Order this document
as AN441/D

AN441

MC68HC05E0 EPROM Emulator

By Peter Topping
MCU Applications Group
Motorola Ltd, East Kilbride

INTRODUCTION

The MC68HC05E0 is a versatile member of the M6805 family of microprocessors. Unlike most other versions it has no on-chip ROM but instead can address a full 64K of external memory. This memory could simply be a ROM or EPROM containing the required program but can also include RAM and/or additional hardware. In addition to the external busses required to support this capability, the MC68HC05E0 has the usual I/O, timers etc. found on single-chip microprocessors.

The EPROM emulator described here illustrates a typical application of this type of microprocessor. In addition to the program EPROM it employs a keyboard, LCD, serial communication and 64K of paged RAM. The emulator can replace with RAM the program EPROM or ROM (up to 64K x 8) in a microprocessor based target system. This is done by connecting the emulator to the target system via a cable to its EPROM socket.

The object code, which can be loaded serially or from an EPROM, can be inspected and modified with the use of a local keyboard and LCD display. The new or modified code can then be used by the target system without having to go through the procedure of erasing and re-programming an EPROM after each software change. A selectable offset in $0100 steps is available in order to position the code correctly in the target system's memory map.

The emulator facilitates the debugging of hardware and software for any system whose control program is to be contained in a 27(C)16/32/64/128/256/512 type EPROM. The control software includes branch offset calculation for 6805 code and is thus particularly suitable for debugging systems using one of the microprocessors from the M68(HC)05/01/11 ranges.

Two basic methods of loading a program are available. The first is applicable when the code is available in an existing EPROM. The contents of this EPROM can be transferred by the microprocessor into the RAM. This method requires an existing EPROM but will prove useful in applications where a small change to an existing program has to be checked before committing to an updated EPROM. This can be done without access to the source or object code. An EPROM can be read from the target system interface (through the emulator's buffers) or from a separate socket wired directly to the microprocessor. The former method allows one socket to be used for both EPROM reading and the target cable. The second method saves having to remove the target cable to read an EPROM but requires an additional socket.

Alternatively, data can be serially loaded in the form of Motorola S-records via an RS232 link. This code can come from a "COM" port of a PC (using the COPY command) or by tapping into the link between a computer and its terminal on a system using an RS232 connection between terminal and host. In this case a TYPE or LIST to the terminal should be used. A verify facility which compares the contents of RAM with serial S-records is also available, as is a routine to dump the current contents of the emulation RAM out on the RS232 interface.

freescale™
semiconductor

PRINCIPLE OF OPERATION

Figure 1 shows a block diagram of the emulator in each of its three main modes of operation. The data/address flow is controlled by MC74HC245 bidirectional tri-statable 8-bit buffers. They constitute two 18-bit buffers for address and control signals and two 8-bit buffers for data. The enabling and direction control signals are supplied by the MC68HC05E0 microprocessor.

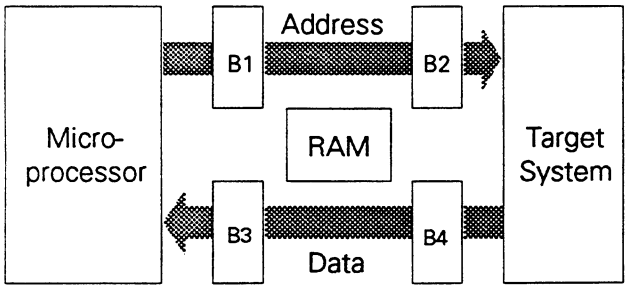

Figure 1a. Mode A:
direct access to the target system's interface

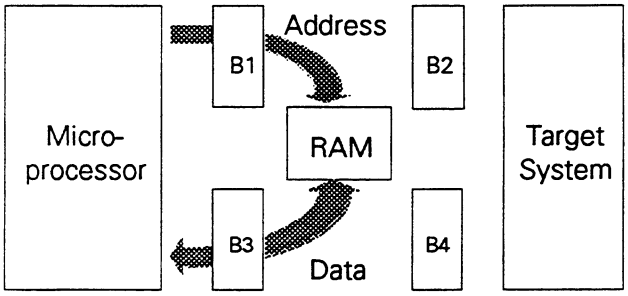

Figure 1b. Mode B:
access to the emulator's RAM

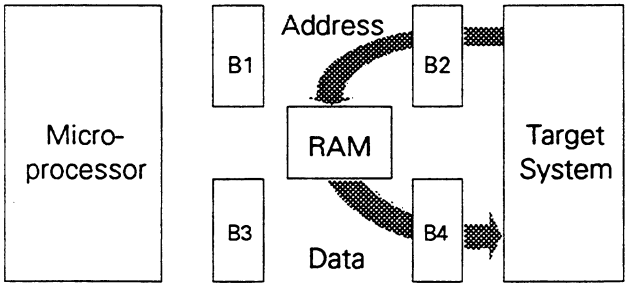

Figure 1c. Mode C:
gives the target system access to the RAM

There are three modes of operation:

a) Mode A allows the microprocessor to read the contents of an EPROM on the target system interface (this is most easily arranged by connecting the cable to the target system via a zero or low insertion force socket) by enabling buffers B1 and B2 to drive from left to right to supply addresses to the socket (the RAM also receives these addresses but its data outputs are disabled). Buffers B3 and B4 are enabled from right to left to return the data from the EPROM to the MC68HC05E0. The RAM is disabled via its chip-enable pin and so does not affect the data bus between buffers B3 and B4.

b) Mode B enables the buffers in such a way that the microprocessor can read from and write to the RAM. B1 is enabled to supply addresses to the RAM from the microprocessor (MC74HC245s were used throughout although a bidirectional buffer is not strictly necessary in this position as B1, if enabled, always drives from left to right). B3 is enabled to allow data to be written to or read from the RAM. The direction control for B3 is by the R/W signal from the MC68HC05E0 (gated with the RAM's chip enable). This mode is used during use of the memory modify facilities. Buffer B4 is disabled so that there is not a bus contention on either of its busses even if the target system is still connected. Buffer B2 is also disabled. The routine (L1) which loads RAM from an EPROM on the target system interface switches between modes A and B for each byte transferred.

c) In the emulation mode (C) the target system plugged into the socket is required to have access to the RAM so buffers B2 and B4 are enabled. B2 passes the addresses from right to left and B4 the data from left to right (as the emulation is for an EPROM, the target system is not allowed to write to the RAM). Buffers B1 and B3 are disabled.

Control line	MODE		
	A	B	C
4,PortB	1	1	0
5,PortB	0	0	1
6,PortB	0	1	0
7,PortB	1	0	0

CIRCUIT

Figure 2 shows the main circuit. An MC74HC138 is used to provide the chip enables. The emulator hardware is enabled in the address range $4000 – $7FFF and the EM64K program EPROM (27(C)64) at $C000 – $FFFF. If the EM64K program is contained in a 27(C)16 its pin 21 (Vpp) should be held high.

An additional socket is shown at address $8000 – $BFFF. This is for the optional LOAD2 facility which allows code to be loaded from EPROM without having to disconnect the cable to the target system. The emulation RAM occupies the address range $4000 – $7FFF. As this is only a 16K address space the RAM is paged. The four pages are selected by I/O lines (port B, bits 0 and 1) from the microprocessor. The memory map of the emulator is shown in figure 7.

The control lines (port B, bits 4–7) are biased by resistors. This holds the system in mode B if the MC68HC05E0 is held in reset and prevents bus contention resulting from an illegal combination of control signals. During hardware debug of the emulator it is advisable to use a current limited power supply (in the range 50–100 mA) as a bus contention can cause sufficiently high currents to damage the buffers.

The display is a 6-digit 4-backplane LCD (eg Hamlin type 4200 or the 8-digit GE type LXD69D3F09KG) which is driven by an MC145000 display driver. The driver is controlled by a 2-line serial link from the microprocessor. A single-backplane (or "static") display can be used as an alternative as shown in Figure 3. Three MC144115 driver chips are used. This circuit requires many more connections to the LCD but allows the use of a more readily available display. A third line (port B bit 3) from the microprocessor is used to supply the enable pins of the MC144115s. The single-backplane display drivers can be supplied directly from the main 5 volt supply but the multiplexed display requires a lower voltage. Figure 2 shows the MC145000 supplied via a 20k potentiometer which serves as a contrast control.

The keyboard uses an MC14028 decoder to minimise the number of I/O pins used. Note that port A bit 6 is used for both the keyboard and the display driver. The LOAD1 and LOAD2 keys overwrite the contents of emulation RAM and should thus not be pressed accidentally. It may therefore be useful for only those LOAD keys actually required to be fitted (usually LOAD1 and LOAD2 will not both be required) and for any parallel LOAD keys fitted to be placed away from the front panel or protected by requiring two keys, connected in series, to be pressed. An accidental press of the serial LOAD key can be aborted by pressing RESET.

As the circuit, except for the RS232 interface, is all CMOS the supply current is very low when the microprocessor is in STOP mode. This is a low power mode in which all processing, even the clock, is stopped. In the emulation mode (C) the MC68HC05E0 is in stop mode. In this mode and with no bus activity from the target system (or its interface open circuit) the supply current should be less than 1 µA (this does not include the current taken by the RS232 interface which, if present, can be switched off when not in use, or the 70–80 µA taken by the LCD driver).

It is worth checking that a low supply current is achieved as any excess can be a useful pointer to a wiring fault, particularly open circuit pins. The supply current may be affected by the choice of RAM but the MCM60L256 selected has a specified standby ICC of 2 µA and is typically well below this figure. In many applications the full 64K of RAM will not be required. If this is the case, only the required RAM need be included. 6116 2K RAMs could be used for 2–4K applications and MCM60L64s or equivalents for 8–16K. If using 6064s, their second chip enable pin (E2) should be held high. One MCM60L256 provides 32K. The serial load routine includes a read-back check on each byte sent to RAM so an attempt to write to non-existent RAM will generate an error message indicating the first faulty address. If 16K or less is required then the two 74HC245s handling addresses, A14 and A15, can also be omitted. These buffers have unused pins. The simplest way to ensure that no pins are left open circuit is to wire up the buffers in a manner similar to those actually used. Pins 2–7 of the left-hand buffers are held high while pins 13–18 are connected to the right-hand buffers whose other pins have pull-ups. This arrangement means that there will be no open circuits or bus contentions regardless of the levels of the control lines. If only one memory chip is used, the 74HC00 can be omitted (connect pin 3 of the 74HC32 directly to the RAM's chip enable).

The optional RS232 interface can most easily be implemented using the single-supply MC145407 driver-receiver chip. If outputting of S-records is not required then a simple transistor inverter with a pull-up resistor and a reverse polarity protection diode can be used. This interface is shown in Figure 4.

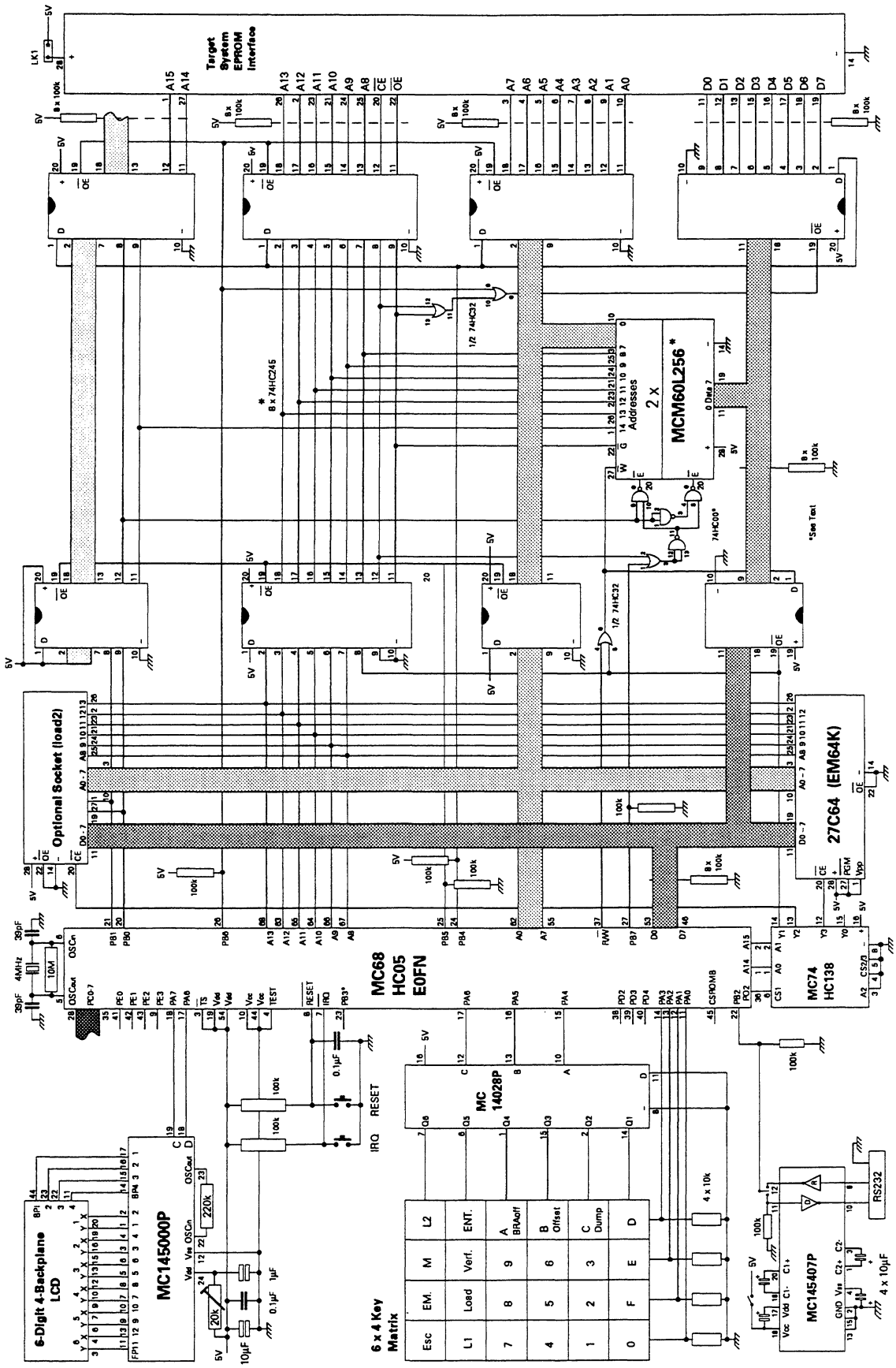

Figure 2. EPROM emulator circuit diagram

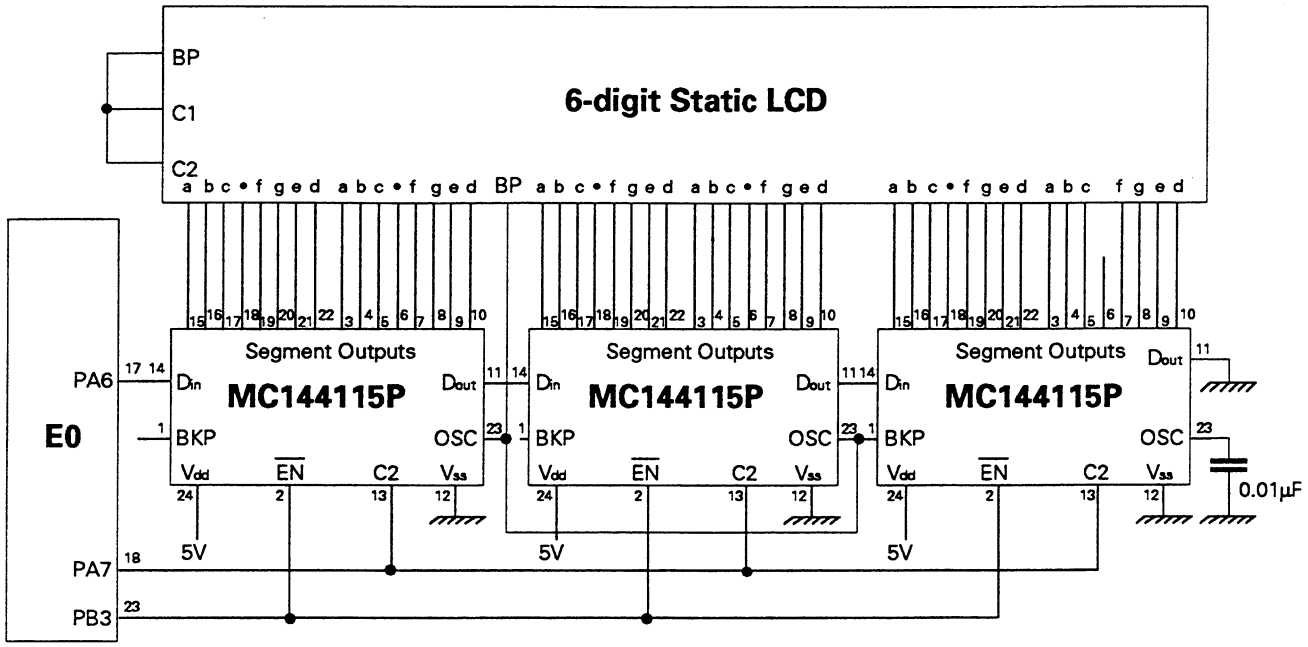

Figure 3. Alternative static LCD display

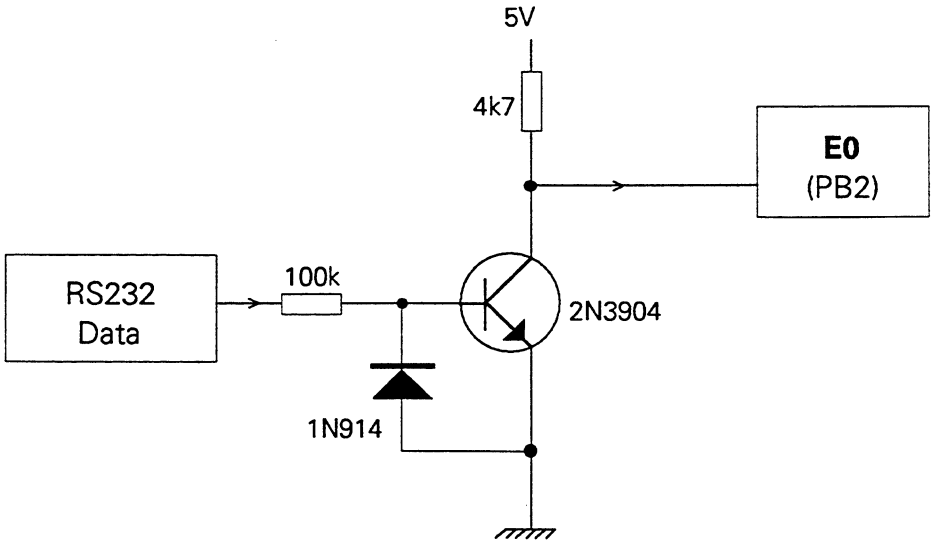

Figure 4. Simple input-only RS232 interface

IDD Monitor

It is often useful with CMOS circuits to provide a simple I_{DD} monitor which shows via an LED whether or not the I_{DD} is above or below a set value. In this application it shows whether or not the microprocessor is in the STOP mode. The required circuit is shown in Figure 5. The current threshold can be chosen by selecting the value of R1. A value of 1kΩ sets the limit at about 500 µA which means the LED should be off in the emulation mode but on otherwise. The 500 µA limit allows the LCD and perhaps an emulator-supplied CMOS target system to be supplied without switching on the LED. When the microprocessor is not in STOP (emulator not in mode C), its I_{DD} is several milliamps and the LED should be lit. In a battery application this circuit would also serve as a useful reminder that the RS232 interface has been left on. If a multiplexed LCD is used it may be preferable not to supply it via this type of monitor circuit as a significant change in contrast may occur when the microprocessor goes into its STOP mode (see Figure 5). The monitor drops about 600mV when the microprocessor is running so the supply voltage should be chosen accordingly; four zinc-carbon or five Ni-Cad cells were found to be satisfactory.

Address trap

The emulator allows memory locations to be examined and changed, but does not provide the breakpoint and trace features normally found in development systems. A limited capability can be made available if address comparators of the type shown in Figure 6 are added. This circuit gives an LED indication if the address selected on the bank of switches is encountered by the program running in the target system. An address coincidence is latched by the 74HC74. To indicate the occurrence of a repetitive event, a one-shot chip could be added.

SERIAL LOAD

To load external Motorola S-records the serial load key (LOAD) should be pressed. The LCD will display "LOAd". S-records should then be supplied at 9600 baud (8-bit, no parity) on the RS232 interface. When an S9 termination record is received, the prompt returns. If an error is detected during a serial load, the load routine stops and displays the address at which the error occurred and the error type.

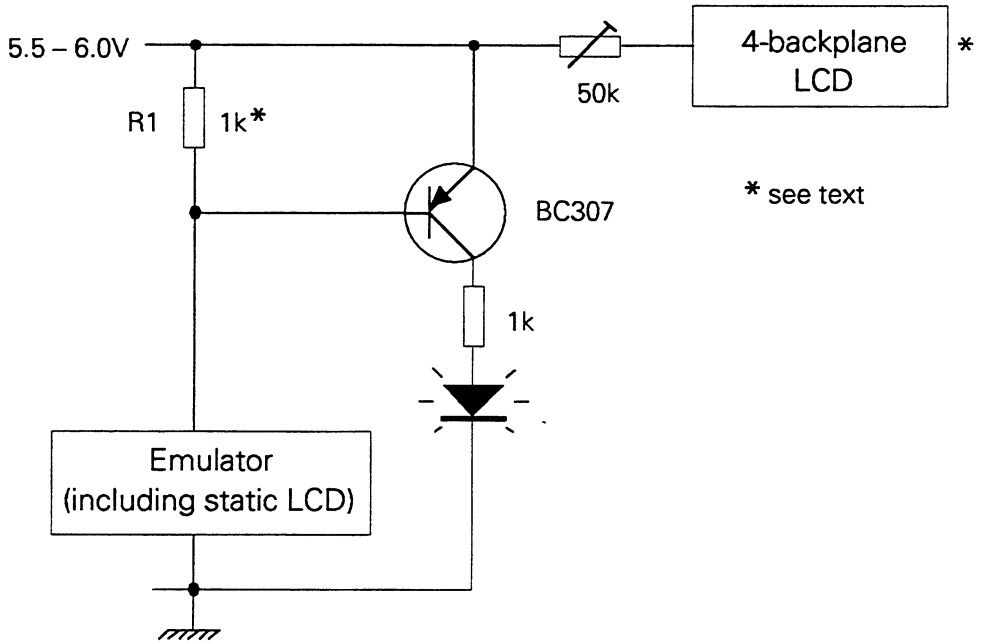

Figure 5. Simple IDD monitor

The following error types are possible:

1: Checksum error, transmitted data or interface faulty.

2: RAM read-back error, RAM faulty or non-existent.

3: ASCII character less than $30 (0) received.

4: ASCII character between $39 (9) and $41 (A) received.

5: ASCII character more than $46 (F) received.

7: Verify error when comparing S-records with emulation RAM.

If, when using the emulator, the target system ceases to function properly, then the verify function can be used to check that the emulation RAM has not been corrupted. The VERIFY function is used exactly like LOAD except that RAM is compared with, rather than loaded by, the S-records.

The address at the start of each S1-record determines the address at which the code will reside in the target system. This address will sometimes be different from that at which the code is required to be loaded into the emulation RAM so an offset may need to be used. The offset byte is entered using the appropriate key and allows an offset of any multiple of $0100. The offset is subtracted from the MSB of the S-record address and this modified address is the physical address at which the data is loaded into the emulation RAM. The S-record output routine adds the offset before transmitting the records. At reset or power-up the offset is initialised to zero.

All addresses entered while using the MEMORY-MODIFY, BRAOFF and DUMP routine use the actual address in the target system. These addresses will only be the same as the physical RAM address if the offset is zero.

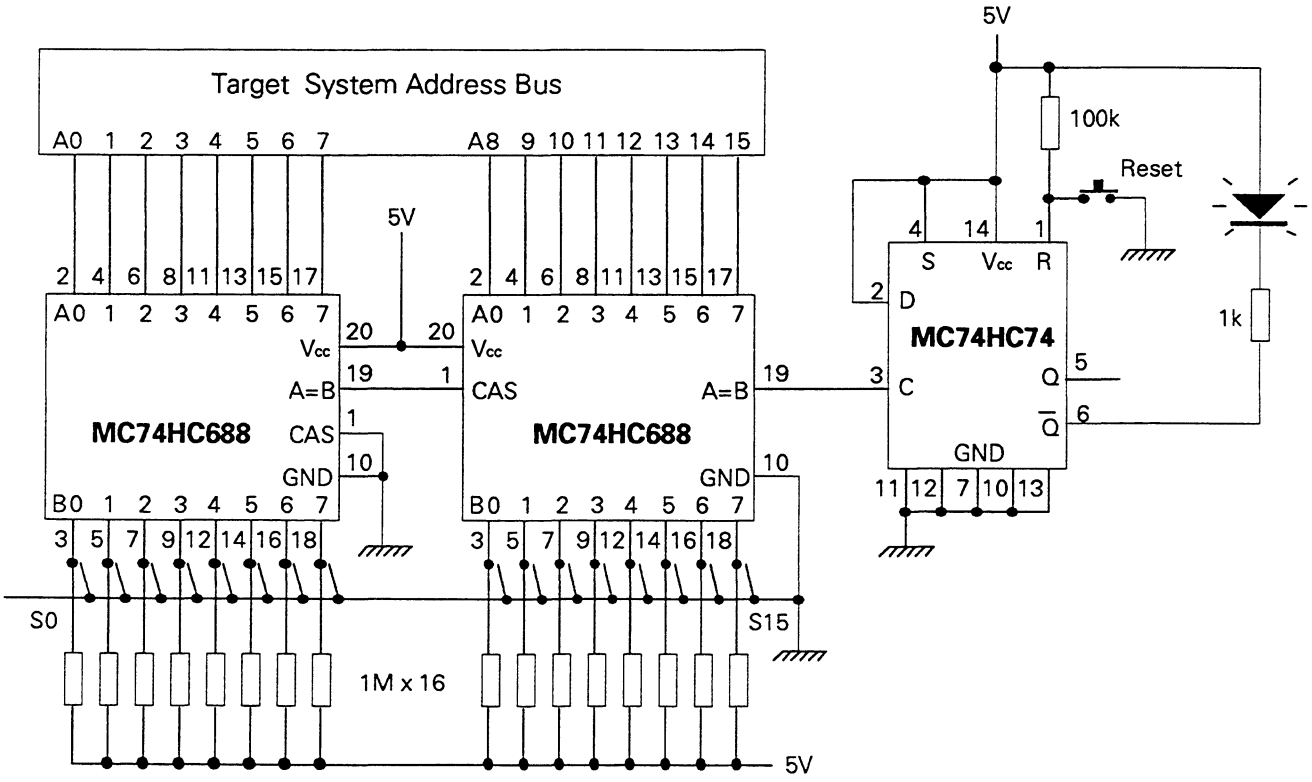

Figure 6. Address Trap

PARALLEL LOAD

When the parallel load functions (LOAD1 and LOAD2) are used, OFFSET has no effect on the transfer of code into RAM. It can, however, still be used to offset the RAM addresses to correspond with the actual address in the target system program when using the memory-modify facilities.

27(C)64/128/256 EMULATION

When emulating a 27(C)512 EPROM, all the RAM is used. For emulation of smaller EPROMS, less RAM is required. The memory used will be at the beginning of RAM (starting at address zero) only if the unused high-order address lines are held low. It may, however, be more convenient to allow one or more of these addresses to be high. The pull-ups included in Figure 2 will hold any uncommitted lines high. For example, a 27(C)64 can be emulated with no hardware change as long as the code is loaded between $E000 and $FFFF. This will often be appropriate as it allows the vectors at the top of the target system's memory to be included. It makes little difference if the target microprocessor has an address space smaller than 64K as the high order addresses will not be present and will be held high. Clearly, the code must still be assembled at the appropriate addresses and the emulator's offset feature used to load the S-records between $E000 and $FFFF. Alternatively, the S-records can be loaded lower in the emulator's address space and the relevant high-order addresses held low. When loading from a smaller EPROM, with a VPP or PRG pin, these pins should be configured correctly for reading (high) and not driven by the emulator. See Figure 9 for the industry-standard EPROM pinouts. As the software does not behave differently for smaller EPROMs, a full 64K transfer will still be made, copying the EPROM several times into the 64K RAM. The actual copy used depends on the levels of the high order addresses as outlined above.

TARGET SYSTEM INTERFACE

Vdd can be connected to the interface by the link shown. The simplest method of use is to make this connection and to use a common supply for the emulator and the target system. If, however, separate supplies are used, then pin 28 should not be connected. If separate supplies are used, care should be taken that they do not differ by more than 0.5V. A delta greater than this may cause a malfunction as a result of the logic level on an input pin being in excess of the chip's Vdd.

In emulation mode the target system has total control of the RAM except for its R/W line. It can thus use the RAM exactly as if it were a ROM or EPROM. Before IRQ (or RESET) is pressed to exit from the emulation mode the target system should be stopped so that it no longer expects the "EPROM" to be there. This will normally be done by holding the target system in reset. If the target system is an M68(HC)05 (eg MC68HC05E0 or MC146805E2) or M68HC11, then it can alternatively be put into its STOP mode. If this is its normal idle condition, then nothing need be done prior to exiting emulation.

EM64K PROGRAM

The EM64K control program is less than 2K bytes long and can thus reside in a 27C64 or 27C16. The circuit is shown for a 27C64 and assumes that the program starts at the beginning of the EPROM. This EPROM is enabled at $C000 (and $E000 as A13 is not used). An assembled listing of the control program is included at the end of this application note.

EM64K KEY FUNCTIONS

Function	KEY	Description of function
LOAD1	L1	Load RAM from target system interface ($4000–$7FFF).
LOAD2	L2	Load RAM from secondary socket ($8000–$BFFF).
SERIAL LOAD	LOAD	Load emulation RAM with S-records via the RS232 interface, during loading LCD shows "LOAd".
VERIFY	Verf	Compare emulation RAM with S-records via the RS232 interface, LCD displays "UErIFy".
EMULATE	EM	Emulator mode. Prompts: "EP ?" for the removal of an EPROM (if present) and connection of the target system (press again if OK) and put micro into EMULATE mode.
MEMORY MODIFY	M	Display/change a RAM location. When pressed the last address is displayed. Press ENTER to display the contents of this address or input a new address followed by ENTER. To change, input new data followed by ENTER. ENTER moves to next address, M moves to previous address, ESCAPE exits.
ENTER	Ent	Enter keyed-in address or data (and move to next address in MEMORY MODIFY).
ESCAPE	Esc	Exit from current function (OFFSET, BRAOFF, DUMP or MEMORY MODIFY).
BRAOFF	A	Calculate branch offset. The address of the branch instruction and of the destination are requested. If a valid branch is calculated it is written into memory and displayed. If not valid then "or" for out of range is displayed. A branch of -128 through +127 relative to the start address of the next instruction is allowed. Esc returns to the normal prompt.
OFFSET	B	Allows entry of an offset to the emulation RAM address. It is subtracted from the most significant byte of the address specified by the incoming S-records. The offset is added to the address by the DUMP function.
DUMP	C	Output emulation RAM contents as S-records via RS232 interface. RAM start and finish addresses are requested. They should be entered followed by ENTER. After the second ENTER, the S-record output starts.
IRQ	IRQ	Abort emulation and return to emulator monitor.
RESET	RESET	Resets emulator, displays prompt (□). Should be used after power-up or if the emulator malfunctions. Can be used instead of IRQ, with the difference that the OFFSET is reset to zero. RESET provides the only exit from a LOAD or VERIFY which has not been terminated correctly by the reception of an S9 record.

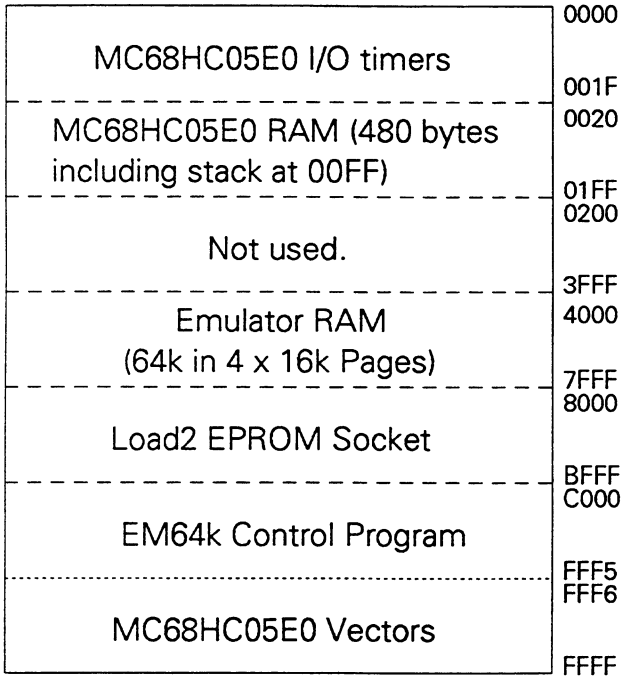

Figure 7. Memory Map

SOFTWARE

A listing of the control program used in the emulator is included in this application note. Some points specific to the MC68HC05E0 are discussed below.

Port D on the MC68HC05E0 can be used as a normal I/O port or can selectively supply special signals. In this application five of the special function are used. These function are selected using the register at address $12. The function used are P02, R/W, A13, A14 and A15. By default only addresses A0 through A12 are available as this will be sufficient in many applications. In this application, however, all the addresses are required. The clock (P02) is used to qualify the chip selects generated by the MC74HC138 and R/W for control of the emulation RAM. The other three pins are left as I/O pins but are not used in this application. The initialisation of $12 can be seen on lines 93 and 94 of the software listing.

The only other register used (apart from the I/O data and DDR registers) is the interrupt control register ($0E). It is written to $01 on lines 82 and 83 of the listing. This operation clears the interrupt flag (bit 3) but keeps the INTMX bit set. This bit enables external interrupts. The registers associated with unused on-chip resources are left at their reset conditions. An important bit in the MC68HC05E0 is the XROM bit (2,$0C). It defaults to a 1 which is appropriate in this application. When it is cleared it constrains the data bus to be input only thus preventing any unnecessary activity in sensitive applications when writing to external memory is not required.

The MC68HC05E0 has the 8-bit index register common to all M6805 microprocessors. It is thus not able to contain a 16-bit extended address. For this reason, loading and storing in the emulator's RAM is carried out using a small program in the micro's RAM. This program consists of an extended LDA or STA instruction followed by a two byte address which can be built in software and an RTS instruction. The four-byte program resides in RAM at locations W2, ADDEH, ADDRL and W3. It allows the full 64K map to be accessed using addresses generated within the program. Address generation is further complicated by the requirement that the emulation RAM is in four 16K pages. The two most significant addresses thus have to be transferred to port lines PB0 and PB1.

SERIAL INTERFACE

Figure 8 shows a suggested method of wiring up the RS232 sockets in an emulator with both loading and dumping capabilities. This arrangement facilitates use of the serial LOAD and DUMP routines of the emulator either via a PC COM port or between a host and terminal connected by an RS232 link. When using a PC the "host" socket should be used. As only one pin on the MC68HC05E0 is used, switching is required to make the required connections. S2 can be eliminated (or left at "L") if only loading is required, as will often be the case. To save power in battery applications, the RS232 interface chip can be switched off using S1. The following table shows possible methods of use.

Set-up	Function	S1	S2	Comments
Host & terminal	Load	On	L	Terminal and host connected. Micro looks at data sent from host to terminal (pins 3).
	Dump	On	D	Connection between terminal and host broken. S-records sent to both host (2) and terminal (3).
PC "COM" port	Load	On	L	S-records loaded from pin 3.
	Dump	On	D	S-records sent to pin 2 on "host" socket (and pin 3 on "terminal" socket).

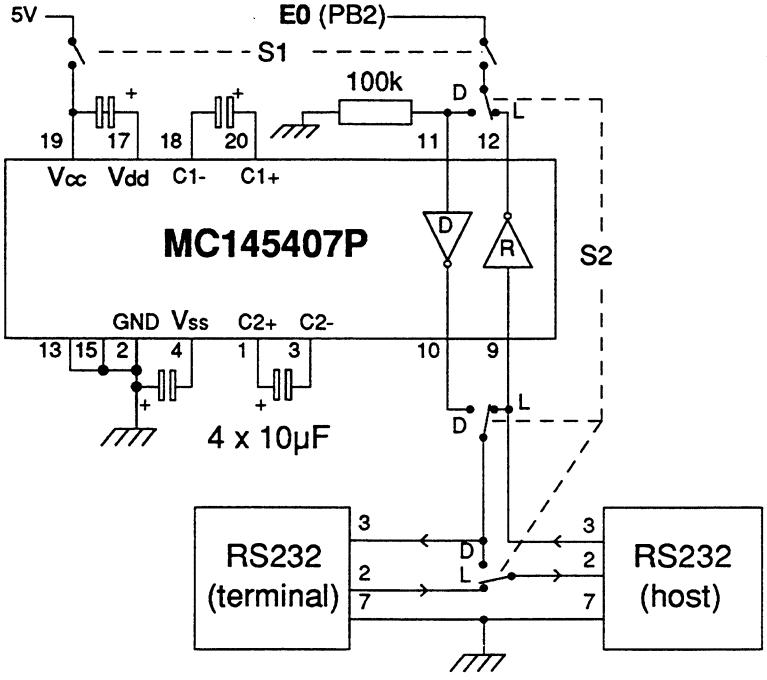

Figure 8. RS232 circuit with LOAD/DUMP switching

Pin	27512	27256	27128	2764
1	A15	Vpp	Vpp	Vpp
2	A12	.	.	.
3	A7	.	.	.
4	A6	.	.	.
5	A5	.	.	.
6	A4	.	.	.
7	A3	.	.	.
8	A2	.	.	.
9	A1	.	.	.
10	A0	.	.	.
11	D0	.	.	.
12	D1	.	.	.
13	D2	.	.	.
14	Vss	.	.	.
15	D3	.	.	.
16	D4	.	.	.
17	D5	.	.	.
18	D6	.	.	.
19	D7	.	.	.
20	Chip enable	.	.	.
21	A10	.	.	.
22	Output enable	.	.	.
23	A11	.	.	.
24	A9	.	.	.
25	A8	.	.	.
26	A13	A13	A13	NC
27	A14	A14	PGM	PGM
28	Vcc	.	.	.

Figure 9. 27(C) 512, 256, 128 and 64 pin-outs
(Table shows the standard 28-pin EPROMs.
Blank entries indicate that the pin is the same as for the 27(C) 512.)

ASSEMBLED LISTING OF THE EM64K CONTROL PROGRAM

```
 1                          ****************************************************
 2                          *                                                  *
 3                          *            MC68HC05E0 EPROM Emulator.            *
 4                          *                                                  *
 5                          *      A 68HC05E0 is used to emulate an  EPROM     *
 6                          *      of up to 64K (27(C)512) with SRAM which     *
 7                          *      can be loaded from an EPROM or serially     *
 8                          *      by S-records via an RS232 interface and     *
 9                          *      changed if required for de-bug etc.         *
10                          *                                                  *
11                          *      P. Topping                  11-Jan-91       *
12                          *                                                  *
13                          ****************************************************
14
15 00000000                 PORTA   EQU     $00             PORT A ADDRESS
16 00000001                 PORTB   EQU     $01               "  B    "
17 00000002                 PORTC   EQU     $02               "  C    "
18 00000003                 PORTD   EQU     $03               "  D    "
19 00000004                 PORTE   EQU     $04               "  E    "
20 00000005                 PORTAD  EQU     $05             PORT A DATA DIRECTION REG.
21 00000006                 PORTBD  EQU     $06               "  B  "       "      "
22 00000007                 PORTCD  EQU     $07               "  C  "       "      "
23 00000008                 PORTDD  EQU     $08               "  D  "       "      "
24 00000009                 PORTED  EQU     $09               "  E  "       "      "
25 0000000e                 ICR     EQU     $0E             INTERRUPT CONTROL REGISTER
26 00000012                 PORTDSF EQU     $12             PORTD ALTERNATIVE FUNCTION REGISTER
27
28                                  ORG     $0020           RAM ALLOCATION
29
30 00000020                 DTABL   RMB     6               LCD BUFFER
31 00000026                 TEMP    RMB     2
32 00000028                 W1      RMB     1
33 00000029                 W2      RMB     1               RAM SUBROUTINE  LDA or STA
34 0000002a                 ADDEH   RMB     1                "      "       ADDRESS MSB
35 0000002b                 ADDRL   RMB     1                "      "          "    LSB
36 0000002c                 W3      RMB     1                "      "       RTS
37 0000002d                 W4      RMB     1
38 0000002e                 W5      RMB     1
39 0000002f                 W6      RMB     1
40 00000030                 ADDRH   RMB     1
41 00000031                 STAT    RMB     1               STATUS BYTE :-
42                          *                               2: REAL ADDRESS (OFFSET)
43                          *                               4: INVALID ADDRESS (BLDRNG)
44                          *                               5: VERYFING (TLOAD)
45                          *                               6: INDIVIDUAL REG. (MEMEX)
46                          *                               7: PUNCH END
47 00000032                 CHKSUM  RMB     1               CHECKSUM
48 00000033                 COUNT   RMB     1               BIT COUNTER
49 00000034                 TMP1    RMB     1
50 00000035                 TMP2    RMB     1
51 00000036                 BCNT    RMB     1               S-RECORD BYTE COUNT
52 00000037                 ERTYP   RMB     1               ERROR TYPE
53 00000038                 ERDAT   RMB     1               ERROR DATA
54 00000039                 OFF     RMB     1               S-RECORD OFFSET
55 0000003a                         RMB     185             UNUSED
56 000000f3                 STACK   RMB     12              13 BYTES USED (1 INTERRUPT
57 000000ff                 SP      RMB     1               AND 4 NESTED SUBROUTINES)
```

em64k.as5

```
                                ****************************************************
                                *                                                  *
                                *       Idle loop and routine to decide which      *
                                *       key has been pressed.                      *
                                *                                                  *
                                ****************************************************

                                        ORG     $E000

0000e000  cde05f        SCAN    JSR     KEYSCN          KEY FOUND ?
0000e003  24fb                  BCC     SCAN            NO, TRY AGAIN
0000e005  5f                    CLRX
0000e006  b728                  STA     W1              CODE OF PRESSED KEY
0000e008  d6e09f        RJ      LDA     CTAB,X          FETCH KEYCODE
0000e00b  b128                  CMP     W1              THIS ONE ?
0000e00d  270b                  BEQ     PJ              YES
0000e00f  c1e0bf                CMP     LAST            NO, LAST CHANCE ?
0000e012  273a                  BEQ     GETCMD          YES, ABORT
0000e014  5c                    INCX                    NO
0000e015  5c                    INCX                    TRY
0000e016  5c                    INCX                    THE
0000e017  5c                    INCX                    NEXT
0000e018  20ee                  BRA     RJ              KEY
0000e01a  a601          PJ      LDA     #1
0000e01c  b70e                  STA     ICR             CLEAR IRQX FLAG
0000e01e  5c                    INCX
0000e01f  dce09f                JMP     CTAB,X
```

em64k.as5

```
                                ****************************************************
                                *                                                  *
                                *       Reset routine.                             *
                                *                                                  *
                                ****************************************************

0000e022  a6e3          START   LDA     #$E3            ENABLE PORTD SPECIAL FUNCTIONS
0000e024  b712                  STA     PORTDSF         P02, R/W, A13, A14 & A15

0000e026  3f00                  CLR     PORTA
0000e028  a6f0                  LDA     #$F0            DISPLAY/KEYBOARD
0000e02a  b705                  STA     PORTAD          I/O

0000e02c  a658                  LDA     #$58            MODE 2, ENABLE (144115) HIGH
0000e02e  b701                  STA     PORTB
0000e030  a6fb                  LDA     #$FB            BITS 0, 1, 3-7 OUTPUTS
0000e032  b706                  STA     PORTBD          BIT 2 INPUT

0000e034  3f02                  CLR     PORTC
0000e036  a6ff                  LDA     #$FF            ALL OUT, NOT USED
0000e038  b707                  STA     PORTCD

0000e03a  3f03                  CLR     PORTD
0000e03c  a61c                  LDA     #$1C            BITS 2, 3 & 4 OUT, NOT USED
0000e03e  b708                  STA     PORTDD

0000e040  3f04                  CLR     PORTE
0000e042  a60f                  LDA     #$0F            BITS 0 - 3 OUT, NOT USED
0000e044  b709                  STA     PORTED

0000e046  3f31                  CLR     STAT
0000e048  3f2b                  CLR     ADDRL           INITIALISE
0000e04a  3f30                  CLR     ADDRH           ADDRESS
0000e04c  3f39                  CLR     OFF

0000e04e  a658          GETCMD  LDA     #$58            MODE 2, ENABLE (144115) HIGH
```

```
    123 0000e050  b701                    STA     PORTB
    124 0000e052  cde235                  JSR     CLRTAB
    125 0000e055  a633                    LDA     #$33            PRINT
    126 0000e057  b720                    STA     DTABL           PROMPT
    127 0000e059  cde1ec            DSCN  JSR     DISTAB
    128 0000e05c  9c                      RSP
    129 0000e05d  20a1                    BRA     SCAN
```

em64k.as5

```
    131                           ***************************************************
    132                           *                                                 *
    133                           *       The  keyboard routine  returns the code   *
    134                           *       of the pressed key in the accumulator.    *
    135                           *                                                 *
    136                           ***************************************************
    137
    138 0000e05f  4f              KEYSCN  CLRA
    139 0000e060  ae06                    LDX     #6              SETUP
    140 0000e062  ab10            KEY1    ADD     #$10            ROW
    141 0000e064  b700                    STA     PORTA
    142 0000e066  b600            COLUMN  LDA     PORTA           READ KEYBOARD
    143 0000e068  b72c                    STA     W3              STORE IT
    144 0000e06a  a50f                    BIT     #$0F            KEY CLOSED ?
    145 0000e06c  2719                    BEQ     COLRET          NO GET OUT
    146 0000e06e  ad1d                    BSR     DBOUNC          ELSE DEBOUNCE
    147 0000e070  b600                    LDA     PORTA           RE-READ KEYPAD
    148 0000e072  b12c                    CMP     W3              SAME KEY CLOSED ?
    149 0000e074  2611                    BNE     COLRET          NO, GET OUT
    150 0000e076  99                      SEC
    151 0000e077  b600            COL1    LDA     PORTA           KEY
    152 0000e079  a50f                    BIT     #$0F            RELEASED ?
    153 0000e07b  26fa                    BNE     COL1            NO TRY AGAIN
    154 0000e07d  ad0e                    BSR     DBOUNC          YES DEBOUNCE
    155 0000e07f  b600                    LDA     PORTA           STILL
    156 0000e081  a50f                    BIT     #$0F            RELEASED ?
    157 0000e083  26f2                    BNE     COL1            NO TRY AGAIN
    158 0000e085  b62c                    LDA     W3              RETURN CHAR IN A-REG
    159 0000e087  2503            COLRET  BCS     KEY2            IF VALID GET OUT
    160 0000e089  5a                      DECX                    ELSE TRY
    161 0000e08a  26d6                    BNE     KEY1            NEXT ROW
    162 0000e08c  81              KEY2    RTS
    163
    164 0000e08d  a60a            DBOUNC  LDA     #10             40mS
    165 0000e08f  b72f                    STA     W6
    166 0000e091  a6ff            DLP     LDA     #$FF            PAUSE
    167 0000e093  21fe            DLOOP   BRN     *               256X12
    168 0000e095  21fe                    BRN     *               CYCLES
    169 0000e097  4a                      DECA
    170 0000e098  26f9                    BNE     DLOOP
    171 0000e09a  3a2f                    DEC     W6
    172 0000e09c  26f3                    BNE     DLP
    173 0000e09e  81                      RTS
```

em64k.as5

```
175                             ***************************************************
176                             *                                                 *
177                             *       Keyboard tables.                          *
178                             *                                                 *
179                             ***************************************************
180
181 0000e09f  51                CTAB    FCB     $51         P    LOAD FROM EPROM ($4000)
182 0000e0a0  cce27b                    JMP     DUMP1
183 0000e0a3  68                        FCB     $68         S    LOAD FROM EPROM ($8000)
184 0000e0a4  cce2db                    JMP     DUMP9
185 0000e0a7  28                        FCB     $28         C    S-RECORD OUTPUT
186 0000e0a8  cce3f8                    JMP     PUNCH
187 0000e0ab  52                        FCB     $52         L    LOAD S-RECORD
188 0000e0ac  cce321                    JMP     TLOAD
189 0000e0af  54                        FCB     $54         V    VERIFY (S-RECORD)
190 0000e0b0  cce31b                    JMP     VERIFY
191 0000e0b3  62                        FCB     $62         G    GO INTO EMULATOR MODE
192 0000e0b4  cce23d                    JMP     MODE3
193 0000e0b7  64                        FCB     $64         M    READ/CHANGE MEMORY
194 0000e0b8  cce4fd                    JMP     MEMEX
195 0000e0bb  38                        FCB     $38         B    ADDRESS OFFSET
196 0000e0bc  cce2c0                    JMP     OFFSET
197 0000e0bf  48                LAST    FCB     $48         A    BRANCH OFFSET CALC.
198 0000e0c0  cce13e                    JMP     BRAOFF
199
200 0000e0c3  11                STABL   FCB     $11         0
201 0000e0c4  21                        FCB     $21         1
202 0000e0c5  22                        FCB     $22         2
203 0000e0c6  24                        FCB     $24         3
204 0000e0c7  31                        FCB     $31         4
205 0000e0c8  32                        FCB     $32         5
206 0000e0c9  34                        FCB     $34         6
207 0000e0ca  41                        FCB     $41         7
208 0000e0cb  42                        FCB     $42         8
209 0000e0cc  44                        FCB     $44         9
210 0000e0cd  48                        FCB     $48         A    BRANCH OFFSET
211 0000e0ce  38                        FCB     $38         B    LOAD OFFSET
212 0000e0cf  28                        FCB     $28         C    OUTPUT S-RECORDS
213 0000e0d0  18                        FCB     $18         D
214 0000e0d1  14                        FCB     $14         E
215 0000e0d2  12                        FCB     $12         F
216 0000e0d3  61                        FCB     $61        10    Esc       CANCEL COMMAND
217 0000e0d4  58                        FCB     $58        11    E         ENTER COMMAND
218 0000e0d5  68                        FCB     $68        12    S         LOAD FROM $8000
219 0000e0d6  64                        FCB     $64        13    M         MEMORY EXAMINE/CHANGE
220 0000e0d7  62                        FCB     $62        14    G         EMULATE
221 0000e0d8  54                        FCB     $54        15    V         VERIFY RAM
222 0000e0d9  52                        FCB     $52        16    L         LOAD RAM
223 0000e0da  51                        FCB     $51        17    P         LOAD FROM $4000
```

em64k.as5

```
225                                     ***************************************************
226                                     *                                                 *
227                                     *       Build a beginning and ending address      *
228                                     *       in TEMP,TEMP+1 & ADDRH,ADDRL resp.        *
229                                     *                                                 *
230                                     ***************************************************
231
232 0000e0db  1931                    BLDRNG  BCLR    4,STAT
233 0000e0dd  1531                            BCLR    2,STAT          EMULATION ADDRESS
234 0000e0df  cde235                          JSR     CLRTAB          PRINT
235 0000e0e2  a6f4                            LDA     #$F4            'BA'
236 0000e0e4  b724                            STA     DTABL+4
237 0000e0e6  a677                            LDA     #$77
238 0000e0e8  b725                            STA     DTABL+5
239 0000e0ea  cde1ec                          JSR     DISTAB
240 0000e0ed  cde5be                          JSR     BLDADR          GET SOURCE ADDR.
241 0000e0f0  2423                            BCC     BLDRN1          VALID?
242 0000e0f2  b630                            LDA     ADDRH           YES
243 0000e0f4  b726                            STA     TEMP            SAVE IT
244 0000e0f6  b62b                            LDA     ADDRL
245 0000e0f8  b727                            STA     TEMP+1
246 0000e0fa  cde570                          JSR     LOAD            FETCH OPCODE OF INSTR.
247 0000e0fd  b72f                            STA     W6              SAVE IT
248 0000e0ff  cde235                          JSR     CLRTAB
249 0000e102  a6f1                            LDA     #$F1            PRINT 'EA'
250 0000e104  b724                            STA     DTABL+4
251 0000e106  a677                            LDA     #$77
252 0000e108  b725                            STA     DTABL+5
253 0000e10a  cde1ec                          JSR     DISTAB
254 0000e10d  cde5be                          JSR     BLDADR          GET DESTINATION ADDR
255 0000e110  2403                            BCC     BLDRN1          VALID?
256 0000e112  b630                            LDA     ADDRH           YES
257 0000e114  81                              RTS
258 0000e115  1831                    BLDRN1  BSET    4,STAT          INVALID
259 0000e117  81                              RTS
260
261                                     ***************************************************
262                                     *                                                 *
263                                     *       Display message.                          *
264                                     *                                                 *
265                                     ***************************************************
266
267 0000e118  bf2f                    DISP    STX     W6
268 0000e11a  3f33                            CLR     COUNT
269 0000e11c  be2f                    DISLP   LDX     W6
270 0000e11e  d6e132                          LDA     DLOAD,X
271 0000e121  be33                            LDX     COUNT
272 0000e123  e720                            STA     DTABL,X
273 0000e125  3c2f                            INC     W6
274 0000e127  3c33                            INC     COUNT
275 0000e129  b633                            LDA     COUNT
276 0000e12b  a106                            CMP     #6
277 0000e12d  25ed                            BLO     DISLP
278 0000e12f  cce1ec                          JMP     DISTAB
279
280 0000e132  0000d0d777e6            DLOAD   FCB     0,0,$D0,$D7,$77,$E6
281 0000e138  d6f1600671b6            VERF    FCB     $D6,$F1,$60,$06,$71,$B6
```

em64k.as5

```
    283                         ****************************************************
    284                         *                                                  *
    285                         *       Calculate branch offset.                   *
    286                         *                                                  *
    287                         ****************************************************
    288
    289 0000e13e  ad9b          BRAOFF  BSR     BLDRNG
    290 0000e140  08313e                BRSET   4,STAT,ORET
    291 0000e143  b62b                  LDA     ADDRL           NO FIND APPARENT
    292 0000e145  a002                  SUB     #2
    293 0000e147  b72b                  STA     ADDRL
    294 0000e149  b630                  LDA     ADDRH
    295 0000e14b  a200                  SBC     #0
    296 0000e14d  b730                  STA     ADDRH
    297 0000e14f  b62b                  LDA     ADDRL
    298 0000e151  b027                  SUB     TEMP+1          OFFSET
    299 0000e153  b72b                  STA     ADDRL
    300 0000e155  b630                  LDA     ADDRH
    301 0000e157  b226                  SBC     TEMP
    302 0000e159  b730                  STA     ADDRH
    303 0000e15b  b62f                  LDA     W6              CHECK OPCODE
    304 0000e15d  a11f                  CMP     #$1F            FOR BIT BRANCH
    305 0000e15f  234e                  BLS     OFFST1
    306 0000e161  b630                  LDA     ADDRH
    307 0000e163  a1ff                  CMP     #$FF            + OR - OFFSET?
    308 0000e165  270b                  BEQ     OFFST2
    309 0000e167  4d                    TSTA                    CHECK OFFSET
    310 0000e168  2674                  BNE     OVRERR          FOR +/- 0
    311 0000e16a  b62b                  LDA     ADDRL
    312 0000e16c  a17f                  CMP     #$7F
    313 0000e16e  226e                  BHI     OVRERR
    314 0000e170  200a                  BRA     OK1
    315
    316 0000e172  b62b          OFFST2  LDA     ADDRL
    317
    318 0000e174  a1ff                  CMP     #$FF
    319 0000e176  2766                  BEQ     OVRERR
    320
    321 0000e178  a180                  CMP     #$80
    322 0000e17a  2562                  BLO     OVRERR
    323 0000e17c  ad06          OK1     BSR     USE             PRINT IT IF VALID
    324 0000e17e  cce000                JMP     SCAN
    325
    326 0000e181  cce04e        ORET    JMP     GETCMD

em64k.as5

    328
    329 0000e184  cde235        USE     JSR     CLRTAB
    330 0000e187  a6d6                  LDA     #$D6            PRINT 'USED'
    331 0000e189  b720                  STA     DTABL
    332 0000e18b  a6b5                  LDA     #$B5
    333 0000e18d  b721                  STA     DTABL+1
    334 0000e18f  a6f1                  LDA     #$F1
    335 0000e191  b722                  STA     DTABL+2
    336 0000e193  a6e6                  LDA     #$E6
    337 0000e195  b723                  STA     DTABL+3
    338 0000e197  b62b                  LDA     ADDRL           PRINT OFFSET
    339 0000e199  cde5f2                JSR     PRTDAT
    340 0000e19c  97                    TAX
    341 0000e19d  b627                  LDA     TEMP+1
    342 0000e19f  ab01                  ADD     #1
    343 0000e1a1  b72b                  STA     ADDRL
    344 0000e1a3  b626                  LDA     TEMP
    345 0000e1a5  a900                  ADC     #0              PUT INTO
    346 0000e1a7  b730                  STA     ADDRH           INSTRUCTION
    347 0000e1a9  9f                    TXA
    348 0000e1aa  1531                  BCLR    2,STAT
```

```
 349 0000e1ac  cce562                    JMP     STORE
 350
 351 0000e1af  b62b              OFFST1  LDA     ADDRL           ADJUST FOR
 352 0000e1b1  a001                      SUB     #1              BIT BRANCH
 353 0000e1b3  b72b                      STA     ADDRL
 354 0000e1b5  b630                      LDA     ADDRH
 355 0000e1b7  a200                      SBC     #0
 356 0000e1b9  b730                      STA     ADDRH
 357 0000e1bb  a1ff                      CMP     #$FF            NEG OFFSET?
 358 0000e1bd  270b                      BEQ     OFFST3          YES
 359 0000e1bf  4d                        TSTA    CHECK FOR
 360 0000e1c0  261c                      BNE     OVRERR          +/- 0 AND -1
 361 0000e1c2  b62b                      LDA     ADDRL
 362 0000e1c4  a17f                      CMP     #$7F
 363 0000e1c6  2216                      BHI     OVRERR
 364 0000e1c8  200a                      BRA     OK2
 365
 366 0000e1ca  b62b              OFFST3  LDA     ADDRL
 367 0000e1cc  a1fe                      CMP     #$FE
 368 0000e1ce  240e                      BHS     OVRERR
 369 0000e1d0  a180                      CMP     #$80
 370 0000e1d2  250a                      BLO     OVRERR
 371
 372 0000e1d4  3c27              OK2     INC     TEMP+1
 373 0000e1d6  2602                      BNE     OFFITS
 374 0000e1d8  3c26                      INC     TEMP
 375 0000e1da  ada8              OFFITS  BSR     USE             PRINT IF VALID
 376 0000e1dc  200b                      BRA     SCJMP
 377
 378 0000e1de  a6d7              OVRERR  LDA     #$D7            PRINT "OR"
 379 0000e1e0  b724                      STA     DTABL+4
 380 0000e1e2  a660                      LDA     #$60
 381 0000e1e4  b725                      STA     DTABL+5
 382 0000e1e6  cde616                    JSR     PRTADR
 383 0000e1e9  cce000            SCJMP   JMP     SCAN
```

em64k.as5

```
 385                             ***************************************************
 386                             *                                                 *
 387                             *       Display table contents.                   *
 388                             *                                                 *
 389                             ***************************************************
 390
 391 0000e1ec  1701              DISTAB  BCLR    3,PORTB         ENABLE (144115) LOW
 392 0000e1ee  ae05                      LDX     #5
 393 0000e1f0  e620              DISCHR  LDA     DTABL,X         LOAD DISPLAY
 394
 395 0000e1f2  bf28              NT1     STX     W1              SAVE INDEX
 396 0000e1f4  1d00                      BCLR    6,PORTA         CLEAR DATA
 397 0000e1f6  ae08                      LDX     #8
 398 0000e1f8  48                DIS1    LSLA                    SET UP
 399 0000e1f9  2402                      BCC     DIS2            BIT OF
 400 0000e1fb  1c00                      BSET    6,PORTA         ACCUMULATOR
 401 0000e1fd  1e00              DIS2    BSET    7,PORTA         CLOCK
 402 0000e1ff  1f00                      BCLR    7,PORTA         IT
 403 0000e201  1d00                      BCLR    6,PORTA         CLEAR DATA
 404 0000e203  5a                        DECX                    COMPLETE?
 405 0000e204  26f2                      BNE     DIS1            NO
 406 0000e206  be28                      LDX     W1              RESTORE INDEX
 407 0000e208  5a                        DECX
 408 0000e209  2ae5                      BPL     DISCHR
 409 0000e20b  1601                      BSET    3,PORTB         ENABLE (144115) HIGH
 410 0000e20d  81                        RTS
 411
```

```
    412                         ******************************************************
    413                         *                                                    *
    414                         *       S-record input RAM accessing.                *
    415                         *                                                    *
    416                         ******************************************************
    417
    418 0000e20e  1201          RAMACC  BSET    1,PORTB         5    5   XFER A14 & A15 TO PORTB
    419 0000e210  0e3002                BRSET   7,ADDRH,A15H    5   10   A15 HIGH ?
    420 0000e213  1301                  BCLR    1,PORTB         5   15   NO
    421 0000e215  1001          A15H    BSET    0,PORTB         5   20   YES
    422 0000e217  0c3002                BRSET   6,ADDRH,A14H    5   25   A14 HIGH ?
    423 0000e21a  1101                  BCLR    0,PORTB         5   30   NO
    424 0000e21c  be30          A14H    LDX     ADDRH           3   33   YES
    425 0000e21e  bf2a                  STX     ADDEH           4   37
    426 0000e220  1c2a                  BSET    6,ADDEH         5   42   A14 HIGH
    427 0000e222  1f2a                  BCLR    7,ADDEH         4   47   A15 LOW
    428
    429 0000e224  0a3108                BRSET   5,STAT,L3       5   52   READING ?
    430 0000e227  aec7                  LDX     #$C7            2   54   NO, WRITING (STA)
    431 0000e229  bf29                  STX     W2              4   58   STA IN
    432 0000e22b  bd29                  JSR     W2             16   74   RAM SUBROUTINE
    433 0000e22d  b72d                  STA     W4              4   78   SAVE FOR READBACK CHECK
    434 0000e22f  aec6          L3      LDX     #$C6            2   80   READING (LDA)
    435 0000e231  bf29                  STX     W2             13   93   LDA IN RAM
    436
    437 0000e233  bc29                  JMP     W2             14  107   121 (89 FOR READ) WITH JSR

em64k.as5

    439                         ******************************************************
    440                         *                                                    *
    441                         *       Clear display table.                         *
    442                         *                                                    *
    443                         ******************************************************
    444
    445 0000e235  ae05          CLRTAB  LDX     #5
    446 0000e237  6f20          CLRLOC  CLR     DTABL,X         CLEAR SIX
    447 0000e239  5a                    DECX                    LOCATIONS IN
    448 0000e23a  2afb                  BPL     CLRLOC          DISPLAY TABLE
    449 0000e23c  81                    RTS
    450
    451                         ******************************************************
    452                         *                                                    *
    453                         *       Emulator mode.                               *
    454                         *                                                    *
    455                         ******************************************************
    456
    457 0000e23d  cde235        MODE3   JSR     CLRTAB
    458 0000e240  a6f1                  LDA     #$F1            E
    459 0000e242  b720                  STA     DTABL
    460 0000e244  a673                  LDA     #$73            P
    461 0000e246  b721                  STA     DTABL+1
    462 0000e248  a663                  LDA     #$63            ?
    463 0000e24a  b723                  STA     DTABL+3
    464 0000e24c  cde1ec                JSR     DISTAB
    465 0000e24f  cde05f        KSC     JSR     KEYSCN          WAIT UNTIL EPROM REMOVED
    466 0000e252  24fb                  BCC     KSC
    467 0000e254  a162                  CMP     #$62            EMULATION CONFIRMED ?
    468 0000e256  2703                  BEQ     CONF
    469 0000e258  cce04e                JMP     GETCMD
    470
    471 0000e25b  a628          CONF    LDA     #$28            MODE 3, ENABLE (144115) HIGH
    472 0000e25d  b701                  STA     PORTB
    473 0000e25f  a6f1                  LDA     #$F1            E
    474 0000e261  b720                  STA     DTABL
    475 0000e263  a6d6                  LDA     #$D6            U
    476 0000e265  b721                  STA     DTABL+1
    477 0000e267  a6d0                  LDA     #$D0            L
```

```
    478 0000e269  b722                    STA     DTABL+2
    479 0000e26b  a677                    LDA     #$77            A
    480 0000e26d  b723                    STA     DTABL+3
    481 0000e26f  a6f0                    LDA     #$F0            T
    482 0000e271  b724                    STA     DTABL+4
    483 0000e273  a6f1                    LDA     #$F1            E
    484 0000e275  b725                    STA     DTABL+5
    485 0000e277  cde1ec                  JSR     DISTAB
    486
    487 0000e27a  8e                STP   STOP

em64k.as5

    489                           ************************************************
    490                           *                                              *
    491                           *       Xfer EPROM contents to RAM from emulation  *
    492                           *       socket ($4000).                        *
    493                           *                                              *
    494                           ************************************************
    495
    496 0000e27b  cde235          DUMP1   JSR     CLRTAB
    497 0000e27e  cde1ec                  JSR     DISTAB
    498 0000e281  a658                    LDA     #$58            MODE 2, ENABLE (144115) HIGH
    499 0000e283  b701                    STA     PORTB
    500 0000e285  1431                    BSET    2,STAT          REAL ADDRESS (NO OFFSET)
    501 0000e287  3f2b                    CLR     ADDRL
    502 0000e289  3f30                    CLR     ADDRH
    503 0000e28b  1201            LLP1    BSET    1,PORTB         5   5   XFER A14 & A15 TO PORTB
    504 0000e28d  0e3002                  BRSET   7,ADDRH,AD15H   5  10   A15 HIGH ?
    505 0000e290  1301                    BCLR    1,PORTB         5  15   NO
    506 0000e292  1001            AD15H   BSET    0,PORTB         5  20   YES
    507 0000e294  0c3002                  BRSET   6,ADDRH,AD14H   5  25   A14 HIGH ?
    508 0000e297  1101                    BCLR    0,PORTB         5  30   NO
    509 0000e299  be30            AD14H   LDX     ADDRH           3  33   YES
    510 0000e29b  bf2a                    STX     ADDEH           4  37
    511 0000e29d  1f2a                    BCLR    7,ADDEH         5  42   A15 LOW
    512 0000e29f  1c2a                    BSET    6,ADDEH         5  47   A14 HIGH
    513 0000e2a1  1d01                    BCLR    6,PORTB
    514 0000e2a3  1e01                    BSET    7,PORTB         READ FROM EMULATOR SOCKET
    515 0000e2a5  cde570                  JSR     LOAD            LOAD BYTE
    516 0000e2a8  1c01                    BSET    6,PORTB
    517 0000e2aa  1f01                    BCLR    7,PORTB         WRITE TO RAM
    518 0000e2ac  cde562                  JSR     STORE           STORE BYTE
    519 0000e2af  3c2b                    INC     ADDRL
    520 0000e2b1  2602                    BNE     SKPH
    521 0000e2b3  3c30                    INC     ADDRH
    522 0000e2b5  b630            SKPH    LDA     ADDRH
    523 0000e2b7  26d2                    BNE     LLP1            MSB ZERO ?
    524 0000e2b9  b62b                    LDA     ADDRL           YES, LSB ZERO ?
    525 0000e2bb  26ce                    BNE     LLP1            IF SO, FINISHED
    526 0000e2bd  cce04e                  JMP     GETCMD
    527
```

```
528                             ****************************************************
529                             *                                                  *
530                             *       Offset for S-record load/send.             *
531                             *                                                  *
532                             ****************************************************
533
534 0000e2c0  1c31                OFFSET  BSET    6,STAT          NO ADDRESS INC/DEC
535 0000e2c2  1431                        BSET    2,STAT          REAL ADDRESS
536 0000e2c4  a6d7                        LDA     #$D7            0
537 0000e2c6  b720                        STA     DTABL
538 0000e2c8  a671                        LDA     #$71            F
539 0000e2ca  b721                        STA     DTABL+1
540 0000e2cc  b722                        STA     DTABL+2
541 0000e2ce  3f23                        CLR     DTABL+3
542 0000e2d0  3f2a                        CLR     ADDEH
543 0000e2d2  3f30                        CLR     ADDRH
544 0000e2d4  a639                        LDA     #OFF
545 0000e2d6  b72b                        STA     ADDRL
546 0000e2d8  cce506                      JMP     MEMEX3
```

em64k.as5

```
548                             ****************************************************
549                             *                                                  *
550                             *       Xfer EPROM contents to RAM from auxiliary  *
551                             *       socket ($8000).                            *
552                             *                                                  *
553                             ****************************************************
554
555 0000e2db  cde235            DUMP9   JSR     CLRTAB
556 0000e2de  cde1ec                    JSR     DISTAB
557 0000e2e1  a658                      LDA     #$58            MODE 2, ENABLE (144115) HIGH
558 0000e2e3  b701                      STA     PORTB
559 0000e2e5  1431                      BSET    2,STAT          REAL ADDRESS (NO OFFSET)
560
561 0000e2e7  ad0f              TLOP    BSR     T19
562 0000e2e9  3c01                      INC     PORTB           NEXT PAGE
563 0000e2eb  b601                      LDA     PORTB
564 0000e2ed  a403                      AND     #3
565 0000e2ef  a103                      CMP     #3              LAST PAGE ?
566 0000e2f1  25f4                      BLO     TLOP
567 0000e2f3  ad03                      BSR     T19             YES
568 0000e2f5  cce04e                    JMP     GETCMD
569
570 0000e2f8  3f2b              T19     CLR     ADDRL
571 0000e2fa  3f30                      CLR     ADDRH
572
573 0000e2fc  be30              LLP9    LDX     ADDRH           YES
574 0000e2fe  bf2a                      STX     ADDEH
575 0000e300  1e2a                      BSET    7,ADDEH         A15 HIGH
576 0000e302  1d2a                      BCLR    6,ADDEH         A14 LOW
577 0000e304  cde570                    JSR     LOAD            READ FROM AUXILIARY SOCKET
578 0000e307  1f2a                      BCLR    7,ADDEH         A15 LOW
579 0000e309  1c2a                      BSET    6,ADDEH         A14 HIGH
580 0000e30b  cde562                    JSR     STORE           WRITE TO EMULATION RAM
581 0000e30e  3c2b                      INC     ADDRL
582 0000e310  2602                      BNE     SKPH9
583 0000e312  3c30                      INC     ADDRH
584 0000e314  b630              SKPH9   LDA     ADDRH
585 0000e316  a140                      CMP     #$40            LAST ADDRESS $3FFF
586 0000e318  26e2                      BNE     LLP9            FINISHED ?
587
588 0000e31a  81                        RTS
```

em64k.as5

```
590                             ****************************************************
591                             *                                                  *
592                             *       RS232 (9600) S-Record receiver (4MHz).     *
593                             *                                                  *
594                             ****************************************************
595
596 0000e31b  1a31               VERIFY  BSET    5,STAT          SERIAL VERIFY
597 0000e31d  ae06                     LDX     #6
598 0000e31f  2003                     BRA     L4
599 0000e321  1b31               TLOAD   BCLR    5,STAT          SERIAL LOAD
600 0000e323  5f                       CLRX
601 0000e324  a681               L4      LDA     #$81            RTS
602 0000e326  b72c                     STA     W3
603 0000e328  cde118                   JSR     DISP            DISPLAY "LOAd OR UErIFy"
604
605 0000e32b  ad5f               INPUT   BSR     INCHD           7 BIT ASCII INTO A
606 0000e32d  a153                     CMP     #'S'            S ?
607 0000e32f  26fa                     BNE     INPUT           NO, TRY AGAIN
608 0000e331  ad59                     BSR     INCHD           YES, GET NEXT CHARACTER
609 0000e333  a139                     CMP     #'9'            9 ?
610 0000e335  276f                     BEQ     NINE            YES, FINISH
611 0000e337  a131                     CMP     #'1'            NO, 1 ?
612 0000e339  26f0                     BNE     INPUT           NO, TRY AGAIN
613
614 0000e33b  3f32               LNGTH   CLR     CHKSUM          YES, CLEAR CHECKSUM
615 0000e33d  3f35                       CLR     TMP2            AND TEMP. STORE
616 0000e33f  cde3bd                     JSR     BYTEI           AND GET BYTE COUNT
617 0000e342  b736                       STA     BCNT            AND SAVE IT
618
619 0000e344  cde3bd             ADDR    JSR     BYTEI           ADDRESS HIGH
620 0000e347  b039                       SUB     OFF             OFFSET
621 0000e349  b730                       STA     ADDRH
622 0000e34b  cde3bd                     JSR     BYTEI           ADDRESS LOW
623 0000e34e  b72b                       STA     ADDRL
624
625 0000e350  cde3bd             DLOP    JSR     BYTEI          75        GET A BYTE
626 0000e353  271d                       BEQ     CHCK            3   78   LAST BYTE ?
627 0000e355  0b310b                     BRCLR   5,STAT,L5       5   83   NO, VERIFYING ?
628 0000e358  b72f                       STA     W6              4   87   YES
629 0000e35a  cde20e                     JSR     RAMACC         66  153   READ RAM
630 0000e35d  b12f                       CMP     W6              3  156   SAME ?
631 0000e35f  2658                       BNE     ERR7            3  159
632 0000e361  2007                       BRA     L6              3  162
633 0000e363  cde20e             L5      JSR     RAMACC         67  150            NO, WRITE TO RAM
634 0000e366  b12d                       CMP     W4              3  153            READBACK
635 0000e368  263f                       BNE     ERR2            3  156            OK ?
636 0000e36a  3c2b               L6      INC     ADDRL           5  161    5 167   INCREMENT LS ADDRESS
637 0000e36c  2602                       BNE     NOOVR           3  164    3 170   OVERFLOW ?
638 0000e36e  3c30                       INC     ADDRH           5  169    5 175   YES, INC. HIGH BYTE
639 0000e370  20de               NOOVR   BRA     DLOP            3  172    3 178

em64k.as5
```

```
                                ********************************************************
                                *                                                      *
                                *       Checksum byte & error routine.                 *
                                *                                                      *
                                ********************************************************

0000e372  bb32          CHCK    ADD     CHKSUM
0000e374  b732                  STA     CHKSUM          DEBUG
0000e376  a1ff                  CMP     #$FF            IS CHECKSUM BYTE OK ?
0000e378  27b1                  BEQ     INPUT           YES, AND AGAIN

0000e37a  ae01                  LDX     #1
0000e37c  b738          ERR     STA     ERDAT           DEBUG
0000e37e  bf37                  STX     ERTYP           DEBUG
0000e380  9f                    TXA
0000e381  cde5f2                JSR     PRTDAT
0000e384  3f24                  CLR     DTABL+4
0000e386  cde616                JSR     PRTADR
0000e389  cce000                JMP     SCAN

                                ********************************************************
                                *                                                      *
                                *       Input routine,  MC68HC05E0 : 0.5 uS.           *
                                *       Cycles per bit at 9600 baud : 208              *
                                *                                                      *
                                ********************************************************

0000e38c  ad60          INCHD   BSR     DEL191        191       GET OUT OF BIT 7
0000e38e  0501fd        INCH    BRCLR   2,PORTB,*       5       IS LINE HIGH ?
0000e391  0401fd                BRSET   2,PORTB,*       5       YES, WAIT FOR START
0000e394  ae07                  LDX     #7              2    6  7 DATA BITS TO READ
0000e396  bf33                  STX     COUNT           4   10
0000e398  ad58                  BSR     DEL110        110  120  +/-3   OF 1st BIT

0000e39a  ad52          INBT    BSR     DEL191        191       +120-208=103
0000e39c  050100                BRCLR   2,PORTB,ZER     5  196  CYC 2 (105) READ
0000e39f  46            ZER     RORA                    3  199  SAVE BIT
0000e3a0  3a33                  DEC     COUNT           5  204
0000e3a2  26f6                  BNE     INBT            3  207

0000e3a4  44                    LSRA                    3   16  MSB A ZERO
0000e3a5  81                    RTS                     6   22

0000e3a6  cce04e        NINE    JMP     GETCMD

0000e3a9  ae02          ERR2    LDX     #2              READBACK FROM RAM
0000e3ab  20cf                  BRA     ERR
0000e3ad  ae03          ERR3    LDX     #3              LESS THAN ASCII 0
0000e3af  20cb                  BRA     ERR
0000e3b1  ae04          ERR4    LDX     #4              BETWEEN ASCII 9 & A
0000e3b3  20c7                  BRA     ERR
0000e3b5  ae05          ERR5    LDX     #5              MORE THAN ASCII F
0000e3b7  20c3                  BRA     ERR
0000e3b9  ae07          ERR7    LDX     #7              VERIFY ERROR
0000e3bb  20bf                  BRA     ERR
```

em64k.as5

```
                                ******************************************************
                                *                                                    *
                                *       Byte input sub-routines.                     *
                                *                                                    *
                                ******************************************************

0000e3bd  adcd          BYTEI   BSR     INCHD           22          MS NIBBLE
0000e3bf  ad17                  BSR     ASCII           35   57     WHAT WAS IT
0000e3c1  48                    LSLA                     3          YES
0000e3c2  48                    LSLA                     3          SHIFT
0000e3c3  48                    LSLA                     3          IT
0000e3c4  48                    LSLA                     3   69     UP
0000e3c5  b734                  STA     TMP1             4   71     AND SAVE IT
0000e3c7  b635                  LDA     TMP2             3   74     RESTORE BYTE
0000e3c9  bb32                  ADD     CHKSUM           3   79     ACCUMULATE
0000e3cb  b732                  STA     CHKSUM           4   83     IN CHECKSUM BYTE
0000e3cd  adbd                  BSR     INCHD           22          LS NIBBLE
0000e3cf  ad07                  BSR     ASCII           35   57     WHAT WAS IT
0000e3d1  bb34                  ADD     TMP1             3   60     ADD TO MS NIBBLE
0000e3d3  b735                  STA     TMP2             4   64     SAVE BYTE
0000e3d5  3a36                  DEC     BCNT             5   69     DECREMENT BYTE COUNT
0000e3d7  81                    RTS                      6   75

0000e3d8  a130          ASCII   CMP     #$30             2          BEFORE ZERO ?
0000e3da  25d1                  BLO     ERR3             3    5     YES, NOT LEGAL
0000e3dc  a139                  CMP     #$39             2    7     AFTER NINE
0000e3de  2203                  BHI     MT9              3   10     YES TRY A-F
0000e3e0  a030                  SUB     #$30             3   13     0-9, CONVERT TO HEX
0000e3e2  81                    RTS                      6   19

0000e3e3  a141          MT9     CMP     #$41             2   12     BEFORE A ?
0000e3e5  25ca                  BLO     ERR4             3   15     YES, NOT LEGAL
0000e3e7  a146                  CMP     #$46             2   17     AFTER F ?
0000e3e9  22ca                  BHI     ERR5             3   20     YES, NOT LEGAL
0000e3eb  a037                  SUB     #$37             3   23     A-F, CONVERT TO HEX
0000e3ed  81            NFND    RTS                      6   29

0000e3ee  ae1d          DEL191  LDX     #29              2
0000e3f0  2002                  BRA     DELAY            3    5
0000e3f2  ae10          DEL110  LDX     #16              2
0000e3f4  5a            DELAY   DECX                     3
0000e3f5  26fd                  BNE     DELAY            3          6xX
0000e3f7  81                    RTS                      6          12+6X (INC BSR)

em64k.as5

                                ******************************************************
                                *                                                    *
                                *       RS232 (9600 @ 4MHz) S-Record transmitter.    *
                                *                                                    *
                                ******************************************************

0000e3f8  1406          PUNCH   BSET    2,PORTBD         BIT 2 OUTPUT
0000e3fa  cde0db                JSR     BLDRNG           BUILD RANGE
0000e3fd  0831a6                BRSET   4,STAT,NINE      NEW ADDRESS ENTERED ?
0000e400  be26                  LDX     TEMP             NO, SWAP ADDRESSES
0000e402  b726                  STA     TEMP
0000e404  bf30                  STX     ADDRH
0000e406  b62b                  LDA     ADDRL
0000e408  be27                  LDX     TEMP+1
0000e40a  bf2b                  STX     ADDRL
0000e40c  b727                  STA     TEMP+1
0000e40e  1f31                  BCLR    7,STAT           CLEAR END FLAG
0000e410  1531                  BCLR    2,STAT           EMULATION ADDRESS

0000e412  b627          LOOP1   LDA     TEMP+1           END LSB
0000e414  b02b                  SUB     ADDRL            CURRENT LSB
```

```
    762 0000e416  b72f                    STA     W6              DIFFERENCE LSB
    763 0000e418  b626                    LDA     TEMP            END MSB
    764 0000e41a  b230                    SBC     ADDRH           CURRENT MSB
    765 0000e41c  260d                    BNE     LOTS            MSB ZERO ?
    766 0000e41e  b62f                    LDA     W6              YES, LOOK AT LSB
    767 0000e420  4c                      INCA                    ADJUST
    768 0000e421  2708                    BEQ     LOTS            WAS $FF ?
    769 0000e423  a120                    CMP     #$20            MORE THAN 23 ?
    770 0000e425  2204                    BHI     LOTS            IF SO USE 23
    771 0000e427  1e31                    BSET    7,STAT          NO, LAST S1 RECORD
    772 0000e429  2002                    BRA     LTE20           LESS THAN OR EQUAL TO 20
    773
    774 0000e42b  a620              LOTS  LDA     #$20
    775 0000e42d  ab03              LTE20 ADD     #$03            ADD BYTE COUNT & ADDRESS
    776 0000e42f  b736                    STA     BCNT            No. BYTES THIS S1 RECORD
    777 0000e431  a653                    LDA     #'S'            S
    778 0000e433  cde484                  JSR     OUCH
    779 0000e436  a631                    LDA     #'1'            1
    780 0000e438  ad4a                    BSR     OUCH
    781 0000e43a  3f32                    CLR     CHKSUM
    782 0000e43c  b636                    LDA     BCNT            BYTE COUNT
    783 0000e43e  ad72                    BSR     BYTEO
    784 0000e440  b630                    LDA     ADDRH           ADDRESS HIGH
    785 0000e442  ad6e                    BSR     BYTEO
    786 0000e444  b62b                    LDA     ADDRL
    787 0000e446  ad6a                    BSR     BYTEO           ADDRESS LOW
    788
    789 0000e448  cde570            LOOP2 JSR     LOAD            GET BYTE
    790 0000e44b  3c2b                    INC     ADDRL           INCREMENT ADDRESS
    791 0000e44d  2602                    BNE     NOVR            OVERFLOW ?
    792 0000e44f  3c30                    INC     ADDRH           YES, INC. HIGH BYTE
    793 0000e451  ad5f              NOVR  BSR     BYTEO           SEND BYTE
    794 0000e453  26f3                    BNE     LOOP2           LAST BYTE ?

em64k.as5

    796                           ********************************************************
    797                           *                                                      *
    798                           *       Checksum byte.                                 *
    799                           *                                                      *
    800                           ********************************************************
    801
    802 0000e455  b632                    LDA     CHKSUM          CHECKSUM
    803 0000e457  43                      COMA                    REQUIRED CHECKSUM BYTE
    804 0000e458  ad58                    BSR     BYTEO           SEND IT
    805 0000e45a  ad22                    BSR     CRLF            CRLF
    806 0000e45c  0f31b3                  BRCLR   7,STAT,LOOP1    FINISHED ?
    807
    808                           ********************************************************
    809                           *                                                      *
    810                           *       S9 record.                                     *
    811                           *                                                      *
    812                           ********************************************************
    813
    814 0000e45f  a653                    LDA     #'S'            S
    815 0000e461  ad21                    BSR     OUCH
    816 0000e463  a639                    LDA     #'9'            9
    817 0000e465  ad1d                    BSR     OUCH
    818 0000e467  a603                    LDA     #$03            3 BYTES
    819 0000e469  ad47                    BSR     BYTEO
    820 0000e46b  a600                    LDA     #$00
    821 0000e46d  ad43                    BSR     BYTEO           DUMMY (0)
    822 0000e46f  a600                    LDA     #$00
    823 0000e471  ad3f                    BSR     BYTEO           ADDRESS
    824 0000e473  a6fc                    LDA     #$FC
    825 0000e475  ad3b                    BSR     BYTEO           CHECKSUM
    826 0000e477  ad05                    BSR     CRLF
    827 0000e479  1506                    BCLR    2,PORTBD        BIT 2 INPUT
```

```
    828 0000e47b  cce04e                    JMP     GETCMD
    829
    830 0000e47e  a60d              CRLF    LDA     #$0D            CR
    831 0000e480  ad02                      BSR     OUCH
    832 0000e482  a60a                      LDA     #$0A            LF

em64k.as5

    834                             ********************************************************
    835                             *                                                      *
    836                             *       Output routine, 208 cycles per bit.            *
    837                             *                                                      *
    838                             ********************************************************
    839
    840 0000e484  1401              OUCH    BSET    2,PORTB         5       MAKE SURE IT'S HIGH
    841 0000e486  ae0a                      LDX     #10             2    7   10 BITS TO SEND
    842 0000e488  bf33                      STX     COUNT           4   11   START, 8 DATA, STOP
    843 0000e48a  9d                        NOP
    844 0000e48b  9d                        NOP
    845 0000e48c  cde3f4                    JSR     DELAY          72   83
    846 0000e48f  98                        CLC                     2   85   START A ZERO
    847 0000e490  2008                      BRA     STAR            3   88
    848
    849 0000e492  ae1c              OUTBT   LDX     #28             2
    850 0000e494  cde3f4                    JSR     DELAY         180  182
    851 0000e497  9d                        NOP                     2  184
    852 0000e498  99                DEL3    SEC                     2  186   FILL WITH ONES FOR STOP
    853 0000e499  46                        RORA                    3  189   GET A BIT
    854 0000e49a  2504              STAR    BCS     OU1             3  192   1 ?
    855 0000e49c  1501                      BCLR    2,PORTB         5  197   NO
    856 0000e49e  2004                      BRA     OBD             3  200
    857 0000e4a0  1401              OU1     BSET    2,PORTB         5        YES
    858 0000e4a2  2000                      BRA     OBD             3
    859 0000e4a4  3a33              OBD     DEC     COUNT           5  205
    860 0000e4a6  26ea                      BNE     OUTBT           3  208   DONE ?
    861 0000e4a8  81                        RTS
    862
    863 0000e4a9  ab30              ASCIO   ADD     #$30            3        CONVERT TO ASCII
    864 0000e4ab  a139                      CMP     #$39            2    5   0-9 ?
    865 0000e4ad  2302                      BLS     NMT9            3    8
    866 0000e4af  ab07                      ADD     #$07            3    8   NO, A-F
    867 0000e4b1  81                NMT9    RTS                     6   14
    868
    869                             ********************************************************
    870                             *                                                      *
    871                             *       Byte output sub-routine.                       *
    872                             *                                                      *
    873                             ********************************************************
    874
    875 0000e4b2  b734              BYTEO   STA     TMP1
    876 0000e4b4  44                        LSRA                    SHIFT
    877 0000e4b5  44                        LSRA                    DOWN TO
    878 0000e4b6  44                        LSRA                    GET MSB
    879 0000e4b7  44                        LSRA
    880 0000e4b8  adef                      BSR     ASCIO           & CONVERT
    881 0000e4ba  adc8                      BSR     OUCH
    882 0000e4bc  b634                      LDA     TMP1            RESTORE BYTE
    883 0000e4be  bb32                      ADD     CHKSUM          ACCUMULATE
    884 0000e4c0  b732                      STA     CHKSUM          IN CHECKSUM BYTE
    885 0000e4c2  b634                      LDA     TMP1
    886 0000e4c4  a40f                      AND     #$0F            LSB
    887 0000e4c6  ade1                      BSR     ASCIO           CONVERT IT
    888 0000e4c8  adba                      BSR     OUCH
    889 0000e4ca  3a36                      DEC     BCNT            DECREMENT BYTE COUNT
    890 0000e4cc  81                        RTS

em64k.as5
```

```
892                             ****************************************************
893                             *                                                  *
894                             *       Segment codes for the MC145000.            *
895                             *                                                  *
896                             ****************************************************
897
898 0000e4cd  d7                  CTABL   FCB     $D7     0
899 0000e4ce  06                          FCB     $06     1
900 0000e4cf  e3                          FCB     $E3     2
901 0000e4d0  a7                          FCB     $A7     3
902 0000e4d1  36                          FCB     $36     4
903 0000e4d2  b5                          FCB     $B5     5
904 0000e4d3  f5                          FCB     $F5     6
905 0000e4d4  07                          FCB     $07     7
906 0000e4d5  f7                          FCB     $F7     8
907 0000e4d6  b7                          FCB     $B7     9
908 0000e4d7  77                          FCB     $77     A
909 0000e4d8  f4                          FCB     $F4     B
910 0000e4d9  d1                          FCB     $D1     C
911 0000e4da  e6                          FCB     $E6     D
912 0000e4db  f1                          FCB     $F1     E
913 0000e4dc  71                          FCB     $71     F
914
915 0000e4dd  cde235            ERROR   JSR     CLRTAB
916 0000e4e0  a6f1                      LDA     #$F1
917 0000e4e2  b721                      STA     DTABL+1
918 0000e4e4  a660                      LDA     #$60
919 0000e4e6  b722                      STA     DTABL+2
920 0000e4e8  b723                      STA     DTABL+3
921 0000e4ea  cce059                    JMP     DSCN
922
923                             ****************************************
924                             *                                      *
925                             *       INPUT ONE CHARACTER            *
926                             *       A REGISTER CONTAINS HEX VALUE  *
927                             *       X REGISTER CONTAINS HEX VALUE  *
928                             *                                      *
929                             ****************************************
930
931 0000e4ed  cde05f            CHRIN   JSR     KEYSCN          GET KEY
932 0000e4f0  24fb                      BCC     CHRIN           IF NOT VALID RETRY
933 0000e4f2  5f                        CLRX
934 0000e4f3  d1e0c3            CHRIN1  CMP     STABL,X         CONVERT
935 0000e4f6  2703                      BEQ     CHRIN2          TO HEX
936 0000e4f8  5c                        INCX
937 0000e4f9  20f8                      BRA     CHRIN1
938 0000e4fb  9f                CHRIN2  TXA                     IF CANCEL
939 0000e4fc  81                        RTS
```

em64k.as5

```
941                             ****************************************************
942                             *                                                  *
943                             *       Memory examine/change.                     *
944                             *                                                  *
945                             ****************************************************
946
947 0000e4fd  1531              MEMEX   BCLR    2,STAT          EMULATION ADDRESS
948 0000e4ff  cde5b9                    JSR     GETADR          GET ADDRESS
949 0000e502  a110                      CMP     #$10            ESCAPE ?
950 0000e504  2752                      BEQ     MEMEX4
951
952 0000e506  ad68              MEMEX3  BSR     LOAD            LOAD DATA
953 0000e508  cde5f2                    JSR     PRTDAT          PRINT IT
954 0000e50b  cde5af                    JSR     GETNYB          GET NEW NIBBLE
955 0000e50e  a110                      CMP     #$10            ESCAPE ?
956 0000e510  2746                      BEQ     MEMEX4
957 0000e512  a111                      CMP     #$11            ENTER ?
```

```
   958 0000e514  271a                    BEQ     ADRINC
   959 0000e516  a113                    CMP     #$13            MEMORY ?
   960 0000e518  272e                    BEQ     ADRDEC
   961
   962 0000e51a  a10f                    CMP     #$0F
   963 0000e51c  2208                    BHI     CMDMDL          VALID HEX ?
   964
   965 0000e51e  cde5f2          MEMEX1  JSR     PRTDAT          PRINT IT
   966 0000e521  cde59d                  JSR     GETBY2          SHIFT IN NEXT
   967 0000e524  25f8                    BCS     MEMEX1          IF VALID TRY AGAIN
   968 0000e526  a111            CMDMDL  CMP     #$11            ENTER ?
   969 0000e528  2614                    BNE     MEMEX2          NO
   970 0000e52a  b629                    LDA     W2              RESTORE ACCA
   971 0000e52c  ad34                    BSR     STORE           YES STORE IT
   972 0000e52e  25d6                    BCS     MEMEX3          STORE VALID ?
   973 0000e530  0c3125          ADRINC  BRSET   6,STAT,MEMEX4
   974 0000e533  3c2b                    INC     ADDRL           YES GOTO
   975 0000e535  2602                    BNE     MEMEX5          NEXT
   976 0000e537  3c30                    INC     ADDRH
   977 0000e539  cde616          MEMEX5  JSR     PRTADR          PRINT IT
   978 0000e53c  20c8                    BRA     MEMEX3          REPEAT
   979 0000e53e  a113            MEMEX2  CMP     #$13            M ?
   980 0000e540  2616                    BNE     MEMEX4          NO
   981 0000e542  b629                    LDA     W2
   982 0000e544  ad1c                    BSR     STORE
   983 0000e546  25be                    BCS     MEMEX3
   984 0000e548  0c310d          ADRDEC  BRSET   6,STAT,MEMEX4
   985 0000e54b  3d2b                    TST     ADDRL           YES THEN
   986 0000e54d  2602                    BNE     CMDMB2          GET PREVIOUS
   987 0000e54f  3a30                    DEC     ADDRH           ADDRESS
   988 0000e551  3a2b            CMDMB2  DEC     ADDRL
   989 0000e553  cde616                  JSR     PRTADR          PRINT IT
   990 0000e556  20ae                    BRA     MEMEX3          REPEAT
   991 0000e558  0d3102          MEMEX4  BRCLR   6,STAT,NORM2
   992 0000e55b  3f2b                    CLR     ADDRL
   993 0000e55d  1d31            NORM2   BCLR    6,STAT
   994 0000e55f  cce04e                  JMP     GETCMD

em64k.as5

   996                           ********************************************
   997                           *                                          *
   998                           *       LOAD/STORE AT ADDRH(EH), ADDRL     *
   999                           *                                          *
  1000                           ********************************************
  1001
  1002 0000e562  aec7            STORE   LDX     #$C7            SET-UP
  1003 0000e564  ad0c                    BSR     LDSTCM          ROUTINE
  1004 0000e566  b72d                    STA     W4              TO DO
  1005 0000e568  ad06                    BSR     LOAD            TWO BYTE
  1006 0000e56a  b12d                    CMP     W4              STORE
  1007 0000e56c  2701                    BEQ     STRTS
  1008 0000e56e  99                      SEC
  1009 0000e56f  81              STRTS   RTS
  1010
  1011 0000e570  aec6            LOAD    LDX     #$C6            SET-UP ROUTINE
  1012 0000e572  bf29            LDSTCM  STX     W2              TO DO
  1013 0000e574  ae81                    LDX     #$81            TWO BYTE
  1014 0000e576  bf2c                    STX     W3              LOAD
  1015 0000e578  043120                  BRSET   2,STAT,NORM     REAL ADDRESS ?
  1016
  1017 0000e57b  b72d                    STA     W4
  1018 0000e57d  b630                    LDA     ADDRH
  1019 0000e57f  b039                    SUB     OFF
  1020 0000e581  b72e                    STA     W5
  1021 0000e583  b62d                    LDA     W4
  1022
  1023 0000e585  1201            RMCC    BSET    1,PORTB         5   5    XFER A14 & A15 TO PORTB
```

```
1024 0000e587  0e2e02                  BRSET   7,W5,A15HI      5  10   A15 HIGH ?
1025 0000e58a  1301                    BCLR    1,PORTB         5  15   NO
1026 0000e58c  1001            A15HI   BSET    0,PORTB         5  20   YES
1027 0000e58e  0c2e02                  BRSET   6,W5,A14HI      5  25   A14 HIGH ?
1028 0000e591  1101                    BCLR    0,PORTB         5  30   NO
1029 0000e593  be2e            A14HI   LDX     W5              3  33   YES
1030 0000e595  bf2a                    STX     ADDEH           4  37
1031 0000e597  1c2a                    BSET    6,ADDEH         5  42   A14 HIGH
1032 0000e599  1f2a                    BCLR    7,ADDEH         4  47   A15 LOW
1033 0000e59b  bc29            NORM    JMP     W2             14  61   67 (66 FOR LDA) WITH JSR
1034
1035                           ********************************************
1036                           *                                          *
1037                           *       Build a byte.                      *
1038                           *                                          *
1039                           ********************************************
1040
1041 0000e59d  b729            GETBY2  STA     W2
1042 0000e59f  ad0e                    BSR     GETNYB
1043 0000e5a1  240b                    BCC     GETBRT
1044 0000e5a3  3829                    ASL     W2
1045 0000e5a5  3829                    ASL     W2
1046 0000e5a7  3829                    ASL     W2
1047 0000e5a9  3829                    ASL     W2
1048 0000e5ab  ba29                    ORA     W2
1049 0000e5ad  99                      SEC
1050 0000e5ae  81              GETBRT  RTS

em64k.as5

1052                           ********************************************
1053                           *                                          *
1054                           *       Get one character into ACCA        *
1055                           *       X destroyed, C set if hex.         *
1056                           *                                          *
1057                           ********************************************
1058
1059 0000e5af  cde4ed          GETNYB  JSR     CHRIN           GET CHARACTER
1060 0000e5b2  98                      CLC
1061 0000e5b3  a10f                    CMP     #$0F            VALID HEX?
1062 0000e5b5  2201                    BHI     GETRET          NO
1063 0000e5b7  99                      SEC                     YES
1064 0000e5b8  81              GETRET  RTS
1065
1066                           ********************************************
1067                           *                                          *
1068                           *       Build address A,X dest., address   *
1069                           *       in ADDRL/ADDRH, C set if new.      *
1070                           *                                          *
1071                           ********************************************
1072
1073 0000e5b9  cde235          GETADR  JSR     CLRTAB          BLANK DISPLAY
1074 0000e5bc  ad58                    BSR     PRTADR
1075 0000e5be  adef            BLDADR  BSR     GETNYB          GET CHARACTER
1076 0000e5c0  250a                    BCS     GETAD1          VALID HEX?
1077 0000e5c2  a110                    CMP     #$10
1078 0000e5c4  272b                    BEQ     GETRTS
1079 0000e5c6  a111                    CMP     #$11            NO ENTER?
1080 0000e5c8  2727                    BEQ     GETRTS          NO TRY AGAIN
1081 0000e5ca  20ed                    BRA     GETADR
1082 0000e5cc  3f30            GETAD1  CLR     ADDRH           INIT HIGH ADDRESS
1083 0000e5ce  b72b                    STA     ADDRL           PUT CHAR AWAY
1084 0000e5d0  ad44                    BSR     PRTADR          PRINT NEW ADDRESS
```

```
1085 0000e5d2  addb          GETALP  BSR     GETNYB          GET ANOTHER CHAR
1086 0000e5d4  2412                  BCC     GETARG          VALID?
1087 0000e5d6  48                    ASLA                    YES
1088 0000e5d7  48                    ASLA                    SHIFT IT IN
1089 0000e5d8  48                    ASLA
1090 0000e5d9  48                    ASLA
1091 0000e5da  ae04                  LDX     #4
1092 0000e5dc  48            GETASF  ASLA
1093 0000e5dd  392b                  ROL     ADDRL
1094 0000e5df  3930                  ROL     ADDRH
1095 0000e5e1  5a                    DECX
1096 0000e5e2  26f8                  BNE     GETASF
1097 0000e5e4  ad30                  BSR     PRTADR          PRINT NEW ADDR
1098 0000e5e6  20ea                  BRA     GETALP          GET ANOTHER CHAR
1099 0000e5e8  a110          GETARG  CMP     #$10            ESCAPE ?
1100 0000e5ea  2705                  BEQ     GETRTS
1101 0000e5ec  a111                  CMP     #$11            ENTER ?
1102 0000e5ee  26e2                  BNE     GETALP          NO TRY AGAIN
1103 0000e5f0  99                    SEC                     YES SET FLAG
1104 0000e5f1  81            GETRTS  RTS
```

em64k.as5

```
1106                         ***************************************************
1107                         *                                                 *
1108                         *       Print one byte (from A).                  *
1109                         *                                                 *
1110                         *       Print address ADDRH.ADDRL.                *
1111                         *                                                 *
1112                         ***************************************************
1113
1114 0000e5f2  ae04          PRTDAT  LDX     #4              PRINT IN LAST TWO LCD DIGITS
1115 0000e5f4  bf28          PRTBYT  STX     W1
1116 0000e5f6  b72d                  STA     W4
1117 0000e5f8  44                    LSRA
1118 0000e5f9  44                    LSRA
1119 0000e5fa  44                    LSRA
1120 0000e5fb  44                    LSRA
1121 0000e5fc  97                    TAX
1122 0000e5fd  d6e4cd                LDA     CTABL,X
1123 0000e600  be28                  LDX     W1
1124 0000e602  e720                  STA     DTABL,X
1125 0000e604  b62d                  LDA     W4
1126 0000e606  a40f                  AND     #$0F
1127 0000e608  97                    TAX
1128 0000e609  d6e4cd                LDA     CTABL,X
1129 0000e60c  be28                  LDX     W1
1130 0000e60e  e721                  STA     DTABL+1,X
1131 0000e610  cde1ec                JSR     DISTAB
1132 0000e613  b62d                  LDA     W4
1133 0000e615  81                    RTS
1134
1135 0000e616  b72e          PRTADR  STA     W5              PRINT ADDRESS (FIRST 4 DIGITS)
1136 0000e618  bf2c                  STX     W3
1137 0000e61a  b630                  LDA     ADDRH
1138 0000e61c  5f                    CLRX
1139 0000e61d  add5                  BSR     PRTBYT
1140 0000e61f  b62b                  LDA     ADDRL
1141 0000e621  ae02                  LDX     #2
1142 0000e623  adcf                  BSR     PRTBYT
1143 0000e625  b62e                  LDA     W5
1144 0000e627  be2c                  LDX     W3
1145 0000e629  81                    RTS
1146
```

```
1147                                ****************************************************
1148                                *                                                  *
1149                                *       MC68HC05E0 Vectors.                        *
1150                                *                                                  *
1151                                ****************************************************
1152
1153                                        ORG     $FFF4
1154
1155 0000fff4  e022                        FDB     START           SERIAL
1156 0000fff6  e022                        FDB     START           TIMER B
1157 0000fff8  e022                        FDB     START           TIMER A
1158 0000fffa  e04e                        FDB     GETCMD          EXTERNAL INTERRUPT
1159 0000fffc  e022                        FDB     START           SWI
1160 0000fffe  e022                        FDB     START           RESET
1161                                        END
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor