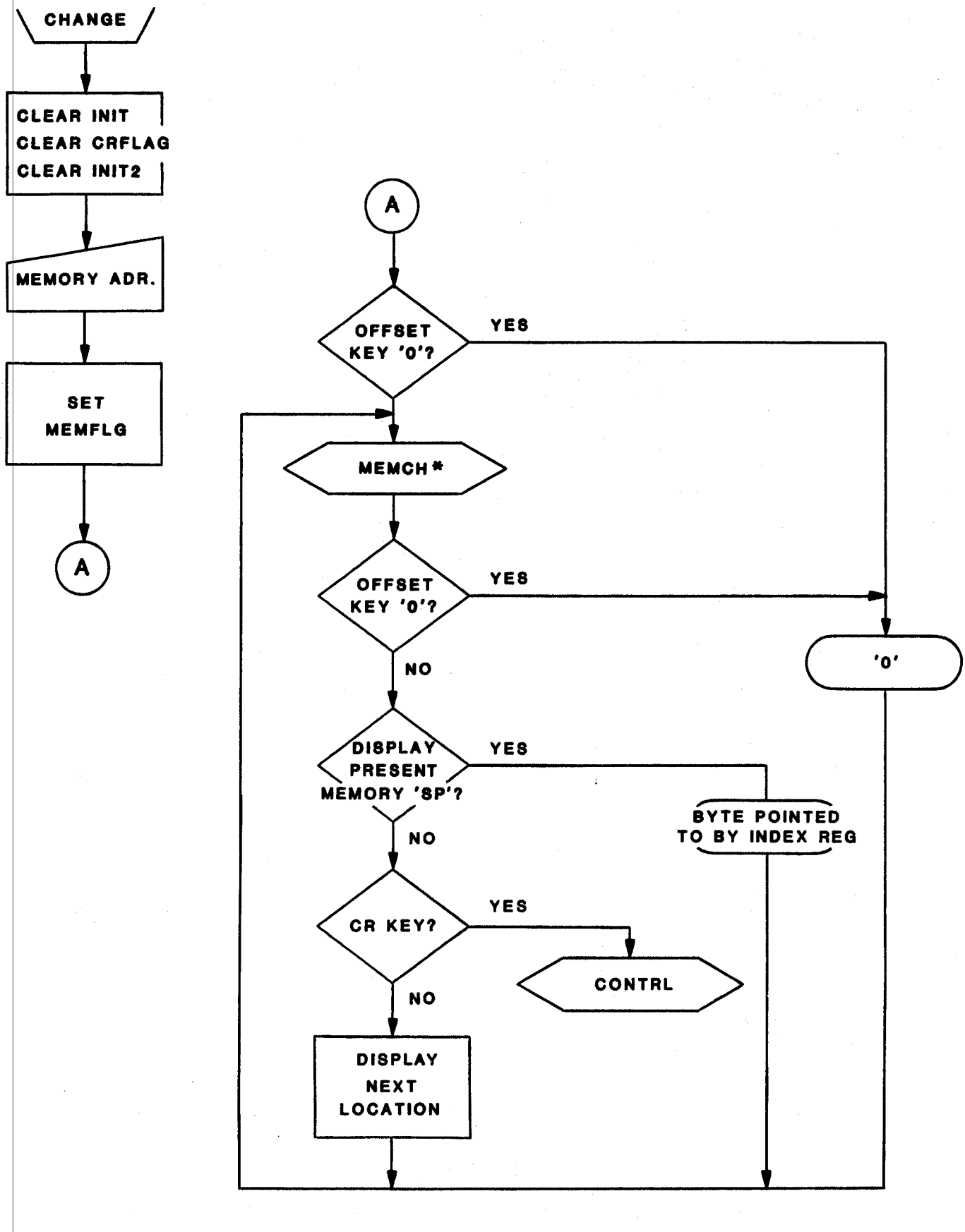

ADDRES
BEGADD
ENDADD
RETURN


INADDR
CLEAR FLG24
A CHARACTER
COMMAND KEY?
YES
RETURN 2 BYTES IN INDEX REGISTER
RETURN
NO
HEXASC
HEX KEY?
NO
YES
PACK IT
INBYTE
SET FLG24
A CHARACTER
COMMAND KEY?
YES
RETURN BYTE IN 'A'-ACCUM
RETURN
NO
HEXASC
HEX KEY?
NO
YES
PACK IT

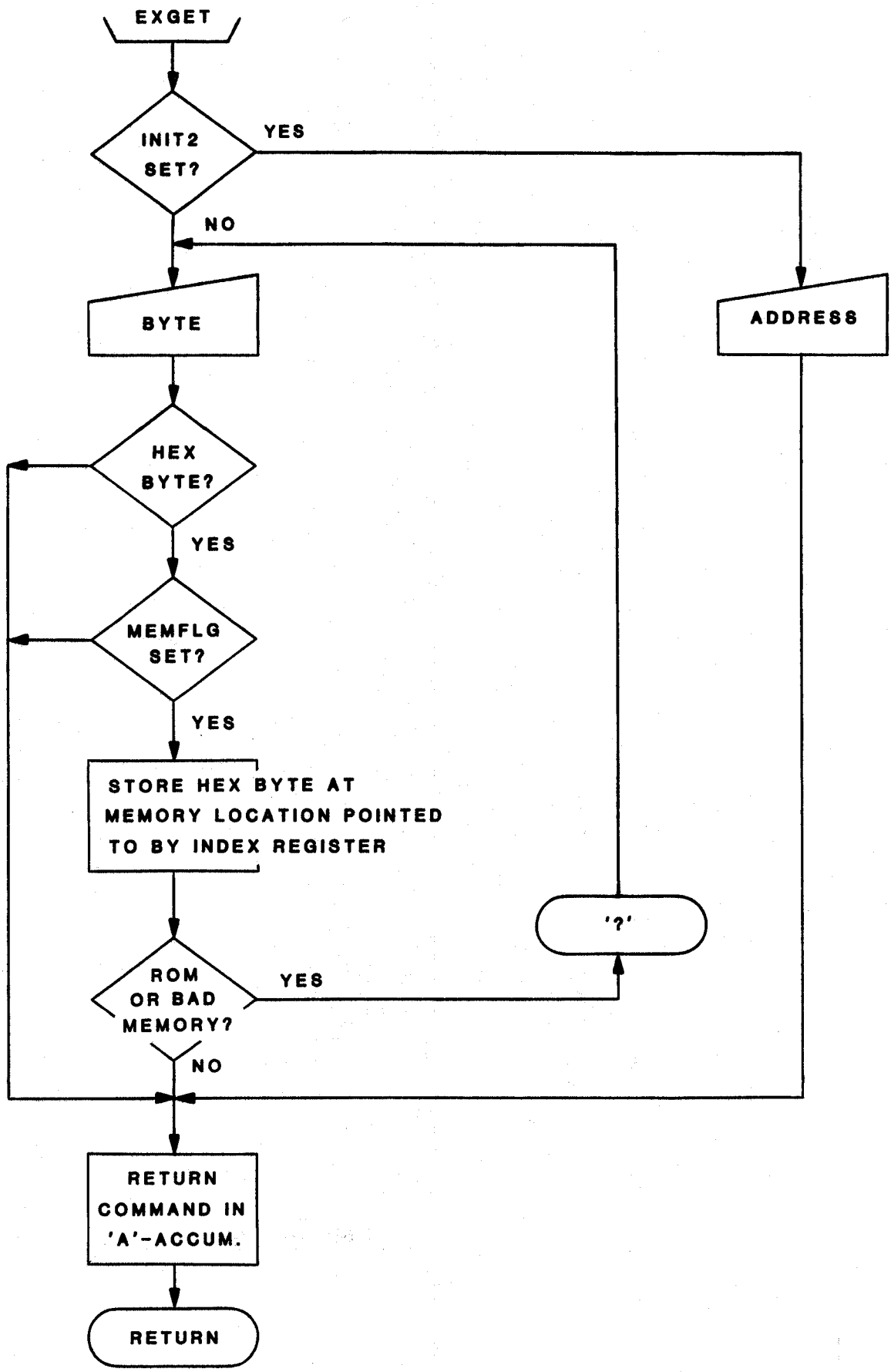

LOAD
VERFY
CLEAR VERIFY
SET VERIFY
BAUD
Z-FLAG CLEAR?
NO
YES
SET S3FLAG
GO1
ERROR SET?
YES
NO
STARTING ADDRESS
ERROR MESSAGE
RETURN
RETURN
BAUD
'3 OR 12?'
CLEAR S3FLAG
A CHARACTER
YES
CR KEY?
CONTRL
NO
YES
3 KEY?
NO
SET Z-FLAG
CLEAR Z-FLAG
RETURN

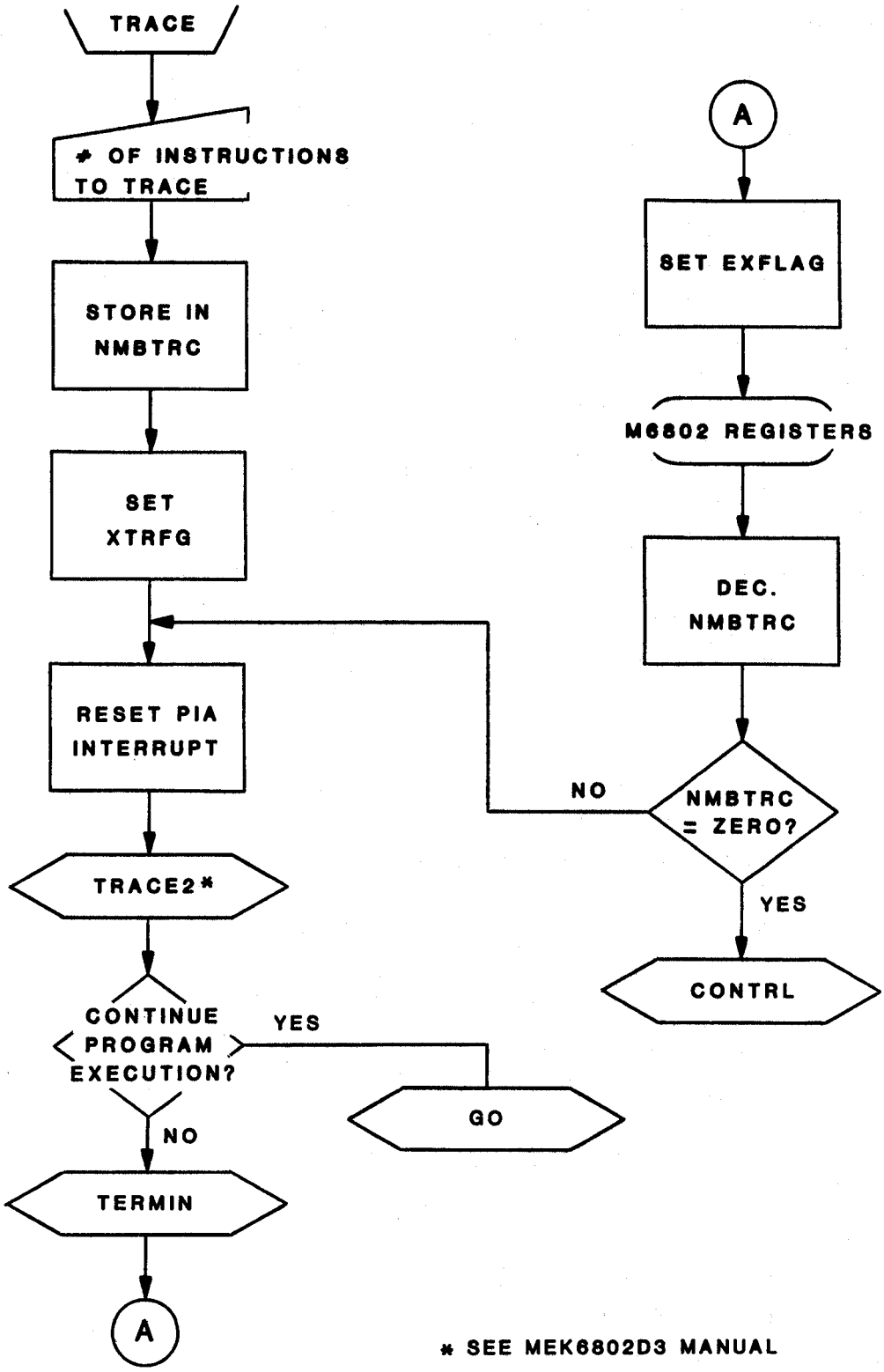


* SEE MEK6802D3 MANUAL

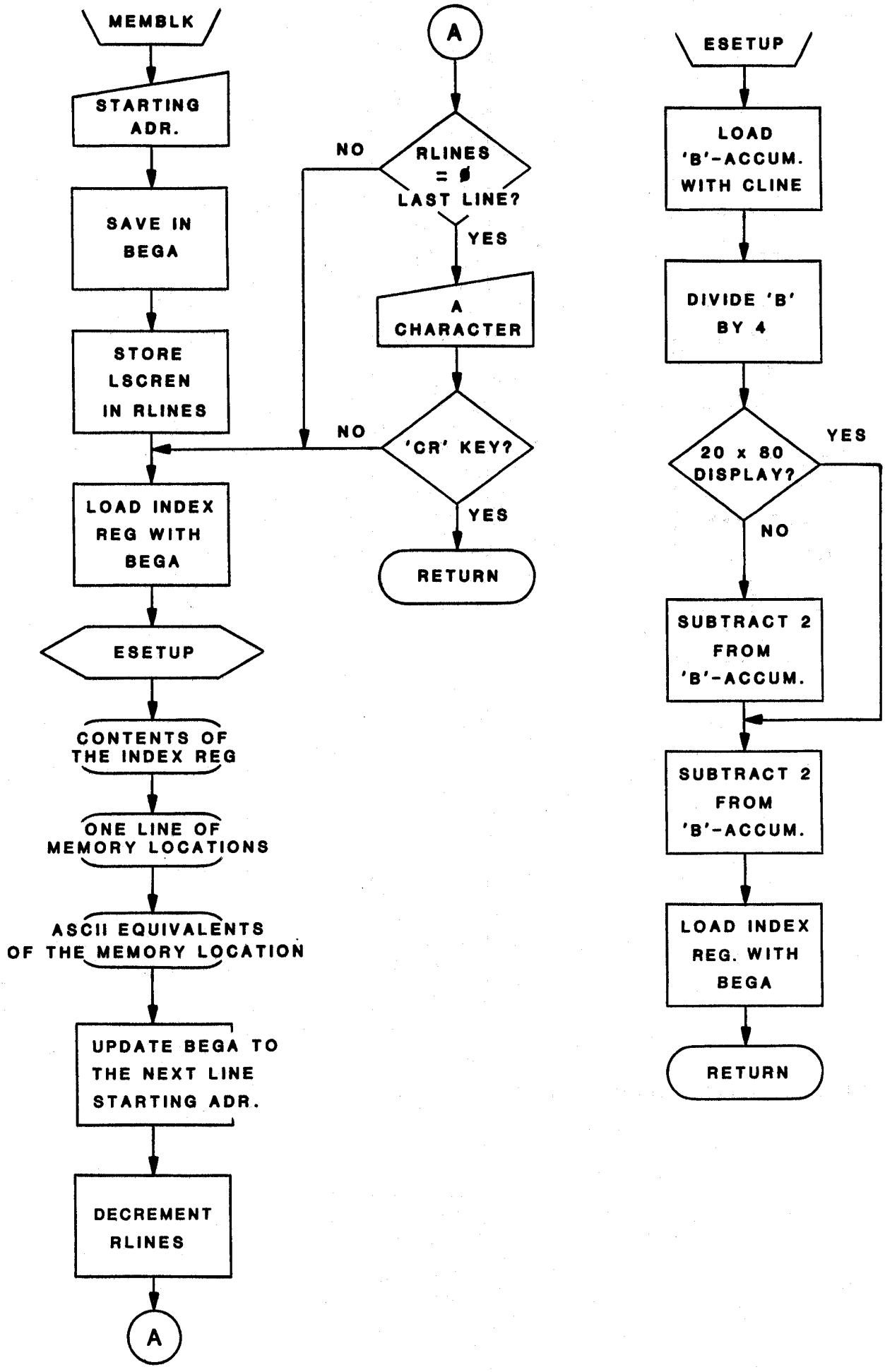


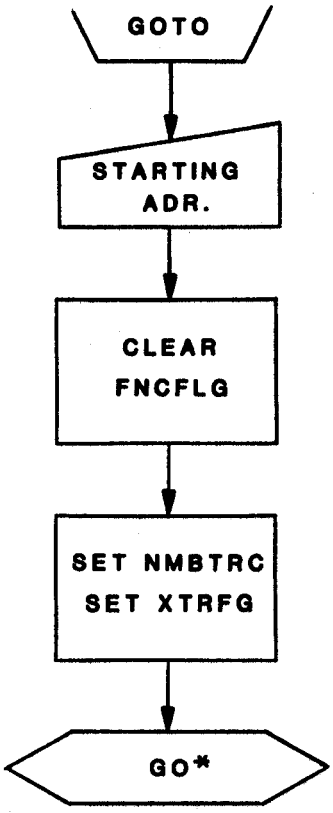
* SEE MEK6802D3 MANUAL

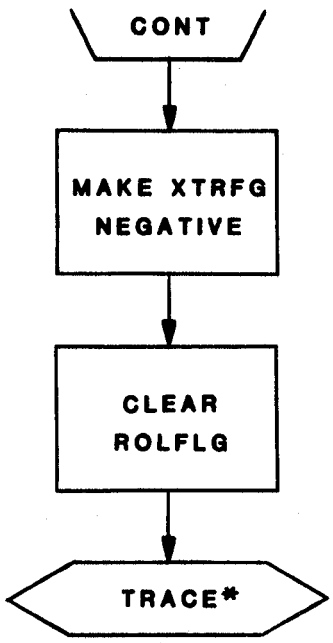

EXGET
INIT2 SET?
YES
NO
BYTE
ADDRESS
HEX BYTE?
YES
MEMFLG SET?
YES
STORE HEX BYTE AT MEMORY LOCATION POINTED TO BY INDEX REGISTER
'?'
ROM OR BAD MEMORY?
YES
NO
RETURN COMMAND IN 'A'-ACCUM.
RETURN

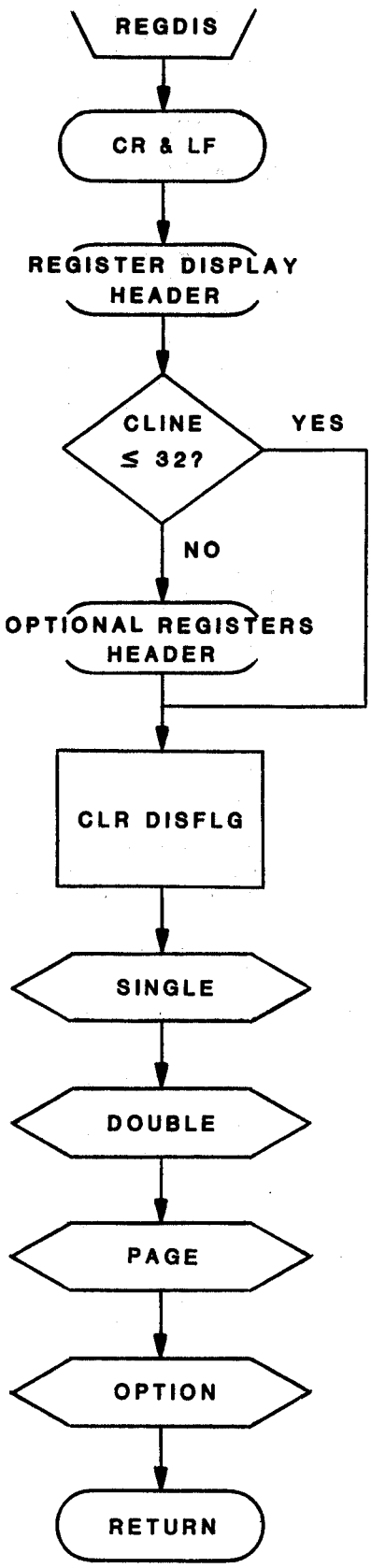


* SEE MEK6802D3 MANUAL

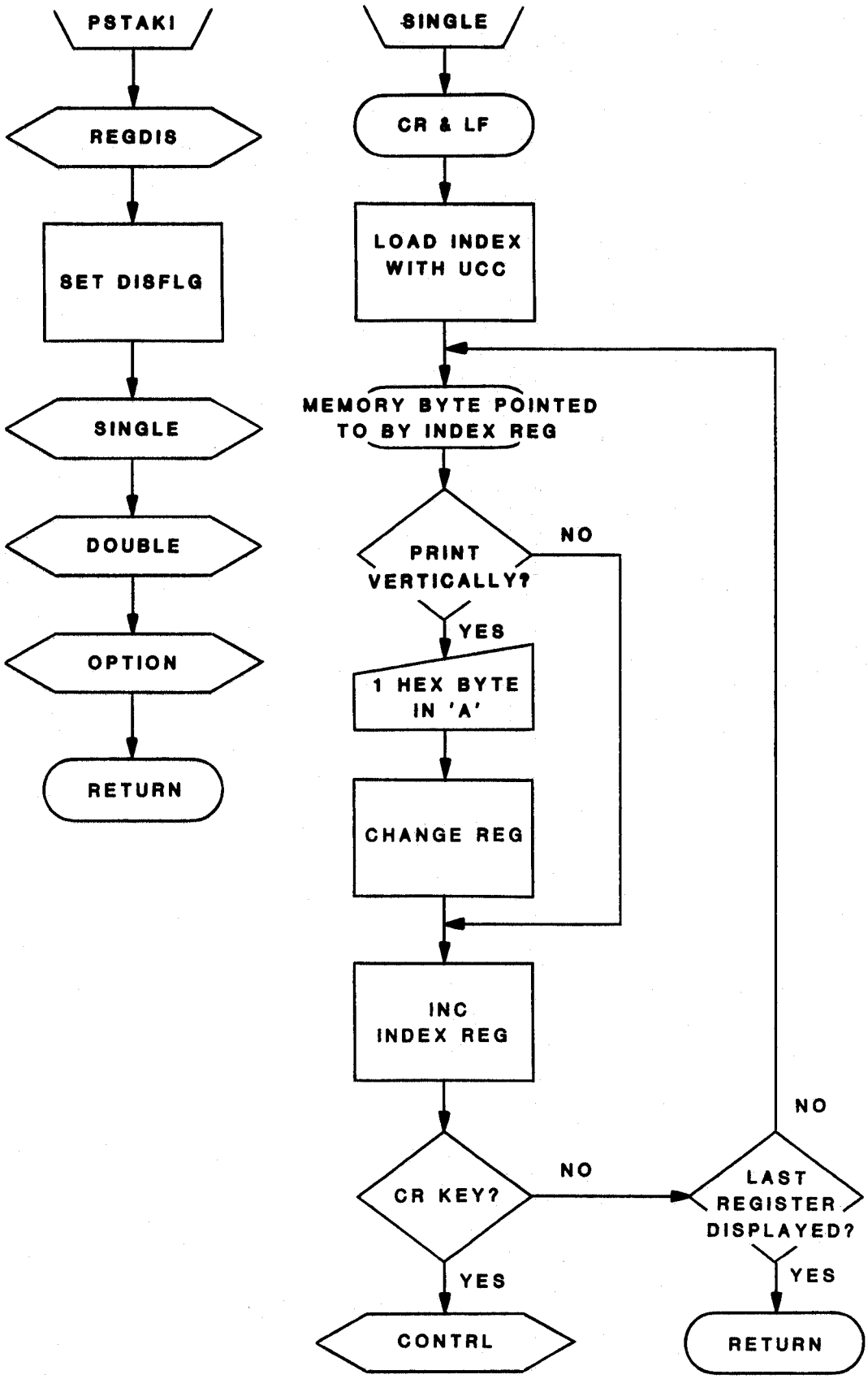

MEMBLK
STARTING ADR.
SAVE IN BEGA
STORE LSCREN IN RLINES
LOAD INDEX REG WITH BEGA
ESETUP
CONTENTS OF THE INDEX REG
ONE LINE OF MEMORY LOCATIONS
ASCII EQUIVALENTS OF THE MEMORY LOCATION
UPDATE BEGA TO THE NEXT LINE STARTING ADR.
DECREMENT RLINES
A
A
RLINES = 0 LAST LINE?
NO
YES
A CHARACTER
'CR' KEY?
NO
YES
RETURN
ESETUP
LOAD 'B'-ACCUM. WITH CLINE
DIVIDE 'B' BY 4
20 x 80 DISPLAY?
YES
NO
SUBTRACT 2 FROM 'B'-ACCUM.
SUBTRACT 2 FROM 'B'-ACCUM.
LOAD INDEX REG. WITH BEGA
RETURN

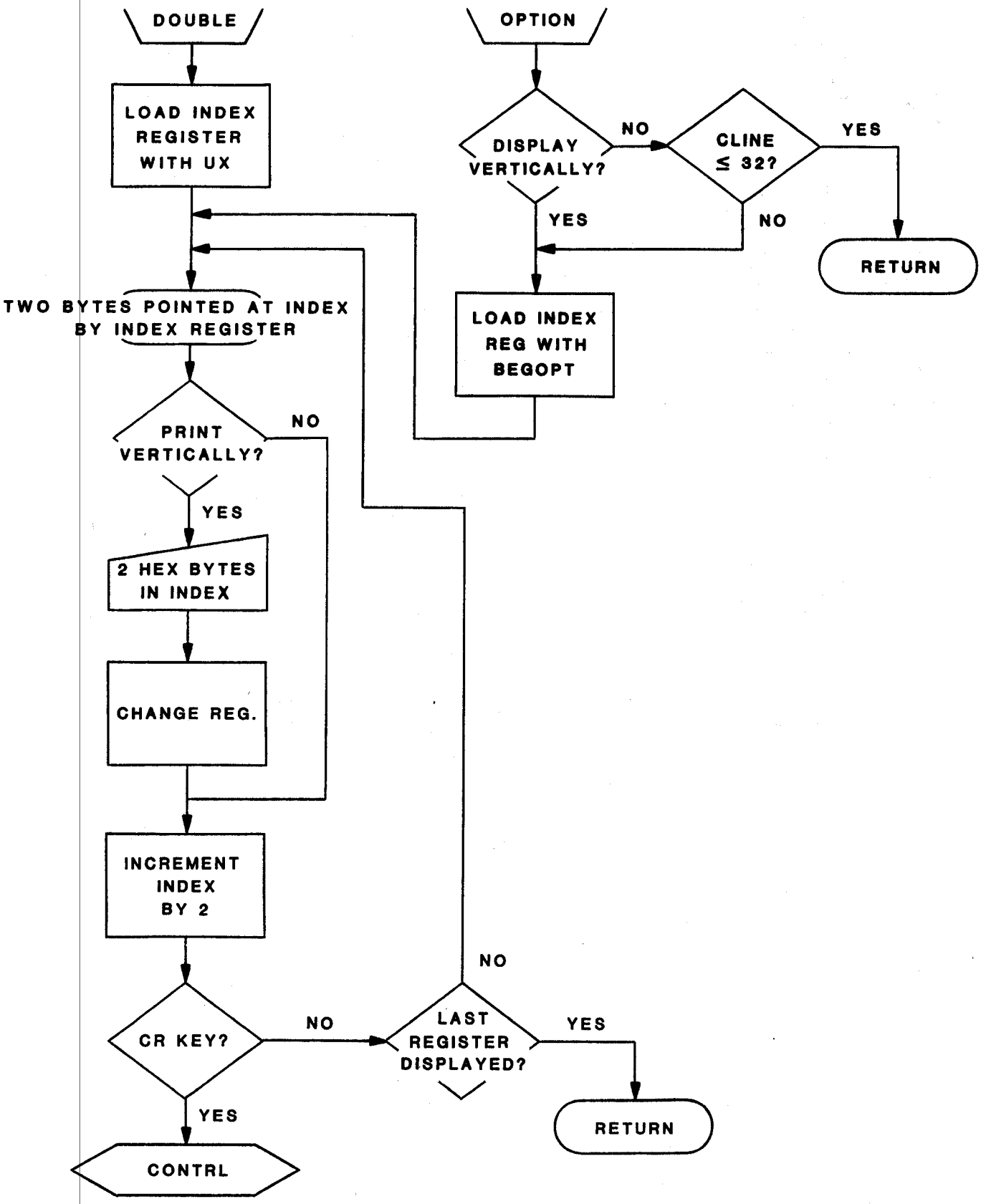


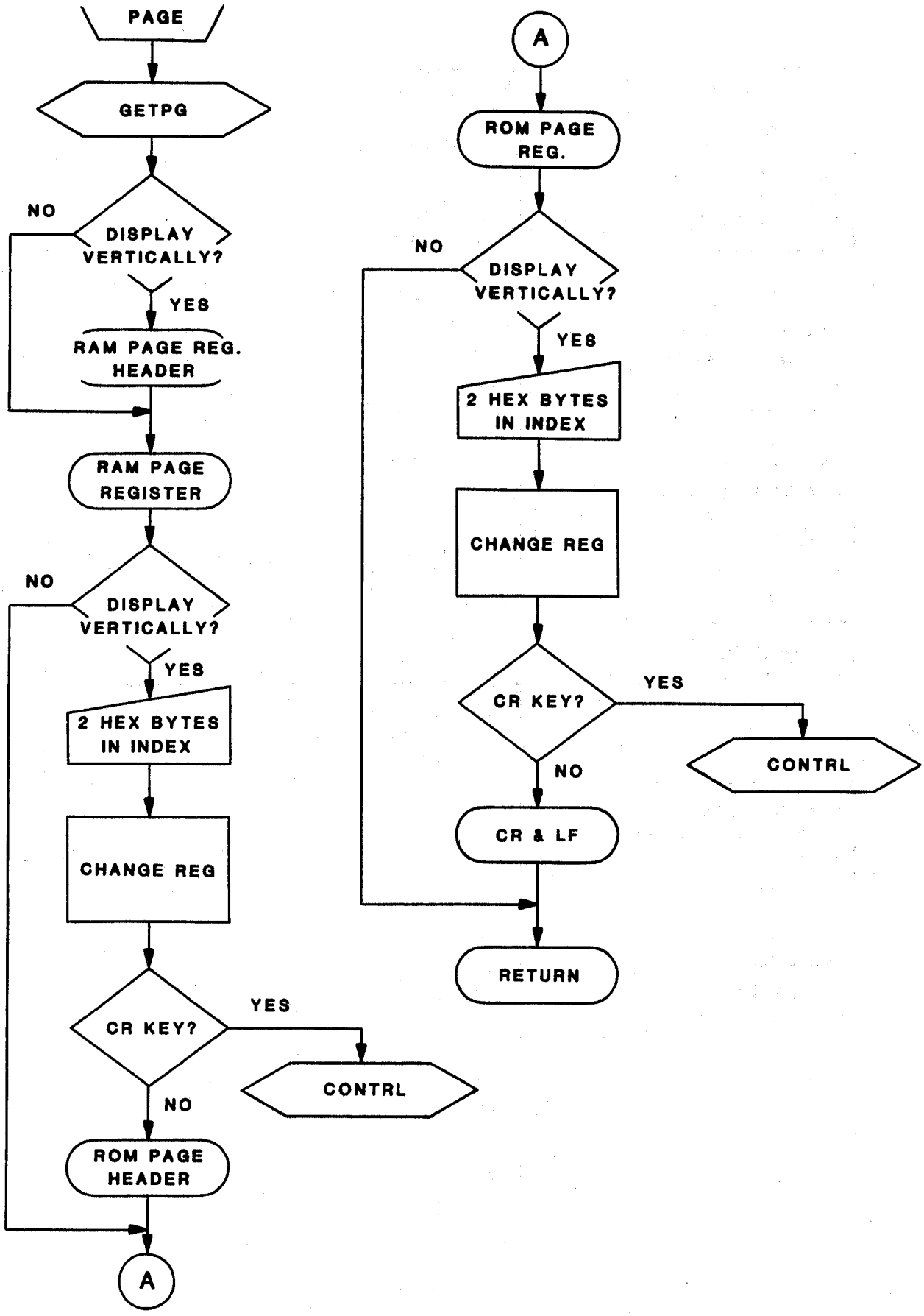


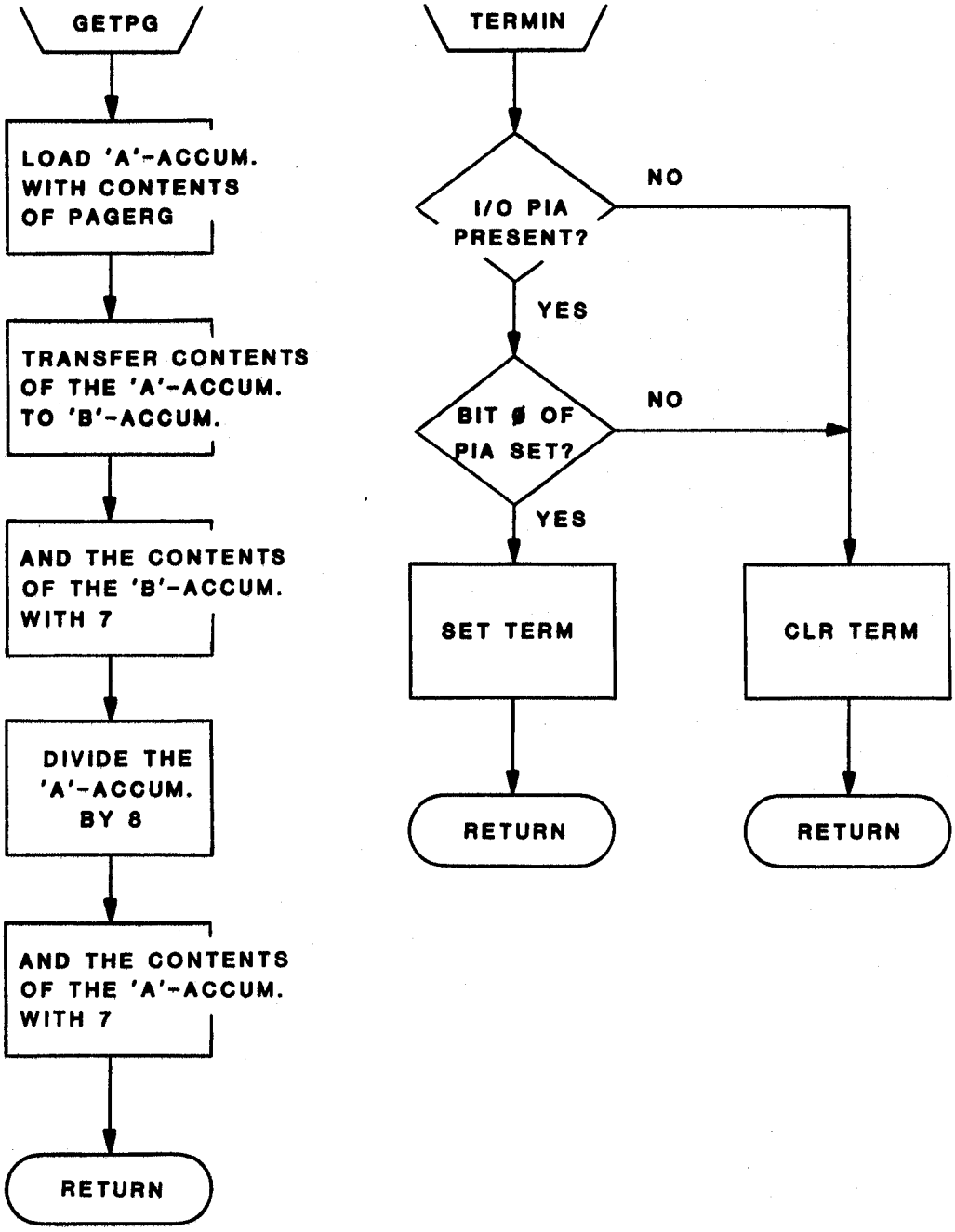


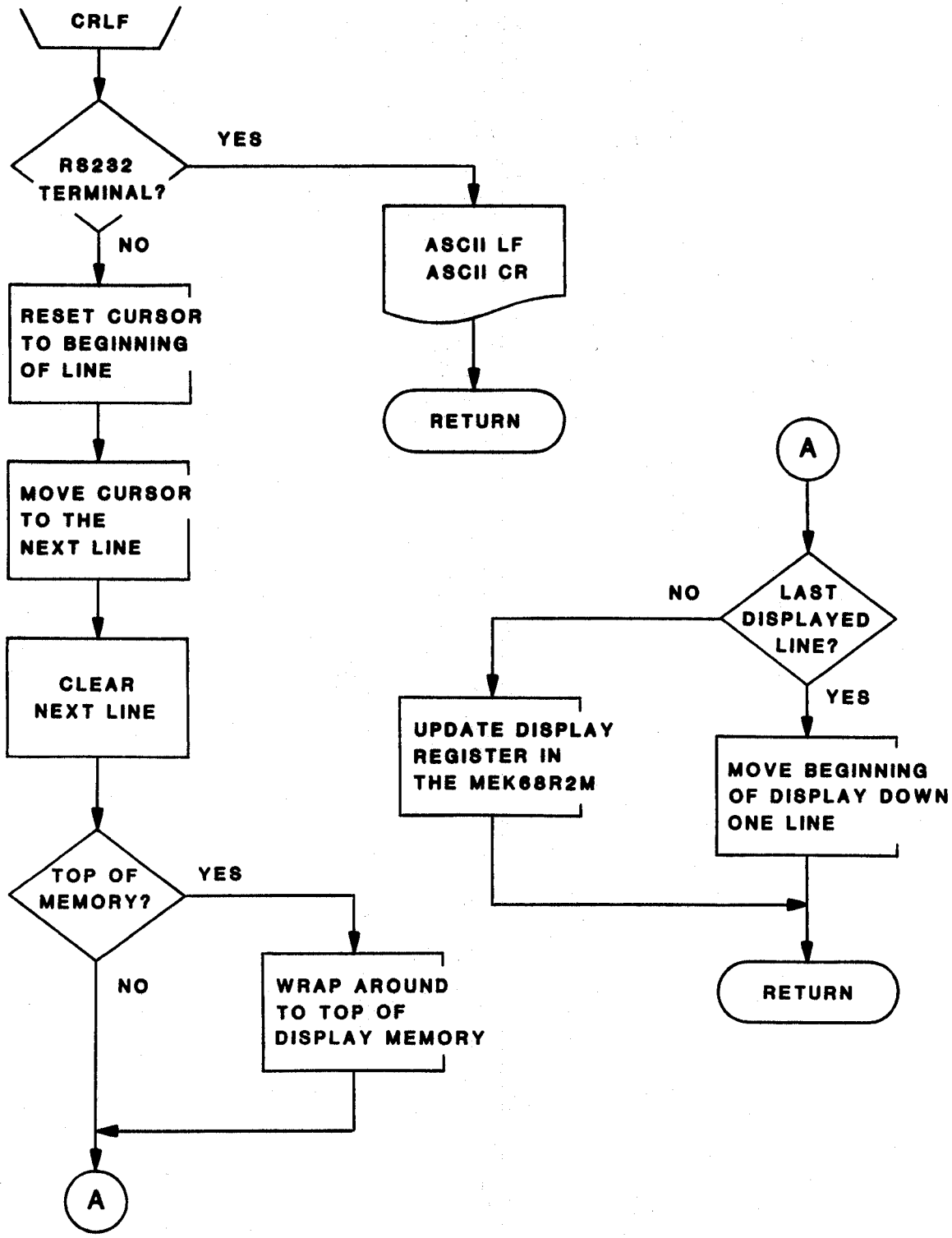
*SEE MEK6802D3 MANUAL FOR MORE INFORMATION


PSTAKI
REGDIS
SET DISFLG
SINGLE
DOUBLE
OPTION
RETURN
SINGLE
CR & LF
LOAD INDEX WITH UCC
MEMORY BYTE POINTED TO BY INDEX REG
PRINT VERTICALLY?
NO
YES
1 HEX BYTE IN 'A'
CHANGE REG
INC INDEX REG
CR KEY?
NO
YES
CONTRL
LAST REGISTER DISPLAYED?
NO
YES
RETURN


DOUBLE
LOAD INDEX REGISTER WITH UX
TWO BYTES POINTED AT INDEX BY INDEX REGISTER
PRINT VERTICALLY?
NO
YES
2 HEX BYTES IN INDEX
CHANGE REG.
INCREMENT INDEX BY 2
CR KEY?
NO
YES
CONTRL
LAST REGISTER DISPLAYED?
NO
YES
RETURN
OPTION
DISPLAY VERTICALLY?
NO
YES
CLINE ≤ 32?
YES
NO
RETURN
LOAD INDEX REG WITH BEGOPT


PAGE
GETPG
NO
DISPLAY VERTICALLY?
YES
RAM PAGE REG. HEADER
RAM PAGE REGISTER
NO
DISPLAY VERTICALLY?
YES
2 HEX BYTES IN INDEX
CHANGE REG
CR KEY?
YES
CONTRL
NO
ROM PAGE HEADER
A
A
ROM PAGE REG.
NO
DISPLAY VERTICALLY?
YES
2 HEX BYTES IN INDEX
CHANGE REG
CR KEY?
YES
CONTRL
NO
CR & LF
RETURN


GETPG
LOAD 'A'-ACCUM. WITH CONTENTS OF PAGERG
TRANSFER CONTENTS OF THE 'A'-ACCUM. TO 'B'-ACCUM.
AND THE CONTENTS OF THE 'B'-ACCUM. WITH 7
DIVIDE THE 'A'-ACCUM. BY 8
AND THE CONTENTS OF THE 'A'-ACCUM. WITH 7
RETURN
TERMIN
I/O PIA PRESENT?
NO
YES
BIT ø OF PIA SET?
NO
YES
SET TERM
CLR TERM
RETURN
RETURN

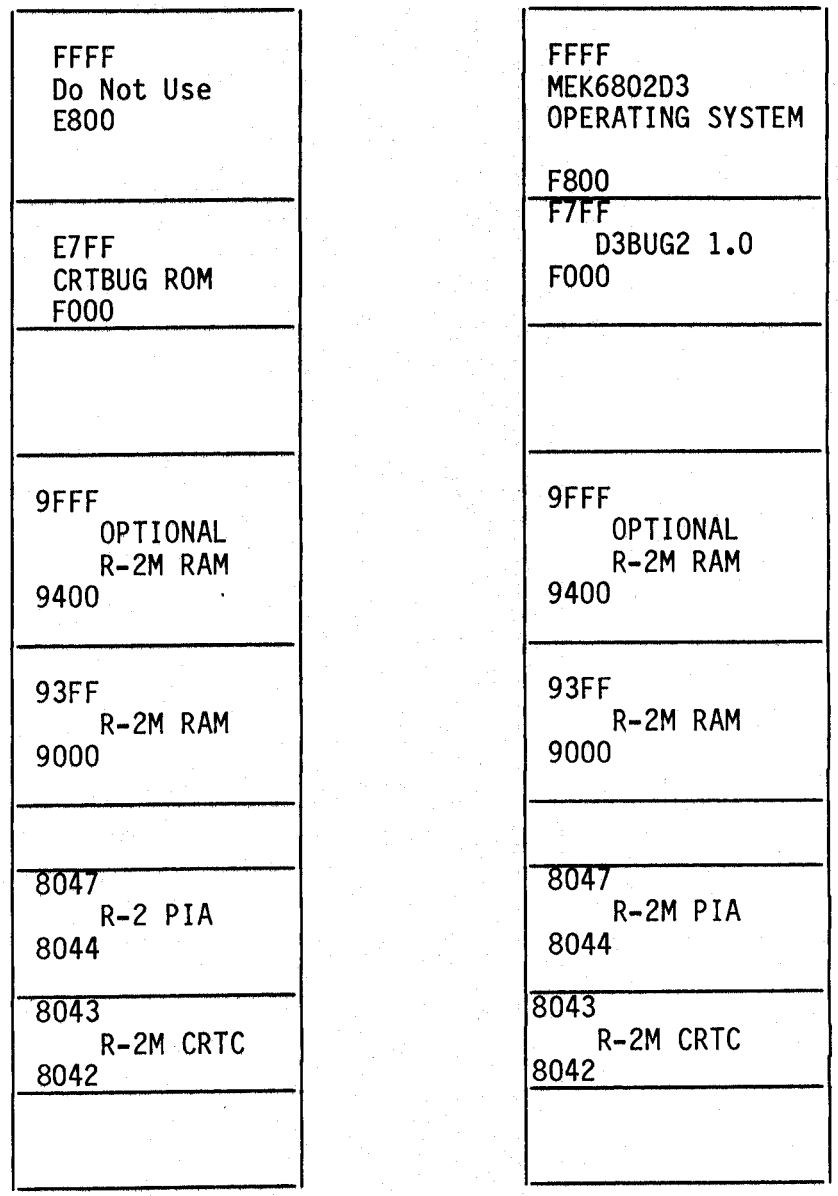

CRLF
RS232 TERMINAL?
YES
NO
ASCII LF
ASCII CR
RETURN
RESET CURSOR TO BEGINNING OF LINE
MOVE CURSOR TO THE NEXT LINE
CLEAR NEXT LINE
TOP OF MEMORY?
YES
NO
WRAP AROUND TO TOP OF DISPLAY MEMORY
A
A
LAST DISPLAYED LINE?
NO
YES
UPDATE DISPLAY REGISTER IN THE MEK68R2M
MOVE BEGINNING OF DISPLAY DOWN ONE LINE
RETURN

APPENDIX 4

DATA SHEET, MC6845

APPENDIX 5

MEMORY MAP



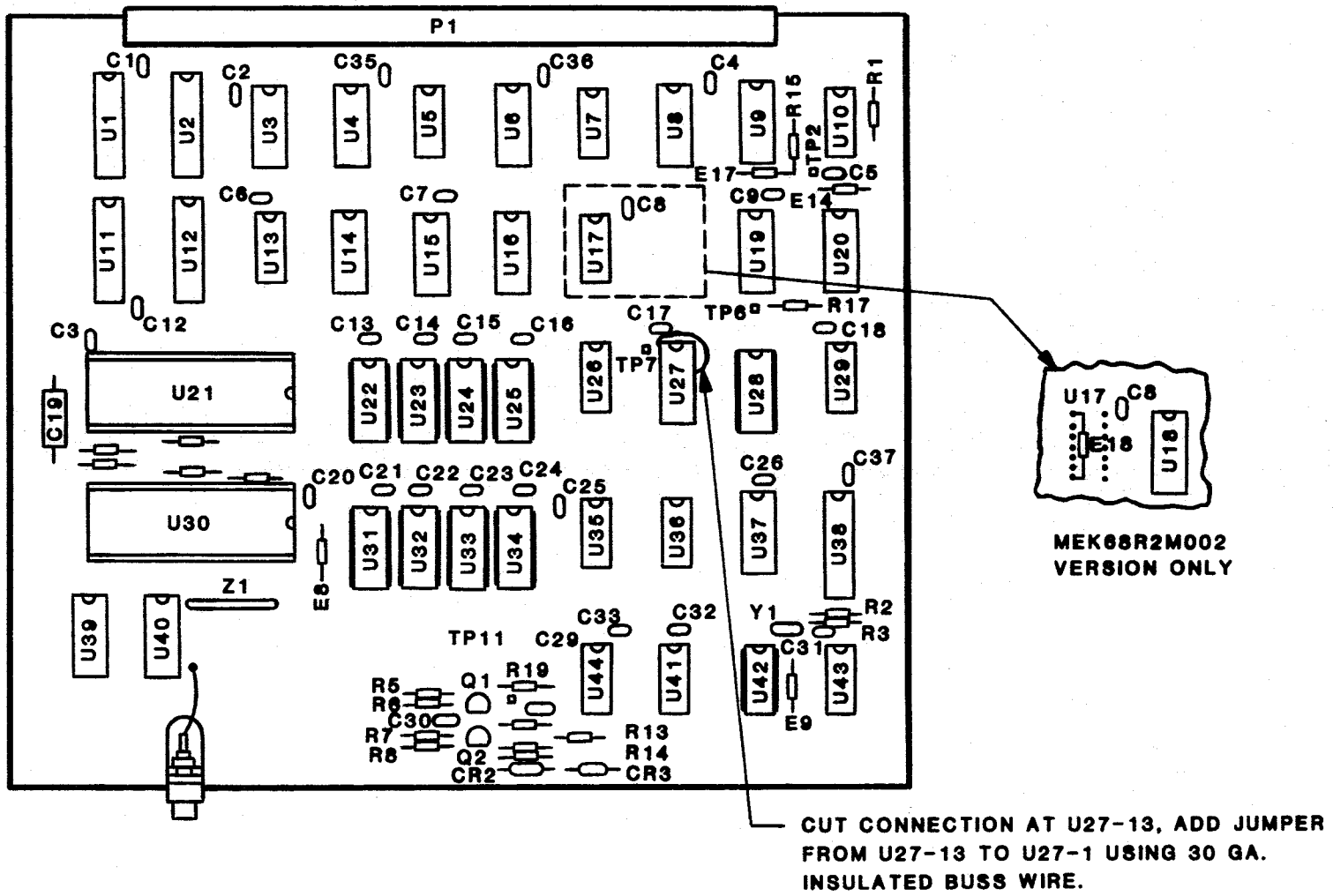
IN MEK6800D2 SYSTEM
(MEK68R2)

IN MEK6802D3 SYSTEM
(MEK68R2M)

APPENDIX 5. MEMORY MAP

APPENDIX 6

PARTS LAYOUT



APPENDIX 6. PARTS LAYOUT

APPENDIX 7

PARTS LIST

APPENDIX 7. PARTS LIST

Item Number	Quantity	Description	Part Number	Reference Designation
1	1	PC Board	MEK68R2/R2M	
2	2	Integrated Circuit	7405	U7, U44
3	1	Integrated Circuit	74S74	U10
4	1	Integrated Circuit	7493N	U42
5	2	Integrated Circuit	74LS04N	U5, U43
6	1	Integrated Circuit	74LS08N	U29
7	1	Integrated Circuit	74LS10N	U26
8	2	Integrated Circuit	74LS11N	U35, U36
9	1	Integrated Circuit	74LS28N	U17 (on R2 version only)
10	1	Integrated Circuit	74LS86N	U41
11	3	Integrated Circuit	74LS138N	U3, U4, U18 (U18 on R2M version only)
12	1	Integrated Circuit	74LS153N	U37
13	6	Integrated Circuit	74LS157N	U8, U9, U14, U15, U16, U20
14	1	Integrated Circuit	74LS166N	U19
15	1	Integrated Circuit	74LS174N	U27
16	4	Integrated Circuit	74LS241N	U1, U2, U11, U12
17	1	Integrated Circuit	74LS260N	U13
18	1	Integrated Circuit	74LS374N	U38
19	1	Integrated Circuit	8T97P/B	U6
20	2	Integrated Circuit	2114P45	U22, U31

APPENDIX 7. PARTS LIST (cont'd)

Item Number	Quantity	Description	Part Number	Reference Designation
21	1	Integrated Circuit	MC6845P	U21
22	1	Integrated Circuit	MCM6670P (ALT MCM6674P	U28
23	1	Integrated Circuit	MC6821P	U30
24	1	Jumper, Zero ohm Resistor		E18 (on R2M version only)
25	8	Jumper, Zero ohm Resistor		E2, E3, E5, E8, E9, E13, E14, E17
26	1	Resistor, 56 ohm 1/4 W, 5%		R7
27	2	Resistor, 220 ohm 1/4 W, 5%		R6, R14
28	1	Resistor, 390 ohm 1/4 W, 5%		R12
29	2	Resistor, 1K 1/4 W, 5%		R3, R2
30	1	Resistor, 2.2K 1/4 W, 5%		R8
31	6	Resistor, 3.3K 1/4 W, 5%		R1, R15, R16, R17, R18, R19
32	1	Resistor, 1.8K 1/4 W, 5%		R13
33	2	Resistor, 10K 1/4 W, 5%		R5, R11
34	1	Resistor Pkg, SIP 2K		Z1
35	2	Transistor	2N3904	Q1, Q2
36	1	Crystal, 12.528 MHz		Y1
37	2	Diode, Fast Switching	1N914	CR2, CR3

APPENDIX 7. PARTS LIST (cont'd)

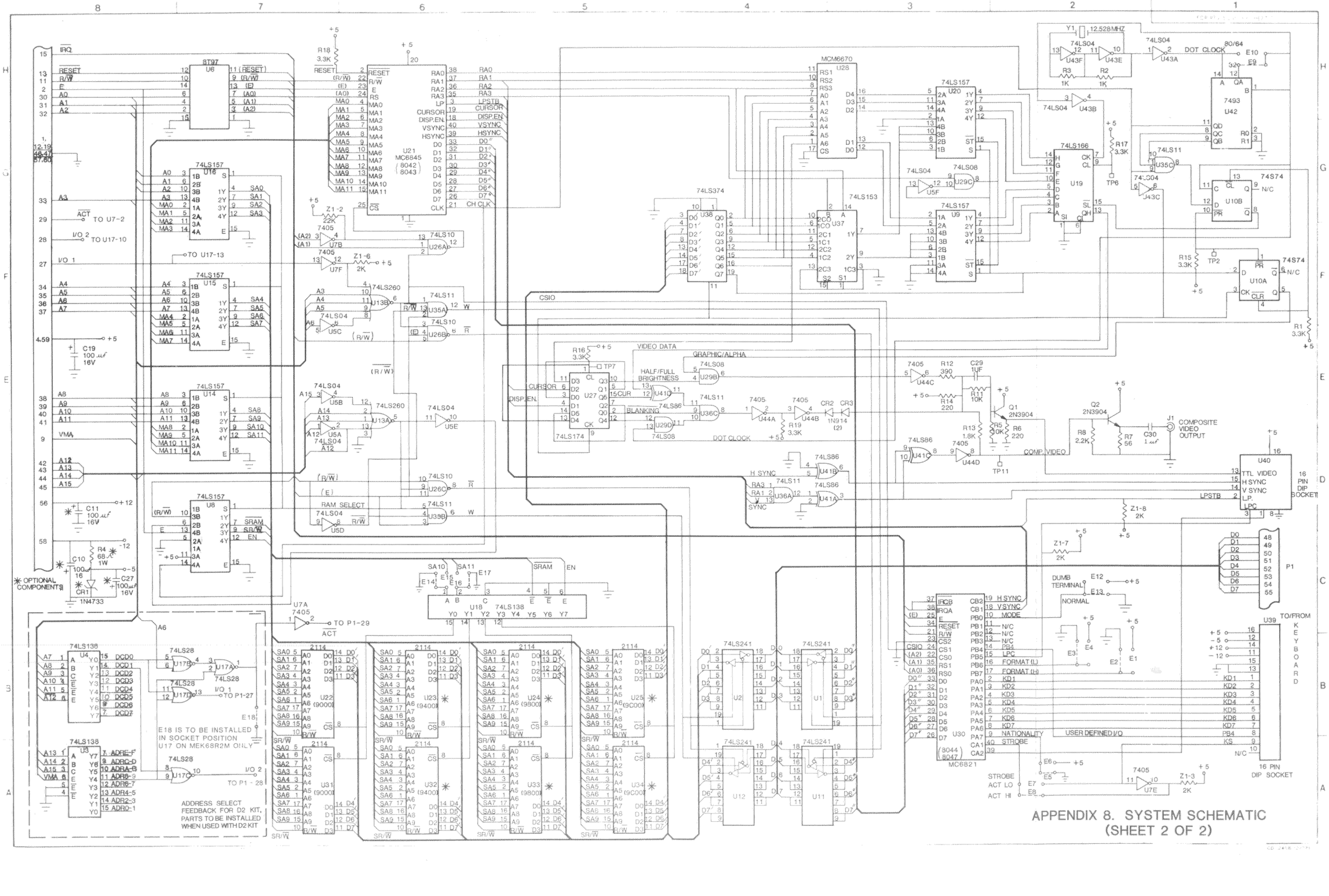
Item Number	Quantity	Description	Part Number	Reference Designation
38	29	Capacitor, 0.1 uF Monolithic Ceramic		C1-C9, C12-C18, C20-C26, C31, C32, C33, C35-C37
39	2	Capacitor, 1.0 uF Monolithic Ceramic		C29, C30
40	1	Capacitor, 10uF 25 V, Elect		C19
41	9	Socket, IC, 18-Pin		U22-U25, U28, U31-U34
42	2	Socket, IC, 40-Pin		U21,EU30
43	3	Connector, R/A-20		P1
44	2	Socket, IC, 16-Pin		U39, U40
45	3	Test Point		TP2, TP6, TP7
46	1	Test Point		TP11
47	1	Mtg Bracket, Phone Jack		
48	1	Phone Jack		
49	1	Socket, IC, 14-Pin		U42
50	1	Key, Polarizing		(For P1)
51	1	CRTBUG 1.0	MCM68316EP4	(For R2 only)
52	1	D3BUG2 1.0	MCM68A316E	(For R2M only)
53	1	Foam, Cond - 1/4 thk x 2 x 2-3/8		
54	2	Screw, 4-40 x 1/4 Lg. PPH		(Reference Only)
55	2	Nut, 4-40 Hex		(Reference Only)

APPENDIX 8

SYSTEM SCHEMATIC

Sheet 1: Index Page

Sheet 2: System Schematic


8
7
6
5
4
3
2
1
U6
8T97
U21
MC6845
8042
8043
U16
74LS157
U15
74LS157
U14
74LS157
U8
74LS157
U4
74LS138
U3
74LS138
U7A
7405
TO P1-29
ACT
TO U7-2
I/O 2
TO U17-10
TO U17-13
I/O 1
TO P1-27
I/O 2
TO P1 - 28
E18 IS TO BE INSTALLED IN SOCKET POSITION U17 ON MEK68R2M ONLY
ADDRESS SELECT FEEDBACK FOR D2 KIT, PARTS TO BE INSTALLED WHEN USED WITH D2 KIT
* OPTIONAL COMPONENTS
1N4733
CR1
C19 100 µf 16V
C11 100 µf 16V
C10 100 µf 16
C27 100 µf 16V
R4 68 1W
R18 3.3K
Z1-2 22K
Z1-6 2K
Z1-8 2K
Z1-7 2K
Z1-3 2K
U18 74LS138
U38 74LS374
U37 74LS153
U20 74LS157
U9 74LS157
U28 MCM6670
U19 74LS166
U42 7493
U10B 74S74
U10A 74S74
U40 16 PIN DIP SOCKET
U39 TO/FROM KEYBOARD
U30 MC6821
U2
U1
U12
U11
U22
U23
U24
U25
U31
U32
U33
U34
U27 74LS174
U7B
U5C
U5B
U5A
U5E
U5D
U13B
U17B
U17A
U17D
U17C
U26A
U26B
U35A
U35B
U36C
U36A
U41B
U41A
U29B
U29C
U29D
U44A
U44B
U44C
U44D
U41C
U43F
U43E
U43A
U43B
U35C
U43C
U10A
U7E
TP1
TP2
TP3
TP4
TP5
TP6
TP7
TP11
Y1 12.528MHZ
R1 3.3K
R2 1K
R3 1K
R5 10K
R6 220
R7 56
R8 2.2K
R9 3.3K
R11 10K
R12 390
R13 1.8K
R14 220
R15 3.3K
R16 3.3K
R17 3.3K
R19 3.3K
C29 1UF
C30 1 µf
Q1 2N3904
Q2 2N3904
J1 COMPOSITE VIDEO OUTPUT
P1
16 PIN DIP SOCKET
CSIO
VIDEO DATA
GRAPHIC/ALPHA
HALF/FULL BRIGHTNESS
CURSOR
DISP.EN.
BLANKING
DOT CLOCK
80/64
E10
E9
H SYNC
V SYNC
SYNC
COMP. VIDEO
TTL VIDEO
LPSTB
DUMB TERMINAL
NORMAL
E12
E13
E1
E2
E3
E4
E5
E6
E7
E8
E14
E15
E16
E17
SRAM
EN
SRAM SELECT
RAM SELECT
USER DEFINED I/O
STROBE
ACT LO
ACT HI
NATIONALITY
KD1
KD2
KD3
KD4
KD5
KD6
KD7
PB4
KS
N/C
APPENDIX 8. SYSTEM SCHEMATIC
(SHEET 2 OF 2)