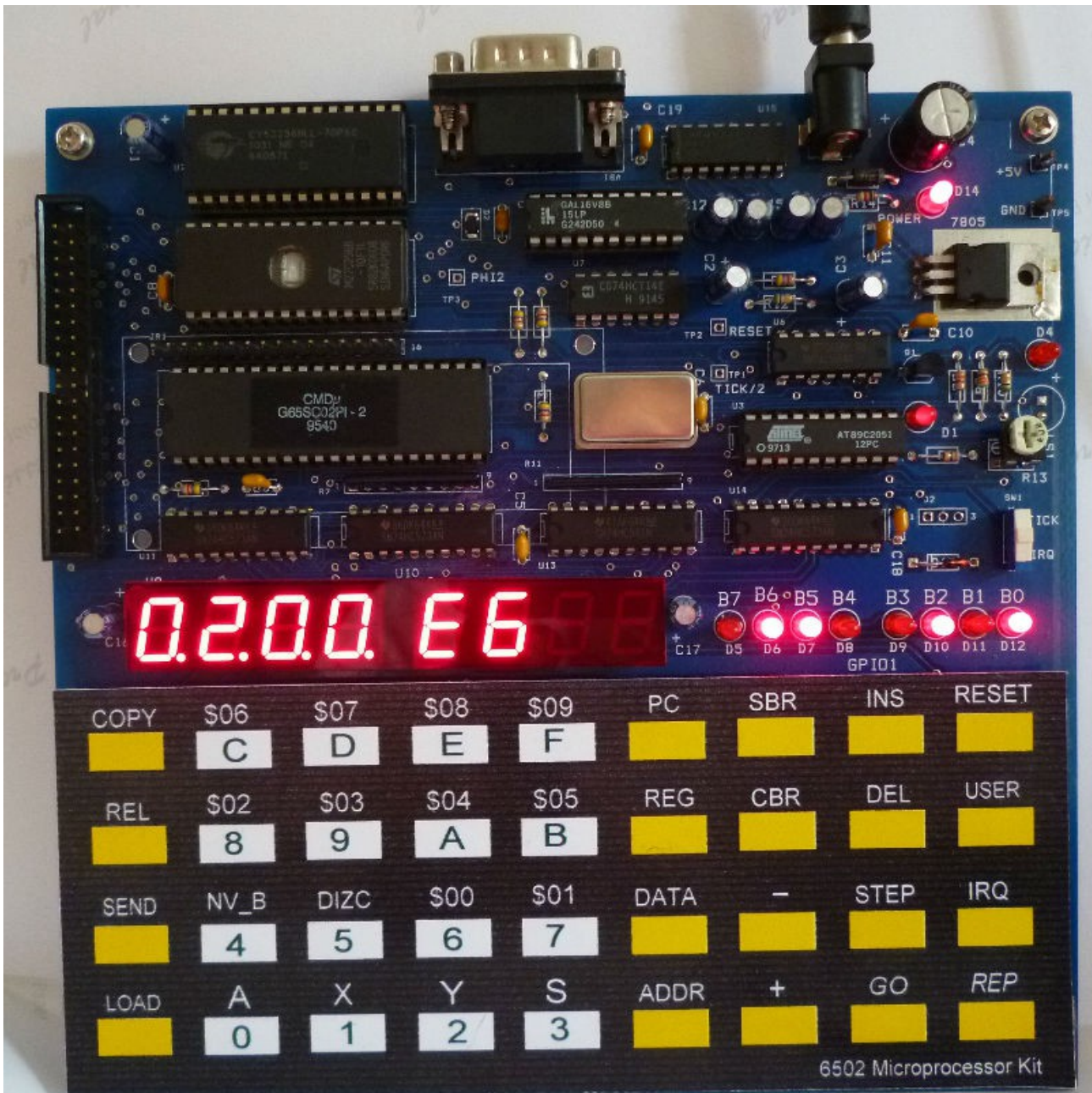


# 6502 Microprocessor Kit User's Manual



# 6502 MICROPROCESSOR KIT

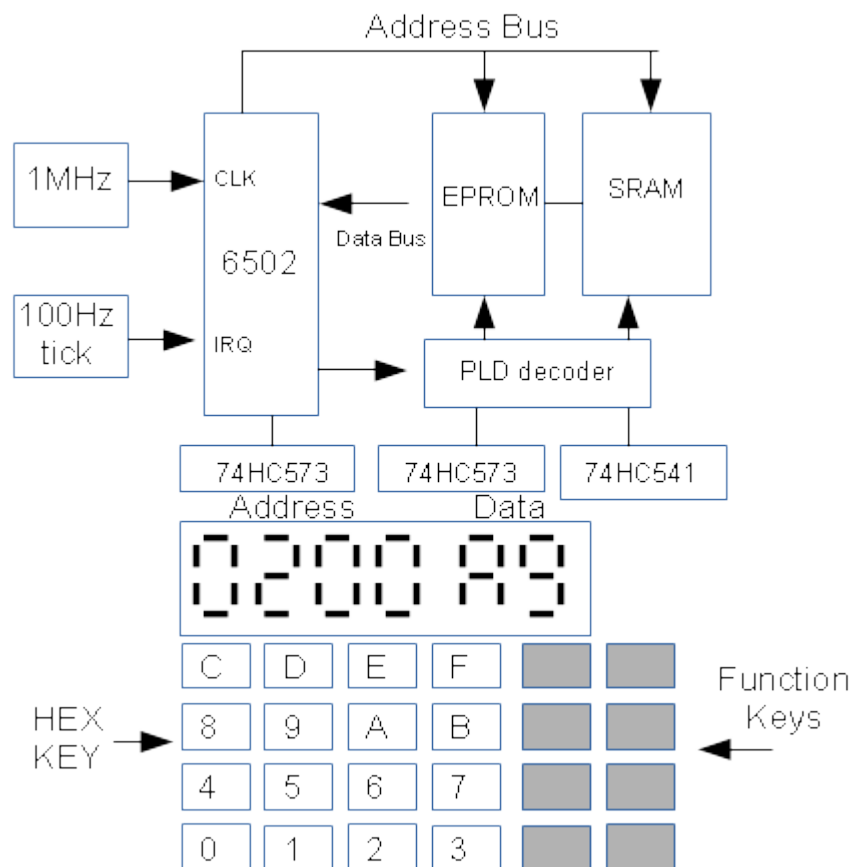
## CONTENTS

OVERVIEW.....	3
FUNCTIONAL BLOCK DIAGARM.....	3
HARDWARE LAYOUT.....	4
KEYBOARD LAYOUT.....	5
HARDWARE FEATURES.....	6
MEMORY AND I/O MAPS.....	6
GPIO1 LED.....	7
CONNECTING KIT TO TERMINAL.....	8
EXPANSION BUS HEADER.....	9
10ms TICK GENERATOR.....	10
RS232C PORT.....	11
DATA FRAME for UART COMMUNICATION.....	11
CONNECTING LCD MODULE.....	13
LOGIC PROBE POWER SUPPLY.....	16
WRITE YOUR OWN MONITOR PROGRAM.....	17
HARDWARE SCHEMATIC, BOM.....	18
MONITOR PROGRAM LISTINGS.....	22

## OVERVIEW

The 6502 Microprocessor kit is a new design single board computer using the G65SC02 as a CPU. This single board computer is a basic learning tool for programming the 6502 with low level instructions hex code. The board has hex keypad and 7-segment display for entering the instruction hex code and test it directly. Students will learn basic of the computer hardware and software of the 6502 easily. The manual also provides monitor program listings, the method to modify or write your own monitor program.

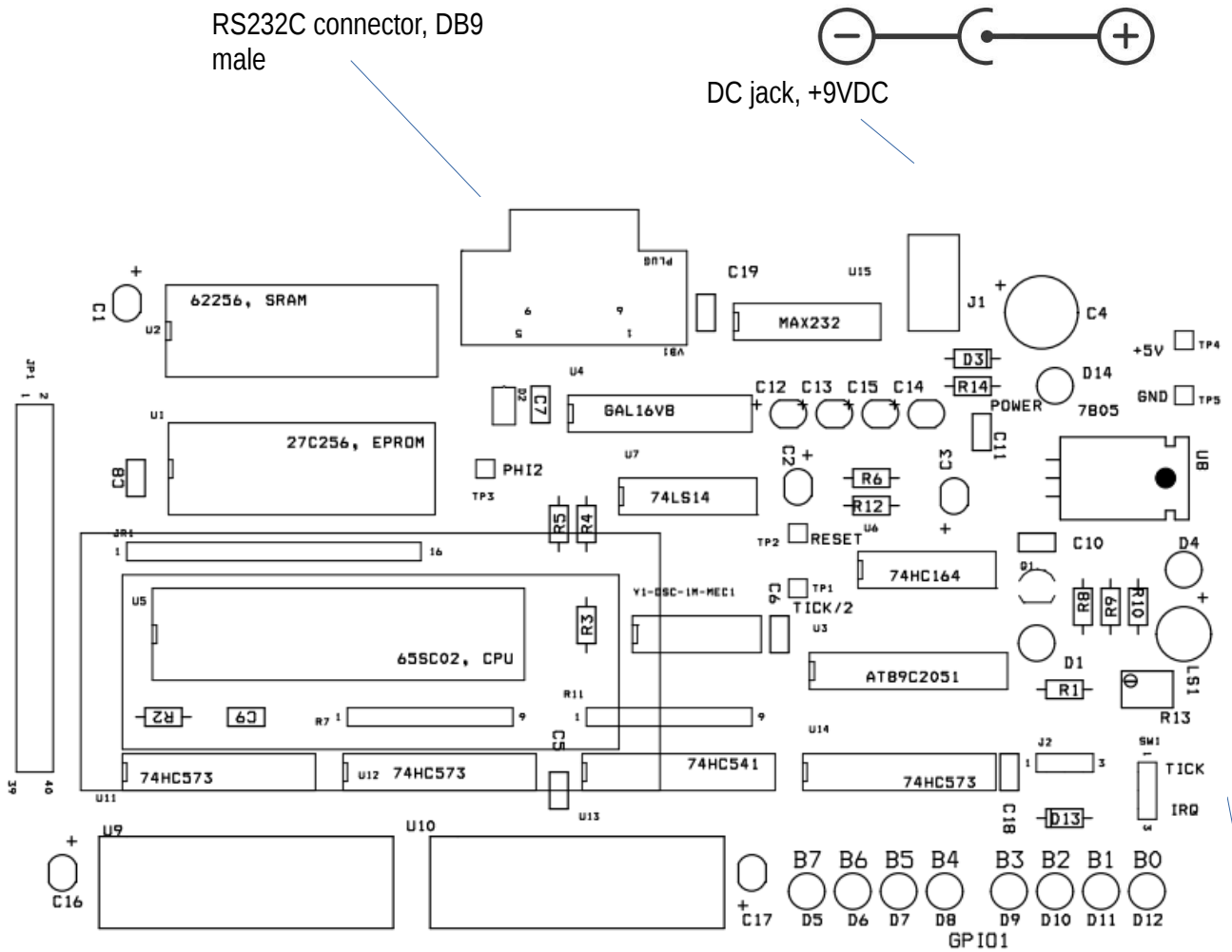
### 6502 KIT FUNCTIONAL BLOCK DIAGRAM



### Notes

1. UART is software control for low speed asynchronous communication.
2. The kit provides 8-bit LCD module interfacing bus.
3. 100Hz Tick generator is for interrupt experiment.
4. Ports for display and keypad interfacing were built with discrete logic IC chips.
5. Memory and Port decoders are made with Programmable Logic Device, PLD.

## HARDWARE LAYOUT



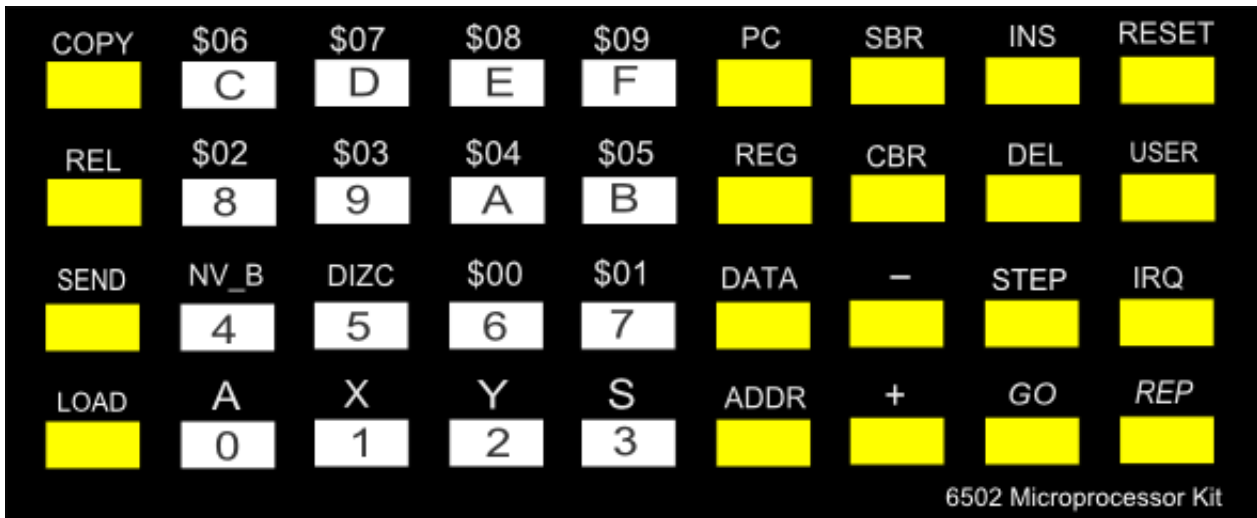
20-pin Text LCD header.

Selector for 10ms tick or  
IRQ key

### Important Notes

1. Plugging or removing the LCD module must be done when the kit is powered off!
2. AC adapter should provide approx. +9VDC, higher voltage will cause the voltage regulator chip becomes hot.
3. The kit has diode protection for wrong polarity of adapter jack. If the center pin is not the positive (+), the diode will be reverse bias, preventing wrong polarity to feed to voltage regulator.

## KEYBOARD LAYOUT



**HEX keys** Hexadecimal number 0 to F with associated user registers, flag bits and page zero memory \$00 to \$09 (use with key REG)

### CPU control keys

**RESET** Reset the CPU, the 6502 will get reset vector from location FFFC and FFFD

**USER** User key for lab test, active low

**IRQ** Make IRQ pin to logic low, used for experimenting with interrupt process

### Monitor function keys

**REP** Repeat the key that pressed, must be pressed together with REP key.

**INS** N/A.

**DEL** N/A

**STEP** Execute user code only single instruction and return to save CPU registers

**GO** Jump from monitor program to user code

- Decrement current display address by one

+ Increment current display address by one

**PC** Set current display address with user Program Counter

<b>REG</b>	Display user registers, flags or page zero \$00 to \$09 with HEX key.	
<b>DATA</b>	Set entry mode of hex keys to Data field	6
<b>ADDR</b>	Set entry mode of hex keys to Address field	
<b>COPY</b>	N/A	
<b>REL</b>	Compute relative byte, used with key + for Start, Destination and key GO	
<b>SEND</b>	N/A	
<b>LOAD</b>	Load Intel or MOS hex file at 2400 bit/s using serial port	

Note, N/A=not available for the current version of monitor program

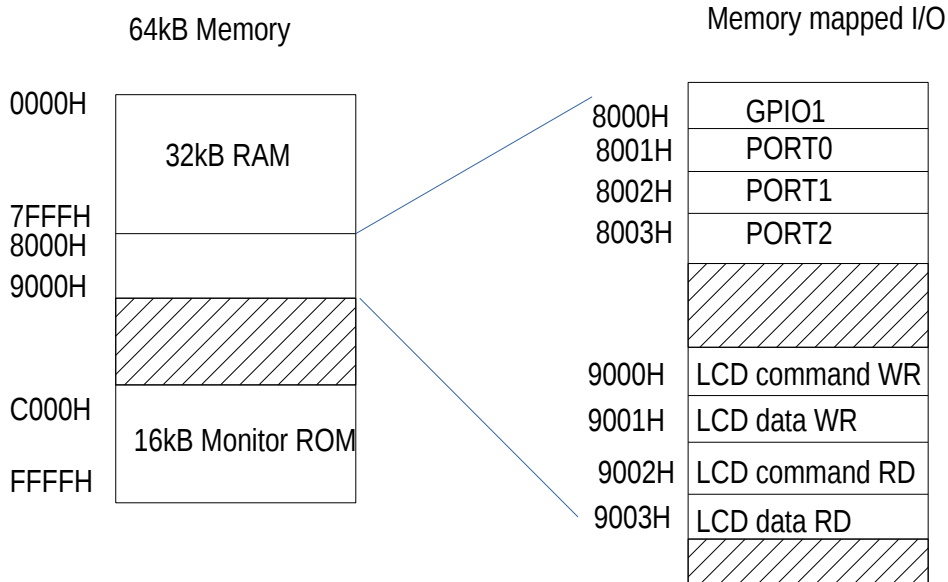
## **HARDWARE FEATURES**

The hardware features are as follows;

- CPU: 65SC02, CMOS 8-bit Microprocessor @1MHz clock
- Memory: 32kB RAM, 16kB EPROM
- Memory and I/O Decoder chip: Programmable Logic Device GAL16V8D
- Display: 6-digit 7-segment LED
- Keyboard: 36 keys
- RS232 port: software controlled UART 2400 bit/s 8n1
- Debugging LED: 8-bit GPIO1 LED at location \$8000
- Tick: 10ms tick produced by 89C2051
- Text LCD interface: direct CPU bus interface text LCD
- Expansion header: 40-pin header

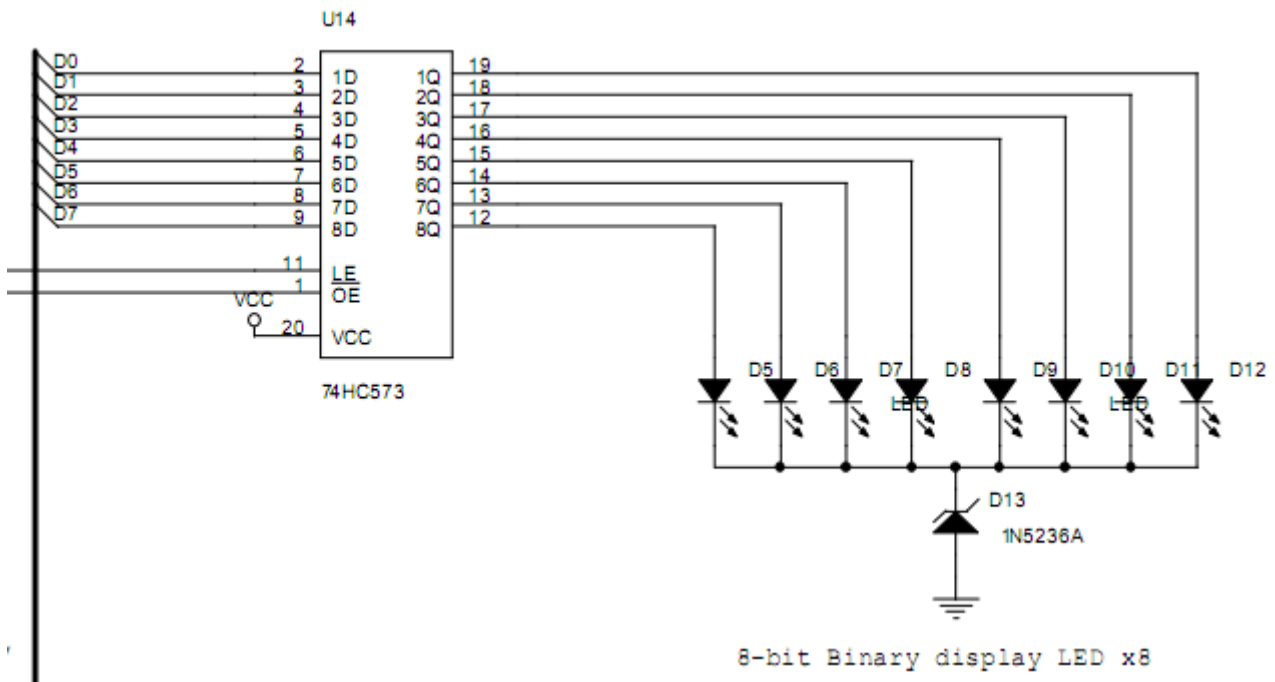
## MEMORY AND I/O MAPS

The first 32kB is RAM space from 0000-7FFFH. Zero page is location 00-FFH. Stack space is 100-1FFFH. User space is from 200H to 7FFFH. The 6502 CPU uses memory space from 8000H-BFFFH for I/O port. The monitor ROM is located at C000H-FFFFH



## GPIO1 LED

The 6502 kit provides a useful 8-bit binary display. It can be used to debug the program or code running demonstration. The I/O address is 8000H. U14 is 8-bit data latch. Logic 1 at the output will make LED lit.



The GPIO1 LED can be used to display accumulator register easily. Let us take a look the sample code below.

Address	Hex code	Label	Instruction	comment
0200	A9 01	MAIN	LDA #1	Load register A with 1
0202	8D 00 80		STA \$8000	Write A to GPIO1@ 8000H

The test code has only two instructions. The first instruction has two bytes machine code, A9 and 01. The second instruction has three bytes, 8D, 00 and 80.

Enter the hex code to memory from 0200 to 0203. Then press PC, and execute the instruction with single step by pressing key STEP. The 2<sup>nd</sup> press STEP key that executes instruction STA \$8000 will make the GPIO1 LED showing the content of register A. Try change the load value to register A.

Another sample is with JUMP instruction. The JUMP instruction will change the Program Counter to 0200, to repeat program running. Now we use zero page at location 0 to be the byte to be incremented. After incrementing, we load it to register A then write to location of GPIO1 at 8000H. And with JMP LOOP instruction, the program will be repeated.

Address	Hex code	Label	Instruction	comment
0200	E6 00	LOOP	INC \$0	Increment location 0
0202	A5 00		LDA \$0	Load A with location 0
0204	8D 00 80		STA \$8000	Write A to GPIO1@ 8000H
0207	4C 00 02		JMP LOOP	Jump back to loop

Again enter the hex code to memory and test it with single step. Now press key STEP and key REP together. Every time when instruction STA \$8000 was executed, did you see the binary number counting?

We will learn more the use of GPIO1 with 6502 Programming Lab Book.



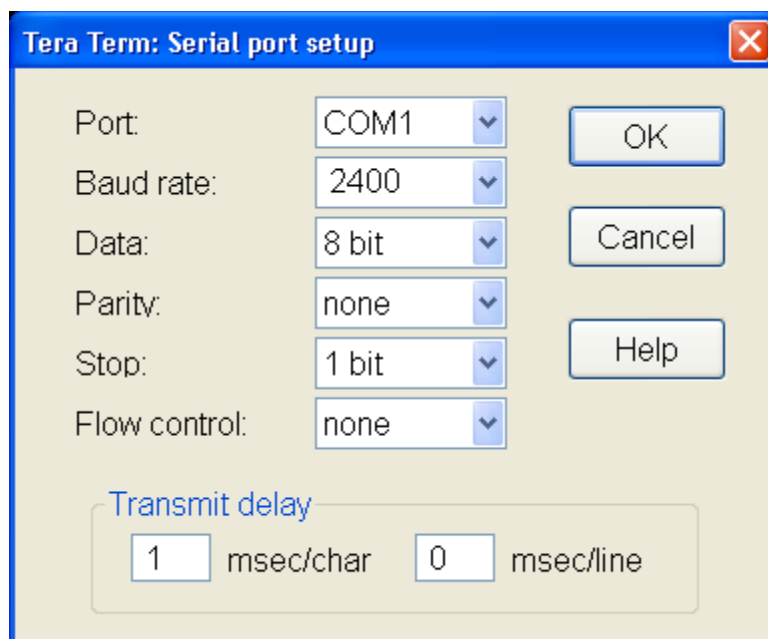
## CONNECTING 6502 KIT TO TERMINAL

For LOAD key, we can connect the 6502 kit to a terminal by RS232C cross cable. You may download free terminal program, teraterm from this URL, <http://tssh2.sourceforge.jp/index.html.en>



The example shows connecting laptop with COM1 port to the RS232C port of the 6502 kit. New laptop has no COM port, we may use the USB-RS232 adapter for converting the USB port to RS232 port.

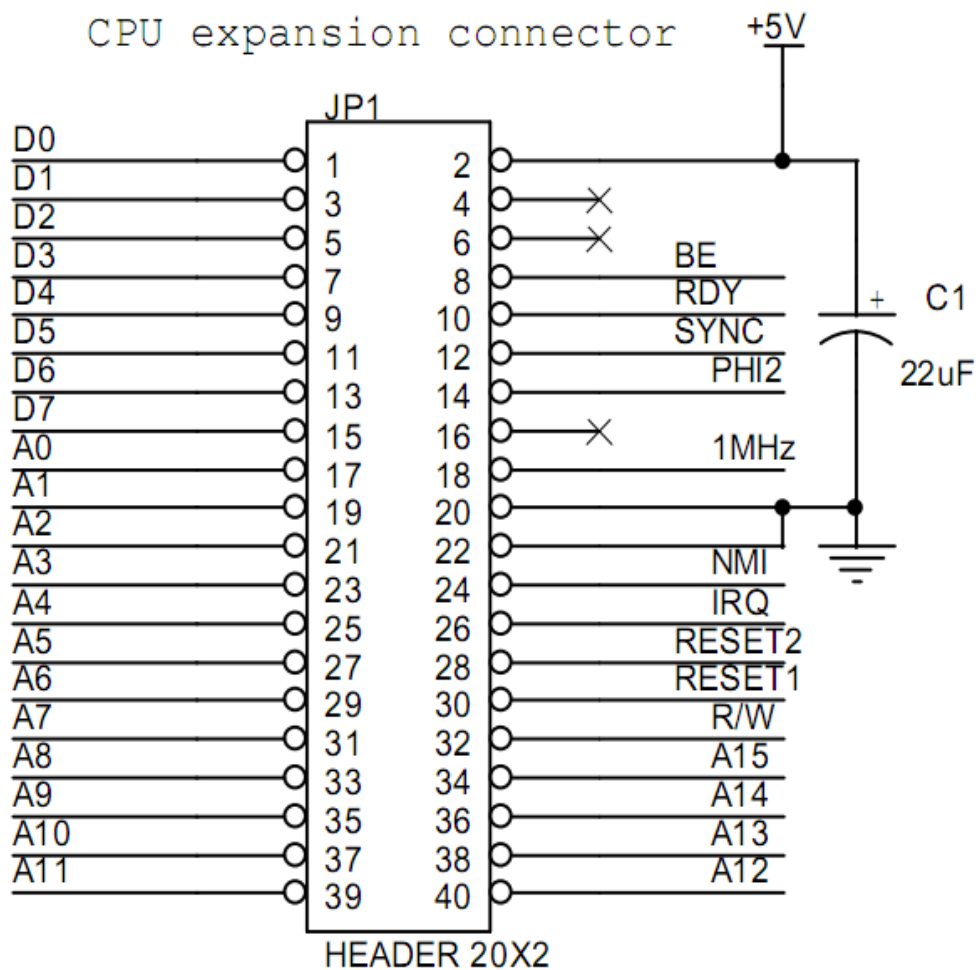
To download Intel or MOS hex file that generated from the assembler or c compiler, **set serial port speed to 2400 bit/s, 8-data bit, no parity, no flow control, one stop bit.**



Press key LOAD, then key GO. The kit will wait for the data stream from terminal. On PC, Click file>Send File>LED.HEX. The kit will read the hex file, write to memory, when completed the start message will be displayed. The kit accepts for both Intel or MOS hex files.

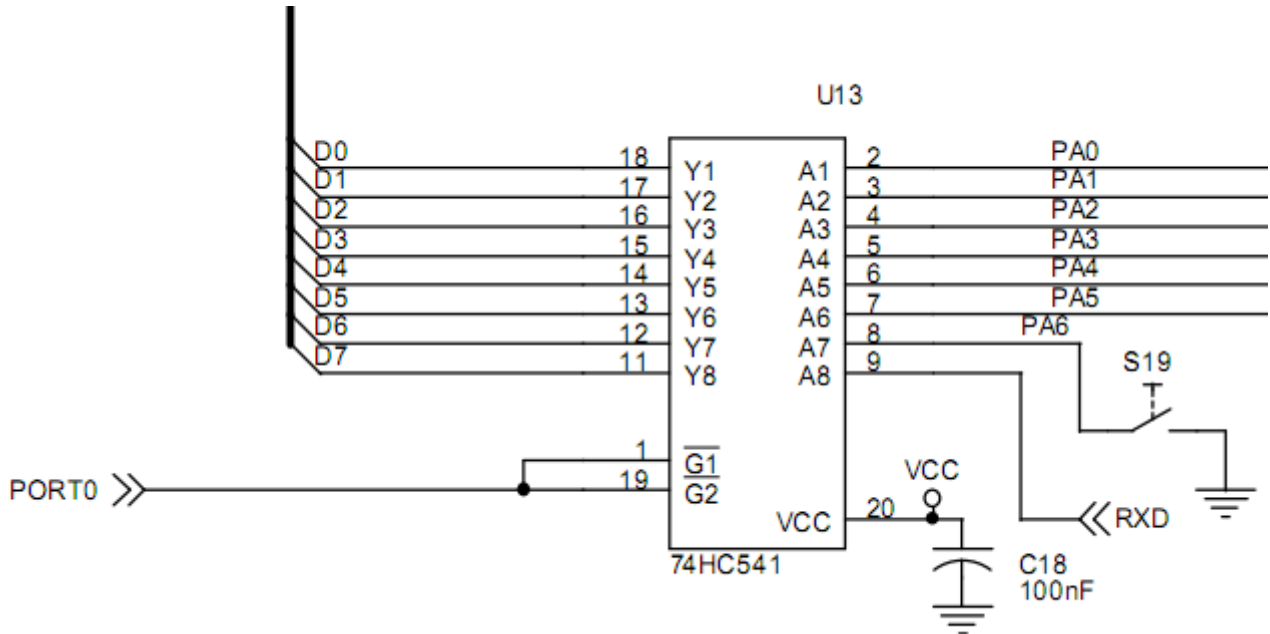
### EXPANSION BUS HEADER

JP1, 40-pin header provides CPU bus signals for expansion or I/O interfacing. Students may learn how to make the simple I/O port, interfacing to Analog-to-Digital Converter, interfacing to stepper motor or AC power circuits.



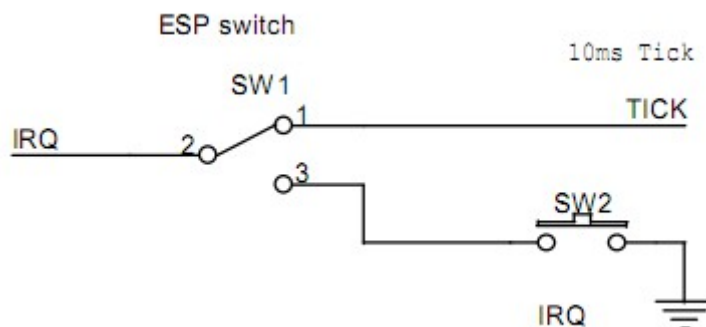
## USER KEY

User key, S19 is one bit active low key switch connected to bit 6 of Port 0. To test the logic of S19, we can use instruction LDA \$8001 and check bit 6 of the accumulator with test bit instruction.

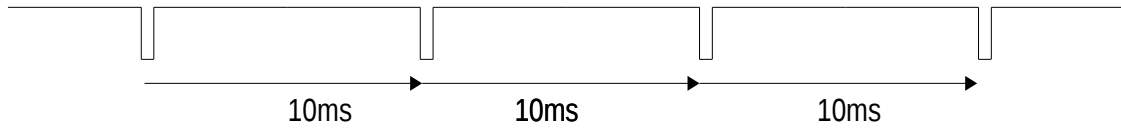


## 10ms TICK GENERATOR

SW1 is a selector for interrupt source between key IRQ or 10ms tick produced by 89C2051 microcontroller. Tick generator is software controlled using timer0 interrupt in the 89C2051 chip. The active low tick signal is sent to P3.7. For tick running indicator, P1.7 drives D1 LED.

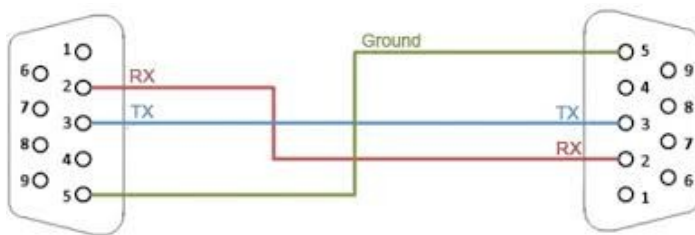


Tick is a 10ms periodic signal for triggering the 6502 IRQ pin. When select SW1 to Tick, the 6502 CPU can be triggered by a maskable interrupt. The 100Hz tick or 10ms tick can be used to produce tasks that executed with multiple of tick. The 6502 kit lab look will show how to use 10ms tick to make a digital timer.



### RS232C PORT

The RS232C port is for serial communication. We can use a cross cable or null MODEM cable to connect between the kit and terminal, or kit #1 to kit #2 for sending or receiving hex file. The connector for both sides are DB9 female. We may build it or buying from computer stores.

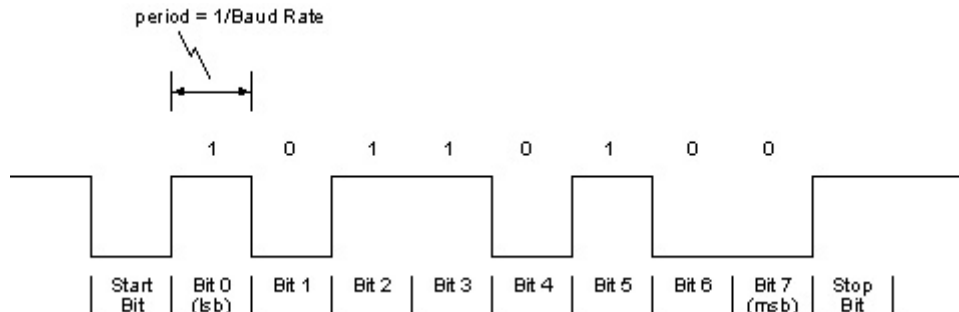


For new PC or laptop computer with USB port, we may have the RS232C port by using the USB to RS232 converter.



## DATA FRAME for UART COMMUNICATION

Serial data that communicated between kit and terminal is asynchronous format. The 6502 kit has no UART chip, instead it uses software controlled to produce bit rate of 2400 bit/s. The data frame is composed of start bit, 8-data bit and stop bit. For our kit, period =  $1/2400$  = 417 microseconds.



Since bit period is provided by machine cycle delay. Thus to send/receive serial data correctly, all interrupts must be disabled.

## CONNECTING LCD MODULE

JR1 is 20-pin header for connecting the LCD module. The example shows connecting the 16x2 lines text LCD module. R12 is a current limit resistor for back-light. R13 is trimmer POT for contrast adjustment. The LCD module is interfaced to the 6502 bus directly. The command and data registers are located in I/O space having address from 9000H to 9003H.



*Be advised that plugging or removing the LCD module must be done when the kit is powered off.*

Text LCD module accepts ASCII codes for displaying the message on screen. Without settings the LCD by software, no characters will be displayed. The first line will be black line by adjusting the R18 for contrast adjustment.

Here is the example program that prints text 6502 Kit on the LCD screen.

```

0001 0000          ; LCD test program for 6502 KIT
0002 0000
0003 0000
0004 0000
0005 0000      BUSY          .EQU 80H
0006 0000
0007 0000      ; below LCD's registers are mapped into memory space
0008 0000
0009 0000      command_write .EQU 9000H
0010 0000      data_write    .EQU 9001H
0011 0000      command_read  .EQU 9002H
0012 0000      data_read     .EQU 9003H
0013 0000
0014 0000
0015 0000          .cseg
0016 0000
0017 0200          .org 200h
0018 0200
0019 0200 4C 5A 02      jmp main
0020 0203
0021 0203
0022 0203      ; wait until LCD ready bit set
0023 0203
0024 0203 48          LcdReady PHA
0025 0204 AD 02 90      ready  LDA command_read
0026 0207 29 80          AND #BUSY
0027 0209 D0 F9          BNE ready    ; loop if busy flag = 1
0028 020B 68          PLA
0029 020C 60          RTS
0030 020D
0031 020D
0032 020D          LCD_command_write
0033 020D 20 03 02      JSR LcdReady
0034 0210 8D 00 90      STA command_write
0035 0213 60          RTS
0036 0214
0037 0214
0038 0214 20 03 02      LCD_data_write JSR LcdReady
0039 0217 8D 01 90      STA data_write
0040 021A 60          RTS
0041 021B
0042 021B
0043 021B 20 03 02      clr_screen JSR LcdReady
0044 021E A9 01          LDA #1
0045 0220 20 0D 02      JSR LCD_command_write
0046 0223 60          RTS
0047 0224
0048 0224 A9 38          InitLcd  LDA #38H
0049 0226 20 0D 02      JSR LCD_command_write
0050 0229 A9 0C          LDA #0CH
0051 022B 20 0D 02      JSR LCD_command_write
0052 022E 20 1B 02      JSR clr_screen
0053 0231 A2 00          LDX #0
0054 0233 A0 00          LDY #0
0055 0235 20 39 02      JSR goto_xy
0056 0238 60          RTS
0057 0239
0058 0239
0059 0239      ; goto_xy(x,y)
0060 0239      ; entry: A = y position
0061 0239      ;          B = x position

```

```

0062 0239
0063 0239 8A      goto_xy    TXA
0064 023A C9 00      CMP #0
0065 023C D0 08      BNE case1
0066 023E 98          TYA
0067 023F 18          CLC
0068 0240 69 80      ADC #80H
0069 0242 20 0D 02   JSR LCD_command_write
0070 0245 60          RTS
0071 0246
0072 0246 C9 01      case1     CMP #1
0073 0248 D0 08      BNE case2
0074 024A 98          TYA
0075 024B 18          CLC
0076 024C 69 C0      ADC #0C0H
0077 024E 20 0D 02   JSR LCD_command_write
0078 0251 60          RTS
0079 0252
0080 0252 60      case2     RTS
0081 0253
0082 0253
0083 0253
0084 0253          ; write ASCII code to LCD at current position
0085 0253          ; entry: A
0086 0253
0087 0253 20 03 02   putch_lcd JSR LcdReady
0088 0256 20 14 02   JSR LCD_data_write
0089 0259 60          RTS
0090 025A
0091 025A
0092 025A
0093 025A 20 24 02   main     JSR InitLcd
0094 025D A9 36      LDA #'6'
0095 025F 20 53 02   JSR putch_lcd
0096 0262 A9 35      LDA #'5'
0097 0264 20 53 02   JSR putch_lcd
0098 0267 A9 30      LDA #'0'
0099 0269 20 53 02   JSR putch_lcd
0100 026C A9 32      LDA #'2'
0101 026E 20 53 02   JSR putch_lcd
0102 0271 A9 20      LDA #' '
0103 0273 20 53 02   JSR putch_lcd
0104 0276 A9 4B      LDA #'K'
0105 0278 20 53 02   JSR putch_lcd
0106 027B A9 69      LDA #'i'
0107 027D 20 53 02   JSR putch_lcd
0108 0280 A9 74      LDA #'t'
0109 0282 20 53 02   JSR putch_lcd
0110 0285
0111 0285 00          brk
0112 0286
0113 0286
0114 0286          .END
0115 0286
0116 0286
0117 0286
0118 0286
0119 0286
tasm: Number of errors = 0

```

## LOGIC PROBE POWER SUPPLY

The kit provides test points TP4(+5V) and TP5(GND) for using the logic probe. Students may learn digital logic signals with logic probe easily. The important signals are RESET (TP2) and PHI2 clock (TP3). Tick signal, however indicated by D1 LED blinking. Logic probe can test it at P3.7 of the 89C2051 microcontroller directly. Red clip is for +5V and Black clip for GND.



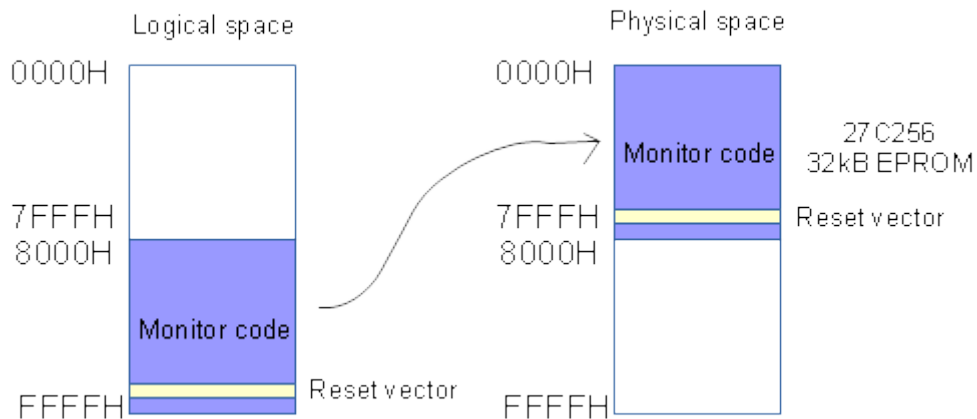


## WRITE YOUR OWN MONITOR PROGRAM

The monitor ROM is U1, 27C256. Source code of the monitor program is available for download. Students can learn and modify the source code by adding more function keys, utility subroutines. The monitor program can be tested in RAM and the move to ROM chip by using the EPROM programmer.

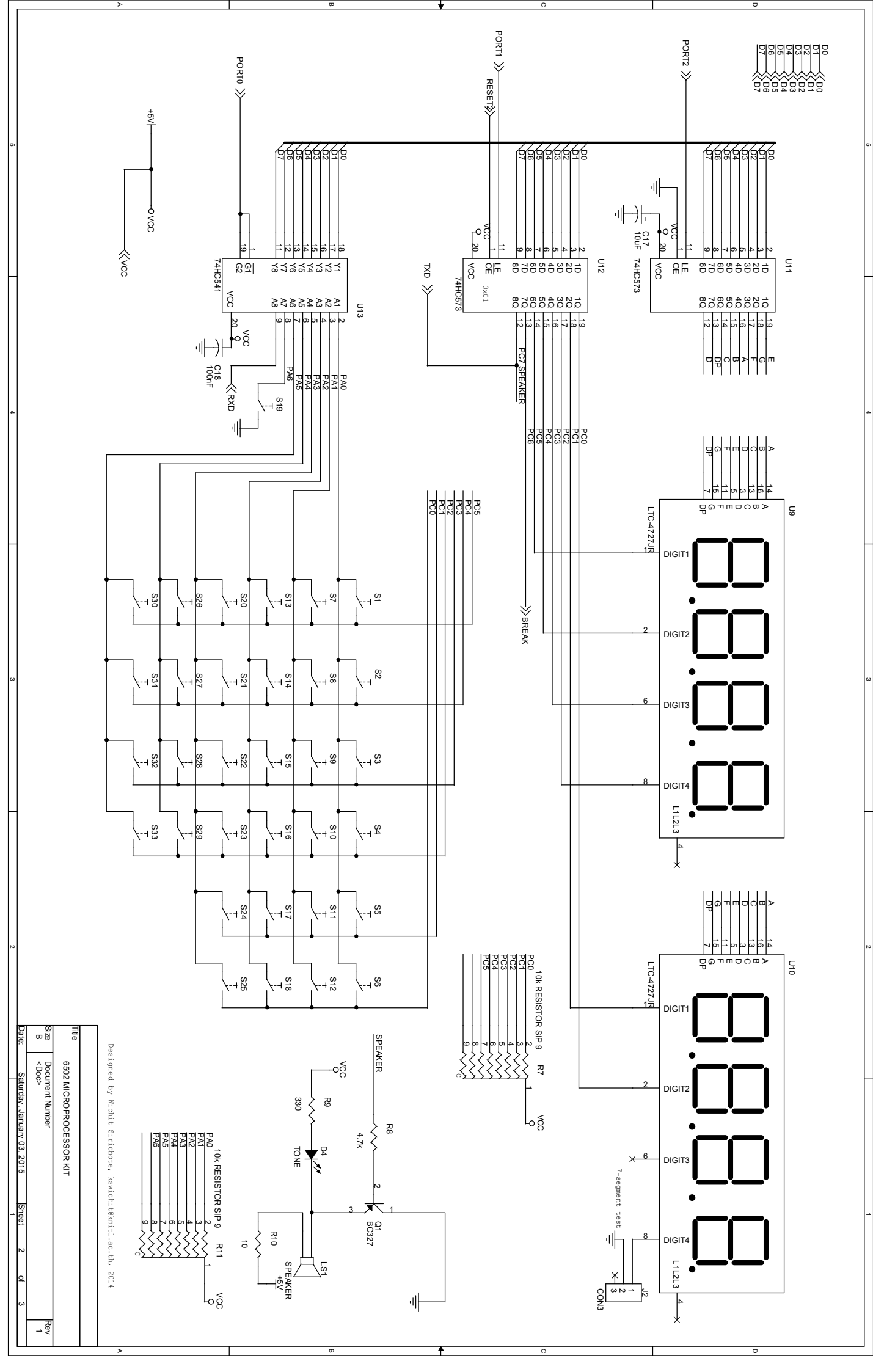
The procedure of modifying the monitor program is as follows.

1. Set the code segment to 1000h, so the hex file can be tested in RAM.
  2. Use TASM with command, d:\6502\tasm -65 monitor.asm
  3. The object file will be hex file, download to the kit and test run.
  4. If it works fine then change code segment to 0c000h for ROM.
  5. The physical address will occupy last block of 64kB space.
  6. Our EPROM is 32kB, so we must move the hex code from 2nd block to the first block.
- Then program the EPROM.



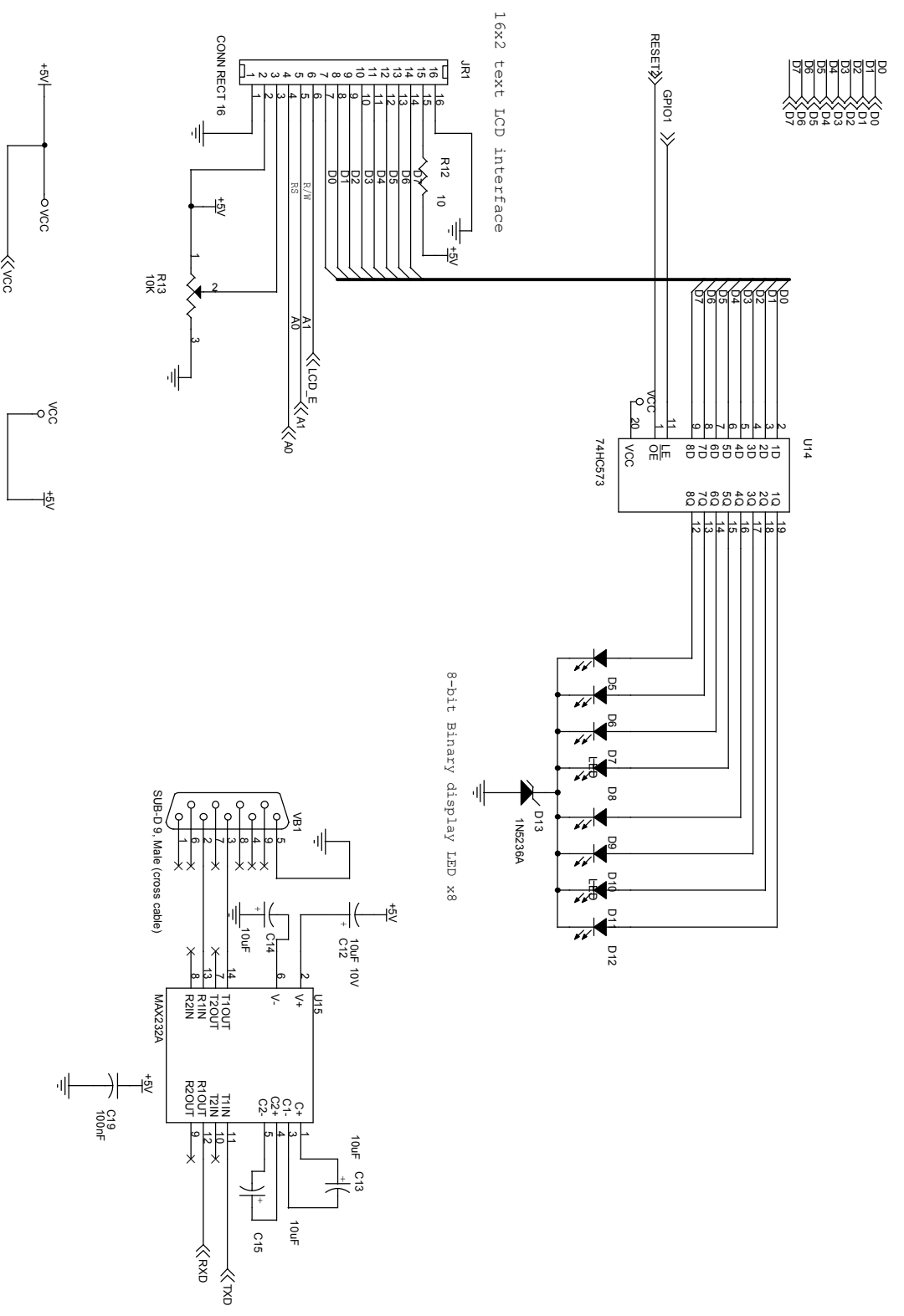
## **HARDWARE SCHEMATIC and PARTS LIST**





Title		6502 MICROPROCESSOR KIT	
Size	Document Number	Sheet	Rev
B	<Doc>	2	1
Date	Saturday, January 03, 2015	of	3

Designed by Wichtl Sirichote, kawichit@kmitl.ac.th, 2014



Title		6502 MICROPROCESSOR KIT	
Size	Document Number	Sheet	3 of 3
B	<Doc>	3	3
Date:	Saturday, January 03, 2015	Rev	<Rev Code>

Designed by Michl Strohner, kwstich@kml.ac.th, 2014

## PARTS LIST

### Semiconductors

U1 27C256, 32kB EPROM  
U2 HM62256B, 32kB Static RAM  
U3 AT89C2051, 20-pin DIP microcontroller  
U4 GAL16V8D, programmable logic device  
U5 65SC02 CMOS CPU  
U6 74HC164, shift register  
U7 74LS14, inverter  
U8 LM2490-5.0, voltage regulator  
U10,U9 LTC-4727JR, seven segment LED  
U11,U12,U14 74HC573, data latch  
U13 74HC541, tristate buffer  
U15 MAX232A, RS232 level converter  
D1,D5,D6,D7,D8,D9,D10, 3mm LED  
D11,D12  
D2 TVS5V\_SOD123 transient voltage suppressor  
D3 1N4007 rectifying diode  
D4 3mm LED  
D13 1N5236A  
Q1 BC327 PNP transistor

### Resistors (all resistors are 1/8W +/-5%)

R1 680  
R2,R3,R5 1k  
R8,R4 4.7k  
R13,R6 10k  
R11,R7 10k RESISTOR SIP 9  
R9 330  
R12,R10 10

### Capacitors

C1 22uF electrolytic  
C2,C13,C14,C15,C16,C17 10uF electrolytic  
C3 10uF 16V electrolytic

C4 1000uF25V electrolytic  
C5,C6,C7,C8,C9 0.1uF disc ceramic  
C10,C11 0.1uF disc ceramic  
C12 10uF 10V electrolytic  
C19,C18 100nF disc ceramic

### Additional parts

JJP1 HEADER 20X2  
JR1 CONN RECT 16 text LCD connector  
J1 DC Input  
J2 CON3 3-pin header  
LS1 SPEAKER

SW1 ESP switch  
SW2 IRQ  
SW3 RESET  
SW4 user  
S1,S2,S3,S4,S5,S6,S7,S8,  
S9,S10,S11,S12,S13,S14,Tact switch  
S15,S16,S17,S18,S19,S20,  
S21,S22,S23,S24,S25,S26,  
S27,S28,S29,S30,S31,S32,  
S33  
TP1,TP2,TP3 TEST POINT  
TP4 +5V  
TP5 GND

VB1 SUB-D 9, Male (cross cable)  
Y1-OSC-1M-MEC1 OSC14  
PCB double side plate through hole  
LED cover Clear RED color acrylic plastic  
Keyboard sticker printable SVG file

## **MONITOR PROGRAM LISTINGS**

```
0001 0000 ;-----
0002 0000 ; Monitor program source code for 6502 Microprocessor Kit
0003 0000 ; Written by Wichit Sirichote, kswichit@kmitl.ac.th
0004 0000 ; Copyright (c) 2015
0005 0000 ;
0006 0000 ; Source code was assembled with tasm assembler
0007 0000 ; Example of using tasm
0008 0000 ;
0009 0000 ; d:\tasm\tasm -65 monitor.asm
0010 0000 ;
0011 0000 ; The object file will be Intel hex file ready for EPROM programmer
0012 0000 ; The physical location is the 2nd block of 64kB space.
0013 0000 ; To program the 32kB EPROM, we must move it to the 1st block.
0014 0000 ;
0015 0000 ;
0016 0000 ;
0017 0000 ; 26 DECEMBER 2014
0018 0000 ; 27 DECEMBER 2014
0019 0000 ; -ADD OUT OF RANGE CHECK FOR RELATIVE BYTE CALCULATION
0020 0000 ; -ADD DOWNLOAD HEX FILE TO MONITOR SOURCE
0021 0000 ; 29 DECEMBER 2014
0022 0000 ; -ADD START MESSAGE ON COLD BOOT
0023 0000 ; 30 DECEMBER 2014
0024 0000 ; -TEST SINGLE STEP WITH 74LS164
0025 0000 ; - ADD REPEAT KEY
0026 0000 ; - LOWER REPEAT SPEED
0027 0000 ; - REMOVE ACCUMALATOR DISPLAY ON BREAK
0028 0000 ; 2 JANUARY 2015
0029 0000 ; - REMOVE BINARY DISPLAY IN REGISTER MODE
0030 0000 ; - ADD REGISTER MODE DISPLAY FOR 10 BYTES ZERO PAGE, $00 TO $09
0031 0000 ; 3 JANUARY 2015
0032 0000 ; - PROVIDE PROGRAM COUNTER SAVING FOR SINGLE STEP RUNNING
0033 0000 ; now user may change display address, data, to get back current ad
0034 0000 ; press PC key to restore it, then press step
0035 0000 ;
0036 0000 ; 27 February 2015
0037 0000 ; lower brightness of the 7-segment
0038 0000 ; calibrate beep frequency to 523Hz
0039 0000 ; 2 March 2015
0040 0000 ; add hex file download using MOS hex format
0041 0000 ; now the board can accept both Intel and MOS hex file format. Tested
0042 0000 ; no error, the GPIO1 will display 0D (CR), if error it will show 01.
0043 0000 ;
0044 0000 ; 21 March fix cold boot message, make beep on cold boot
0045 0000
0046 0000
0047 0000
0048 0000
0049 0000 ; address of the I/O ports
0050 0000
0051 0000 GPIO1 .EQU 8000H
0052 0000 PORT0 .EQU 8001H
0053 0000 PORT1 .EQU 8002H
0054 0000 PORT2 .EQU 8003H
0055 0000
0056 0000 DIGIT .EQU 8002H
0057 0000 SEG7 .EQU 8003H
0058 0000 KIN .EQU 8001H
0059 0000
0060 0000
0061 0000
0062 0000
0063 0000 ; page zero register definition
0064 0000 ; LOCATION $00 TO $7F ARE 128 BYTES FOR USER PROGRAM TESTING
0065 0000
0066 0000 .DSEG
0067 0080 .ORG 80H
0068 0080
0069 0080 ; zero page memory definitions for monitor use
0070 0080 REG_E .BLOCK 1
0071 0081 REG_D .BLOCK 1
0072 0082 REG_B .BLOCK 1
0073 0083 REG_C .BLOCK 1
0074 0084 HL .BLOCK 2 ; 84H = L 85H = H
0075 0086 DE .BLOCK 2
0076 0088 REG_A .BLOCK 1
```



```

0077 0089
0078 0089      _ERROR  .BLOCK 1      ; ERROR FLAG FOR INTEL HEX FILE DOWNLOADING
0079 008A      BCC      .BLOCK 2      ; BYTE CHECK SUM
0080 008C      BUFFER  .BLOCK 6      ; 8BH - 90H PAGE ZERO DISPLAY BUFFER
0081 0092      INVALID .BLOCK 1      ; INVALID KEY HAS BEEN PRESSED FLAG BIT
0082 0093      ; 0 VALID
0083 0093      ; 1 INVALID
0084 0093
0085 0093      KEY      .BLOCK 1
0086 0094      STATE   .BLOCK 1
0087 0095      ZERO_FLAG .BLOCK 1      ; ZERO WHEN HEX KEY PRESSED FOR ADDRESS OR
0088 0096
0089 0096      DISPLAY .BLOCK 2      ; display address
0090 0098
0091 0098      PC_USER  .BLOCK 2      ; FOR SAVING CURRENT PC, ON RESET, IT SETS TO
0092 009A      USER_A  .BLOCK 1
0093 009B      USER_X  .BLOCK 1
0094 009C      USER_Y  .BLOCK 1
0095 009D      USER_S  .BLOCK 1      ; USER STACK POINTER
0096 009E      USER_P  .BLOCK 1      ; PROGRAM STATUS REGISTER
0097 009F      SAVE_SP .BLOCK 1      ; SAVE SYSTEM STACK
0098 00A0
0099 00A0      START_ADDRESS .BLOCK 2
0100 00A2      DESTINATION .BLOCK 2      ; FOR OFFSET BYTE CALCULATION
0101 00A4      OFFSET_BYTE .BLOCK 2      ; OFFSET BYTE = DESTINATION - START_ADDRESS
0102 00A6      COLD     .BLOCK 1      ; COLD BOOT OR WARM BOOT
0103 00A7
0104 00A7      REPDELAY .BLOCK 1
0105 00A8      SAVE_X   .BLOCK 1
0106 00A9      SAVE_Y   .BLOCK 1
0107 00AA
0108 00AA      DEBUG    .BLOCK 2      ; FOR PROGRAM DEBUGGING
0109 00AC
0110 00AC
0111 00AC
0112 00AC      .CSEG
0113 00AC
0114 C000      .ORG 0C000H      ; START ADDRESS FOR ROM
0115 C000      ; .ORG 1000H      ; START ADDRESS FOR CODE TESTING IN RAM
0116 C000
0117 C000 A9 BF      LDA #$BF ; turn off break signal
0118 C002 8D 02 80      STA PORT1
0119 C005 A9 00      LDA #0
0120 C007 8D 03 80      STA PORT2 ; turn of 7-segment
0121 C00A
0122 C00A      ; power up delay
0123 C00A
0124 C00A A2 00      LDX #0
0125 C00C CA      POWER_UP_DELAY DEX
0126 C00D D0 FD      BNE POWER_UP_DELAY
0127 C00F
0128 C00F      ; jump to main code
0129 C00F
0130 C00F 4C 72 C8      JMP MAIN
0131 C012
0132 C012      ;----- 2400 BIT/S SOFTWARE UART -----
0133 C012      ; one bit delay for 2400 bit/s UART
0134 C012
0135 C012 A0 4C      BIT_DELAY LDY #76      ; 1190 Hz TEST AT 1MHZ OSCILLATOR
0136 C014 88      LOOP      DEY
0137 C015 D0 FD      BNE LOOP
0138 C017 60      RTS
0139 C018
0140 C018      ; 1.5 bit delay
0141 C018
0142 C018 A0 72      BIT1_5_DELAY LDY #114      ; DELAY 1.5 BIT
0143 C01A 88      LOOP1      DEY
0144 C01B D0 FD      BNE LOOP1
0145 C01D 60      RTS
0146 C01E
0147 C01E      ; SEND ASCII LETTER TO TERMINAL
0148 C01E      ; ENTRY: A
0149 C01E
0150 C01E 85 80      SEND_BYTE: STA REG_E ; SAVE ACCUMULATOR
0151 C020
0152 C020 A9 3F      LDA #3FH ; start bit is zero

```

```
0153 C022 8D 02 80 STA PORT1
0154 C025 20 12 C0 JSR BIT_DELAY ; delay one bit
0155 C028
0156 C028 A9 08 LDA #8 ; 8-data bit wil be sent
0157 C02A 85 81 STA REG_D
0158 C02C
0159 C02C A5 80 CHK_BIT: LDA REG_E
0160 C02E 29 01 AND #1
0161 C030 F0 08 BEQ SEND_ZERO
0162 C032
0163 C032 A9 BF LDA #0BFH
0164 C034 8D 02 80 STA PORT1
0165 C037
0166 C037 4C 42 C0 JMP NEXT_BIT
0167 C03A
0168 C03A
0169 C03A A9 3F SEND_ZERO: LDA #3FH
0170 C03C 8D 02 80 STA PORT1
0171 C03F 4C 42 C0 JMP NEXT_BIT
0172 C042
0173 C042 20 12 C0 NEXT_BIT: JSR BIT_DELAY
0174 C045
0175 C045 46 80 LSR REG_E
0176 C047 C6 81 DEC REG_D
0177 C049 D0 E1 BNE CHK_BIT
0178 C04B
0179 C04B A9 BF LDA #0BFH
0180 C04D 8D 02 80 STA PORT1
0181 C050 20 12 C0 JSR BIT_DELAY
0182 C053 60 RTS
0183 C054
0184 C054
0185 C054 ; RECEIVE BYTE FROM 2400 BIT/S TERMINAL
0186 C054 ; EXIT: A
0187 C054
0188 C054 AD 01 80 CIN LDA PORT0
0189 C057 29 80 AND #80H
0190 C059 D0 F9 BNE CIN
0191 C05B
0192 C05B 20 18 C0 JSR BIT1_5_DELAY
0193 C05E
0194 C05E A9 07 LDA #7
0195 C060 85 81 STA REG_D
0196 C062 A9 00 LDA #0
0197 C064 85 80 STA REG_E
0198 C066
0199 C066
0200 C066
0201 C066 AD 01 80 CHK_BIT_RX LDA PORT0
0202 C069 29 80 AND #80H
0203 C06B D0 09 BNE BIT_IS_ONE
0204 C06D
0205 C06D A5 80 LDA REG_E
0206 C06F 29 7F AND #7FH
0207 C071 85 80 STA REG_E
0208 C073 4C 7F C0 JMP NEXT_BIT_RX
0209 C076
0210 C076 A5 80 BIT_IS_ONE LDA REG_E
0211 C078 09 80 ORA #80H
0212 C07A 85 80 STA REG_E
0213 C07C 4C 7F C0 JMP NEXT_BIT_RX
0214 C07F
0215 C07F 20 12 C0 NEXT_BIT_RX JSR BIT_DELAY
0216 C082
0217 C082 46 80 LSR REG_E
0218 C084
0219 C084 C6 81 DEC REG_D
0220 C086 D0 DE BNE CHK_BIT_RX
0221 C088
0222 C088 20 12 C0 JSR BIT_DELAY ; CENTER OF STOP BIT
0223 C08B
0224 C08B A5 80 LDA REG_E
0225 C08D
0226 C08D 60 RTS
0227 C08E
0228 C08E
```

```

0229 C08E          ; PRINT TEXT FROM STRING AREA
0230 C08E          ; ENTRY: X POINTED TO OFFSET
0231 C08E
0232 C08E BD 00 EF PSTRING LDA TEXT1,X
0233 C091 C9 00          CMP #0
0234 C093 D0 01          BNE PRINT_IT
0235 C095 60          RTS
0236 C096
0237 C096 20 1E C0 PRINT_IT JSR SEND_BYTE
0238 C099 E8          INX
0239 C09A 4C 8E C0      JMP PSTRING
0240 C09D
0241 C09D          CR          .EQU 0DH
0242 C09D          LF          .EQU 0AH
0243 C09D          EOS          .EQU 0
0244 C09D
0245 C09D          ;NEW LINE
0246 C09D          ; PRINT CR, LF
0247 C09D
0248 C09D A9 0D NEW_LINE LDA #0DH
0249 C09F 20 1E C0      JSR SEND_BYTE
0250 C0A2 A9 0A          LDA #0AH
0251 C0A4 20 1E C0      JSR SEND_BYTE
0252 C0A7 60          RTS
0253 C0A8
0254 C0A8
0255 C0A8          ; WRITE NIBBLE TO TERMINAL
0256 C0A8 29 0F OUT1X AND #0FH
0257 C0AA 18          CLC
0258 C0AB 69 30          ADC #30H
0259 C0AD C9 3A          CMP #3AH
0260 C0AF 90 03          BCC OUT1X1
0261 C0B1 18          CLC
0262 C0B2 69 07          ADC #7
0263 C0B4 20 1E C0 OUT1X1 JSR SEND_BYTE
0264 C0B7 60          RTS
0265 C0B8
0266 C0B8
0267 C0B8 48 OUT2X PHA
0268 C0B9
0269 C0B9 4A          LSR A
0270 C0BA 4A          LSR A
0271 C0BB 4A          LSR A
0272 C0BC 4A          LSR A
0273 C0BD
0274 C0BD          ; STA GPIO1
0275 C0BD
0276 C0BD 20 A8 C0      JSR OUT1X
0277 C0C0 68          PLA
0278 C0C1 20 A8 C0      JSR OUT1X
0279 C0C4 60          RTS
0280 C0C5
0281 C0C5
0282 C0C5          ; INCREMENT HL
0283 C0C5          ; INCREMENT 16-BIT POINTER FOR 16-BIT MEMORY ACCESS
0284 C0C5
0285 C0C5 18 INC_HL CLC
0286 C0C6 A5 84          LDA HL
0287 C0C8 69 01          ADC #1
0288 C0CA 85 84          STA HL
0289 C0CC A5 85          LDA HL+1
0290 C0CE 69 00          ADC #0
0291 C0D0 85 85          STA HL+1
0292 C0D2 60          RTS
0293 C0D3
0294 C0D3
0295 C0D3          ; PRINT LINE OF MEMORY POINTED TO HL
0296 C0D3
0297 C0D3 20 9D C0 PRINT_LINE JSR NEW_LINE
0298 C0D6 A9 10          LDA #16
0299 C0D8 85 83          STA REG_C
0300 C0DA
0301 C0DA
0302 C0DA
0303 C0DA A5 85          LDA HL+1
0304 C0DC 20 B8 C0      JSR OUT2X

```

```

0305  CODF A5 84          LDA HL
0306  COE1 20 B8 C0      JSR OUT2X
0307  COE4
0308  COE4 A9 3A          LDA #' ':'
0309  COE6 20 1E C0      JSR SEND_BYTE
0310  COE9
0311  COE9 A0 00          PRINT_LINE2 LDY #0
0312  COEB B1 84          LDA (HL),Y
0313  COED
0314  COED 20 B8 C0      JSR OUT2X
0315  COF0
0316  COF0 A9 20          LDA #' '
0317  COF2 20 1E C0      JSR SEND_BYTE
0318  COF5
0319  COF5 20 C5 C0      JSR INC_HL
0320  COF8
0321  COF8 C6 83          DEC REG_C
0322  COFA
0323  COFA D0 ED          BNE PRINT_LINE2
0324  COFC
0325  COFC 60            RTS
0326  COFD
0327  COFD                ; CONVERT ASCII TO HEX
0328  COFD                ; ENTRY: A
0329  COFD
0330  COFD 38            TO_HEX      SEC
0331  COFE E9 30          SBC #30H
0332  C100 C9 10          CMP #10H
0333  C102 90 05          BCC ZERO_NINE
0334  C104 29 DF          AND #11011111B
0335  C106 38            SEC
0336  C107 E9 07          SBC #7
0337  C109
0338  C109 60            ZERO_NINE RTS
0339  C10A
0340  C10A                ; CONVERT TWO ASCII LETTERS TO SINGLE BYTE
0341  C10A                ; EXIT: A
0342  C10A
0343  C10A 20 54 C0      GET_HEX    JSR CIN
0344  C10D 20 FD C0      JSR TO_HEX
0345  C110 0A            ASL A
0346  C111 0A            ASL A
0347  C112 0A            ASL A
0348  C113 0A            ASL A
0349  C114
0350  C114 8D 00 80      STA GPIO1
0351  C117
0352  C117 85 88          STA REG_A
0353  C119
0354  C119 20 54 C0      JSR CIN
0355  C11C 20 FD C0      JSR TO_HEX
0356  C11F 18            CLC
0357  C120 65 88          ADC REG_A
0358  C122
0359  C122 60            RTS
0360  C123
0361  C123                ; CONVERT TWO ASCII LETTERS TO SINGLE BYTE
0362  C123                ; EXIT: A
0363  C123
0364  C123 20 54 C0      GET_HEX2   JSR CIN
0365  C126 48            PHA
0366  C127 20 1E C0      JSR SEND_BYTE ; ECHO TO TERMINAL
0367  C12A 68            PLA
0368  C12B 20 FD C0      JSR TO_HEX
0369  C12E 0A            ASL A
0370  C12F 0A            ASL A
0371  C130 0A            ASL A
0372  C131 0A            ASL A
0373  C132
0374  C132 8D 00 80      STA GPIO1
0375  C135
0376  C135 85 88          STA REG_A
0377  C137
0378  C137 20 54 C0      JSR CIN
0379  C13A 48            PHA
0380  C13B 20 1E C0      JSR SEND_BYTE

```

```

0381 C13E 68          PLA
0382 C13F 20 FD C0   JSR TO_HEX
0383 C142 18          CLC
0384 C143 65 88      ADC REG_A
0385 C145
0386 C145 60          RTS
0387 C146
0388 C146          ;-----
0389 C146          SET_NEW_ADDRESS
0390 C146
0391 C146 20 1E C0   JSR SEND_BYTE
0392 C149 A2 1C      LDX #PROMPT&00FFH
0393 C14B 20 8E C0   JSR PSTRING
0394 C14E 20 23 C1   JSR GET_HEX2
0395 C151 85 85      STA HL+1
0396 C153 20 23 C1   JSR GET_HEX2
0397 C156 85 84      STA HL
0398 C158 60          RTS
0399 C159
0400 C159 18          ADD_BCC CLC
0401 C15A 65 8A      ADC BCC
0402 C15C 85 8A      STA BCC
0403 C15E 60          RTS
0404 C15F
0405 C15F          ;-----
0406 C15F          ; GET_RECORD READS INTEL HEX FILE AND SAVE TO MEMORY
0407 C15F
0408 C15F A9 00      GET_RECORD LDA #0
0409 C161 85 89      STA _ERROR
0410 C163
0411 C163 20 54 C0   GET_RECORD1 JSR CIN
0412 C166 C9 3A      CMP #' ':'
0413 C168 F0 07      BEQ GET_RECORD2
0414 C16A
0415 C16A C9 3B      CMP #$3B          ; ';'
0416 C16C D0 F5      BNE GET_RECORD1
0417 C16E
0418 C16E 4C ED C1   JMP GET_MOS2
0419 C171
0420 C171
0421 C171          GET_RECORD2
0422 C171
0423 C171 A9 00      LDA #0
0424 C173 85 8A      STA BCC
0425 C175
0426 C175 20 0A C1   JSR GET_HEX
0427 C178 85 83      STA REG_C          ; GET NUMBER OF BYTE
0428 C17A
0429 C17A 20 59 C1   JSR ADD_BCC
0430 C17D
0431 C17D 20 0A C1   JSR GET_HEX
0432 C180 85 85      STA HL+1
0433 C182
0434 C182 20 59 C1   JSR ADD_BCC
0435 C185
0436 C185 20 0A C1   JSR GET_HEX
0437 C188 85 84      STA HL          ; GET LOAD ADDRESS
0438 C18A
0439 C18A 20 59 C1   JSR ADD_BCC
0440 C18D
0441 C18D 20 0A C1   JSR GET_HEX
0442 C190
0443 C190 C9 00      CMP #0
0444 C192
0445 C192 F0 14      BEQ DATA_RECORD
0446 C194
0447 C194 20 54 C0   WAIT_CR JSR CIN
0448 C197 C9 0D      CMP #0DH
0449 C199 D0 F9      BNE WAIT_CR
0450 C19B
0451 C19B 8D 00 80   STA GPIO1
0452 C19E
0453 C19E A5 89      LDA _ERROR
0454 C1A0 C9 01      CMP #1
0455 C1A2 D0 03      BNE NOERROR
0456 C1A4

```

```

0457 C1A4          ; SHOW ERROR ON LED
0458 C1A4          ; JSR OUT_OFF_RANGE
0459 C1A4
0460 C1A4 8D 00 80      STA GPIO1
0461 C1A7
0462 C1A7          NOERROR
0463 C1A7 60          RTS
0464 C1A8
0465 C1A8          DATA_RECORD
0466 C1A8
0467 C1A8 20 0A C1      JSR GET_HEX
0468 C1AB A0 00        LDY #0
0469 C1AD 91 84        STA (HL),Y ; WRITE TO MEMORY
0470 C1AF
0471 C1AF 20 59 C1      JSR ADD_BCC
0472 C1B2
0473 C1B2 8D 00 80      STA GPIO1
0474 C1B5
0475 C1B5 20 C5 C0      JSR INC_HL
0476 C1B8
0477 C1B8 C6 83        DEC REG_C
0478 C1BA D0 EC        BNE DATA_RECORD ; UNTIL C=0
0479 C1BC
0480 C1BC A5 8A          LDA BCC
0481 C1BE 49 FF        EOR #0FFH ; ONE'S COMPLEMENT
0482 C1C0 18          CLC
0483 C1C1 69 01        ADC #1 ; TWO'S COMPLEMENT
0484 C1C3 85 8A        STA BCC
0485 C1C5
0486 C1C5
0487 C1C5 20 0A C1      JSR GET_HEX ; GET BYTE CHECK SUM
0488 C1C8
0489 C1C8 C5 8A        CMP BCC ; COMPARE WITH BYTE CHECK SUM
0490 C1CA F0 04        BEQ SKIP11
0491 C1CC
0492 C1CC A9 01        LDA #1
0493 C1CE 85 89        STA _ERROR ; ERROR FLAG =1
0494 C1D0
0495 C1D0
0496 C1D0          SKIP11
0497 C1D0
0498 C1D0 4C 63 C1      JMP GET_RECORD1 ; NEXT LINE
0499 C1D3
0500 C1D3
0501 C1D3
0502 C1D3          SEND_PROMPT
0503 C1D3
0504 C1D3 20 9D C0        JSR NEW_LINE
0505 C1D6 A5 85          LDA HL+1
0506 C1D8 20 B8 C0      JSR OUT2X
0507 C1DB A5 84          LDA HL
0508 C1DD 20 B8 C0      JSR OUT2X
0509 C1E0
0510 C1E0
0511 C1E0 A2 1C          LDX #PROMPT&00FFH
0512 C1E2 20 8E C0      JSR PSTRING
0513 C1E5 60          RTS
0514 C1E6
0515 C1E6          ;=====
0516 C1E6          ; get MOS record
0517 C1E6          ; sample MOS record
0518 C1E6          ;
0519 C1E6          ;18 0200 A9018500182600A5008D0080A200A00088D0FDCAD0F84C05 09B3
0520 C1E6          ;01 0218 02 001D
0521 C1E6          ;00
0522 C1E6          ;
0523 C1E6          ; 18 is number of byte
0524 C1E6          ; 0200 is load address
0525 C1E6          ; A9, 01, 85.. data byte
0526 C1E6          ; 09B3 is 16-bit check sum
0527 C1E6
0528 C1E6
0529 C1E6
0530 C1E6          GET_MOS1
0531 C1E6 20 54 C0      JSR CIN
0532 C1E9 C9 3B        CMP #$3B ; ';'

```

```

0533 C1EB D0 F9          BNE GET_MOS1
0534 C1ED
0535 C1ED              GET_MOS2
0536 C1ED
0537 C1ED A9 00        LDA #0
0538 C1EF 85 8A        STA BCC
0539 C1F1 85 8B        STA BCC+1 ; MOS uses 16-bit checksum
0540 C1F3
0541 C1F3
0542 C1F3 20 0A C1      JSR GET_HEX
0543 C1F6 85 83        STA REG_C ; GET NUMBER OF BYTE
0544 C1F8
0545 C1F8 C9 00        CMP #0
0546 C1FA F0 16        BEQ END_RECORD
0547 C1FC
0548 C1FC 20 5A C2      JSR ADD_BCC_MOS
0549 C1FF
0550 C1FF 20 0A C1      JSR GET_HEX
0551 C202 85 85        STA HL+1
0552 C204
0553 C204 20 5A C2      JSR ADD_BCC_MOS
0554 C207
0555 C207 20 0A C1      JSR GET_HEX
0556 C20A 85 84        STA HL ; GET LOAD ADDRESS
0557 C20C
0558 C20C 20 5A C2      JSR ADD_BCC_MOS
0559 C20F
0560 C20F 4C 26 C2      JMP DATA_RECORD2
0561 C212
0562 C212 20 54 C0      END_RECORD JSR CIN
0563 C215 C9 0D        CMP #0DH
0564 C217 D0 F9        BNE END_RECORD
0565 C219
0566 C219 8D 00 80      STA GPIO1
0567 C21C
0568 C21C A5 89        LDA _ERROR
0569 C21E C9 01        CMP #1
0570 C220 D0 03        BNE NOERROR2
0571 C222
0572 C222              ; SHOW ERROR ON LED
0573 C222
0574 C222 8D 00 80      STA GPIO1
0575 C225
0576 C225
0577 C225              NOERROR2
0578 C225 60          RTS
0579 C226
0580 C226              DATA_RECORD2
0581 C226
0582 C226 20 0A C1      JSR GET_HEX
0583 C229 A0 00        LDY #0
0584 C22B 91 84        STA (HL),Y ; WRITE TO MEMORY
0585 C22D
0586 C22D 20 5A C2      JSR ADD_BCC_MOS
0587 C230
0588 C230 8D 00 80      STA GPIO1
0589 C233
0590 C233 20 C5 C0      JSR INC_HL
0591 C236
0592 C236 C6 83        DEC REG_C
0593 C238 D0 EC        BNE DATA_RECORD2 ; UNTIL C=0
0594 C23A
0595 C23A              ; now get 16-bit check sum
0596 C23A
0597 C23A 20 0A C1      JSR GET_HEX ; GET 16-bit CHECK SUM
0598 C23D 85 85        STA HL+1
0599 C23F              ; STA DEBUG+1
0600 C23F
0601 C23F 20 0A C1      JSR GET_HEX
0602 C242 85 84        STA HL ; check sum now stored in HL+1 and HL
0603 C244              ; STA DEBUG
0604 C244
0605 C244 A5 8B        LDA BCC+1
0606 C246 C5 85        CMP HL+1
0607 C248 D0 09        BNE error_mos
0608 C24A A5 8A        LDA BCC

```

```

0609 C24C C5 84          CMP HL
0610 C24E D0 03          BNE error_mos
0611 C250
0612 C250 4C 57 C2        JMP SKIP12
0613 C253
0614 C253          error_mos
0615 C253 A9 01          LDA #1
0616 C255 85 89          STA _ERROR      ; ERROR FLAG =1
0617 C257
0618 C257
0619 C257          SKIP12
0620 C257
0621 C257 4C E6 C1        JMP GET_MOS1      ; NEXT LINE
0622 C25A
0623 C25A
0624 C25A          ; add 16-bit check sum, stores in BCC+1 and BCC
0625 C25A
0626 C25A          ADD_BCC_MOS
0627 C25A
0628 C25A 18          CLC
0629 C25B 65 8A          ADC BCC
0630 C25D 85 8A          STA BCC
0631 C25F A9 00          LDA #0
0632 C261 65 8B          ADC BCC+1
0633 C263 85 8B          STA BCC+1
0634 C265 60          RTS
0635 C266
0636 C266
0637 C266          ;----- END UART CODE -----
0638 C266
0639 C266          ; SCAN DISPLAY ONLY
0640 C266          ; ENTRY: X POINTED TO NEXT MESSAGE BYTE
0641 C266          ;      FIX_MESSAGE LOCATION
0642 C266
0643 C266          SCAN2:
0644 C266 86 83          STX REG_C
0645 C268 A9 01          LDA #1
0646 C26A 85 80          STA REG_E
0647 C26C
0648 C26C A9 06          LDA #6
0649 C26E 85 84          STA HL
0650 C270
0651 C270          ;to the active column.
0652 C270 A5 80          KCOL2  LDA REG_E
0653 C272
0654 C272 49 FF          EOR #0FFH      ; COMPLEMENT IT
0655 C274
0656 C274 29 BF          AND #0BFH      ; BREAK MUST BE LOGIC '0' TO DISABLE
0657 C276 8D 02 80        STA DIGIT
0658 C279
0659 C279 BD E3 C8        LDA START_MSG,X
0660 C27C 8D 03 80        STA SEG7
0661 C27F
0662 C27F A0 05          LDY #$5
0663 C281 88          DELAY5 DEY
0664 C282 D0 FD          BNE DELAY5
0665 C284
0666 C284 A9 00          LDA #0      ; TURN LED OFF
0667 C286 8D 03 80        STA SEG7
0668 C289
0669 C289
0670 C289
0671 C289
0672 C289 E8          INX
0673 C28A
0674 C28A A5 80          LDA REG_E
0675 C28C 0A          ASL A
0676 C28D 85 80          STA REG_E
0677 C28F
0678 C28F C6 84          DEC HL
0679 C291 D0 DD          BNE KCOL2
0680 C293
0681 C293 A6 83          LDX REG_C
0682 C295
0683 C295 60          RTS
0684 C296

```



```

0685 C296
0686 C296 ; SCAN DISPLAY AND KEYBOARD
0687 C296 ; ENTRY: DISPLAY BUFFER IN PAGE 0
0688 C296 ; EXIT: KEY = -1 NO KEY PRESSED
0689 C296 ; KEY >=0 KEY POSITION
0690 C296 ; REGSITERS USED: X,A,Y
0691 C296
0692 C296 SCAN1:
0693 C296
0694 C296
0695 C296 A2 00 LDX #0
0696 C298
0697 C298 A9 00 LDA #0
0698 C29A 85 83 STA REG_C
0699 C29C
0700 C29C A9 FF LDA #-1
0701 C29E 85 93 STA KEY
0702 C2A0
0703 C2A0 A9 01 LDA #1
0704 C2A2 85 80 STA REG_E
0705 C2A4
0706 C2A4 A9 06 LDA #6
0707 C2A6 85 84 STA HL
0708 C2A8
0709 C2A8 ;to the active column.
0710 C2A8 A5 80 KCOL LDA REG_E
0711 C2AA
0712 C2AA 49 FF EOR #0FFH ; COMPLEMENT IT
0713 C2AC 29 BF AND #0BFH ; MUST BE LOW FOR BREAK
0714 C2AE
0715 C2AE 8D 02 80 STA DIGIT
0716 C2B1
0717 C2B1 B5 8C LDA BUFFER,X
0718 C2B3 8D 03 80 STA SEG7
0719 C2B6
0720 C2B6 A0 30 LDY #$30
0721 C2B8 88 DELAY3 DEY
0722 C2B9 D0 FD BNE DELAY3
0723 C2BB
0724 C2BB A9 00 LDA #0 ; TURN LED OFF
0725 C2BD 8D 03 80 STA SEG7
0726 C2C0
0727 C2C0 A0 32 LDY #50
0728 C2C2 88 DELAY10 DEY
0729 C2C3 D0 FD BNE DELAY10
0730 C2C5
0731 C2C5
0732 C2C5 A9 06 LDA #6
0733 C2C7 85 82 STA REG_B
0734 C2C9
0735 C2C9 AD 01 80 LDA KIN
0736 C2CC
0737 C2CC 85 81 STA REG_D
0738 C2CE
0739 C2CE
0740 C2CE 46 81 KROW LSR REG_D ;Rotate D 1 bit right, bit 0
0741 C2D0 ;of D will be rotated into
0742 C2D0 B0 04 BCS NOKEY ;carry flag.
0743 C2D2
0744 C2D2 A5 83 LDA REG_C
0745 C2D4 85 93 STA KEY
0746 C2D6
0747 C2D6 E6 83 NOKEYINC REG_C ;Increase current key-code by 1.
0748 C2D8
0749 C2D8 C6 82 DEC REG_B
0750 C2DA D0 F2 BNE KROW
0751 C2DC
0752 C2DC E8 INX
0753 C2DD
0754 C2DD A5 80 LDA REG_E
0755 C2DF 0A ASL A
0756 C2E0 85 80 STA REG_E
0757 C2E2
0758 C2E2
0759 C2E2 C6 84 DEC HL
0760 C2E4 D0 C2 BNE KCOL

```

```

0761 C2E6 60          RTS
0762 C2E7
0763 C2E7
0764 C2E7 A0 C8      DEBOUNCE LDY #200
0765 C2E9 88          DELAY4 DEY
0766 C2EA D0 FD      BNE DELAY4
0767 C2EC 60          RTS
0768 C2ED
0769 C2ED             ;-----
0770 C2ED
0771 C2ED 20 96 C2    SCANKEY JSR SCAN1
0772 C2F0 A5 93          LDA KEY
0773 C2F2 C9 FF          CMP #-1
0774 C2F4 F0 16          BEQ KEY_RELEASED
0775 C2F6
0776 C2F6 AD 01 80     LDA PORT0
0777 C2F9 29 40          AND #40H
0778 C2FB D0 F0          BNE SCANKEY
0779 C2FD
0780 C2FD             ; IF REPEAT KEY WAS PRESSED, SLOW DOWN IT
0781 C2FD A9 20          LDA #20H
0782 C2FF 85 A7          STA REPDELAY
0783 C301
0784 C301 20 96 C2     DISPLAY4 JSR SCAN1
0785 C304 C6 A7          DEC REPDELAY
0786 C306 D0 F9          BNE DISPLAY4
0787 C308
0788 C308
0789 C308 A2 00          LDX #0           ; THEN REPEAT KEY PRESS
0790 C30A 86 92          STX INVALID      ; RESET INVALID FLAG
0791 C30C             KEY_RELEASED
0792 C30C
0793 C30C 20 E7 C2     JSR DEBOUNCE
0794 C30F
0795 C30F             UNTIL_PRESS
0796 C30F
0797 C30F 20 96 C2     JSR SCAN1
0798 C312 A5 93          LDA KEY
0799 C314 C9 FF          CMP #-1
0800 C316 F0 F7          BEQ UNTIL_PRESS
0801 C318
0802 C318 20 E7 C2     JSR DEBOUNCE
0803 C31B
0804 C31B 20 96 C2     JSR SCAN1
0805 C31E
0806 C31E A5 93          LDA KEY
0807 C320 AA            TAX
0808 C321 BD FF C8      LDA KEYTAB,X     ; OPEN TABLE
0809 C324
0810 C324             ;STA GPIO1           ; TEST NOW A IS INTERNAL CODE
0811 C324 60          RTS
0812 C325
0813 C325
0814 C325             ; CONVERT LOW NIBBLE IN ACCUMULATOR TO 7-SEGMENT PATTERN
0815 C325             ; ENTRY: A
0816 C325             ; EXIT: A
0817 C325
0818 C325             NIBBLE_7SEG
0819 C325 AA            TAX
0820 C326 BD EF C8      LDA SEGTAB,X
0821 C329 60          RTS
0822 C32A
0823 C32A
0824 C32A             ; CONVERT BYTE TO 7-SEGMENT PATTERN
0825 C32A             ; ENTRY: A
0826 C32A             ; EXIT: DE
0827 C32A
0828 C32A 48           BYTE_7SEG PHA
0829 C32B 29 0F          AND #0FH
0830 C32D 20 25 C3     JSR NIBBLE_7SEG
0831 C330 85 86          STA DE
0832 C332 68            PLA
0833 C333 4A            LSR A
0834 C334 4A            LSR A
0835 C335 4A            LSR A
0836 C336 4A            LSR A

```



```
0913 C39A C9 08      CHK_STATE8  CMP #8
0914 C39C D0 03      BNE CHK_STATE9
0915 C39E 4C F8 C4    JMP HEX_SEND_FILE2
0916 C3A1
0917 C3A1 A5 83      CHK_STATE9  lda REG_C
0918 C3A3 8D 00 80    sta GPIO1
0919 C3A6 A9 01      LDA #1      ; INVALID KEY PRESSED
0920 C3A8 85 92      STA INVALID
0921 C3AA
0922 C3AA
0923 C3AA
0924 C3AA
0925 C3AA
0926 C3AA      ; HEX KEY WAS PRESSED
0927 C3AA
0928 C3AA
0929 C3AA 60      RTS
0930 C3AB
0931 C3AB      ;FFFFFFFFFFFFFFFFFFFF FUNCTION KEY FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0932 C3AB
0933 C3AB      FUNCTION_KEY
0934 C3AB
0935 C3AB C9 19      CMP #19H    ; KEY ADDR
0936 C3AD D0 03      BNE CHK_FUNC1
0937 C3AF 4C 79 C4    JMP KEY_ADDR
0938 C3B2
0939 C3B2 C9 14      CHK_FUNC1  CMP #14H    ; KEY DATA
0940 C3B4 D0 03      BNE CHK_FUNC2
0941 C3B6 4C A9 C4    JMP KEY_DATA
0942 C3B9
0943 C3B9 C9 10      CHK_FUNC2  CMP #10H    ; KEY +
0944 C3BB D0 03      BNE CHK_FUNC3
0945 C3BD 4C 6A C5    JMP KEY_INC
0946 C3C0
0947 C3C0 C9 11      CHK_FUNC3  CMP #11H    ; KEY -
0948 C3C2 D0 03      BNE CHK_FUNC4
0949 C3C4 4C C5 C5    JMP KEY_DEC
0950 C3C7
0951 C3C7 C9 18      CHK_FUNC4  CMP #18H
0952 C3C9 D0 03      BNE CHK_FUNC5
0953 C3CB 4C DE C5    JMP KEY_PC
0954 C3CE
0955 C3CE C9 1B      CHK_FUNC5  CMP #1BH
0956 C3D0 D0 03      BNE CHK_FUNC6
0957 C3D2 4C F2 C5    JMP KEY_REG
0958 C3D5
0959 C3D5 C9 12      CHK_FUNC6  CMP #12H
0960 C3D7 D0 03      BNE CHK_FUNC7
0961 C3D9 4C 43 C6    JMP KEY_GO
0962 C3DC
0963 C3DC C9 1D      CHK_FUNC7  CMP #1DH
0964 C3DE D0 03      BNE CHK_FUNC8
0965 C3E0 4C 65 C4    JMP KEY_REL
0966 C3E3
0967 C3E3 C9 1F      CHK_FUNC8  CMP #1FH
0968 C3E5 D0 03      BNE CHK_FUNC9
0969 C3E7 4C 36 C4    JMP KEY_DOWNLOAD_HEX
0970 C3EA
0971 C3EA C9 13      CHK_FUNC9  CMP #13H
0972 C3EC D0 03      BNE CHK_FUNC10
0973 C3EE 4C 67 C6    JMP KEY_STEP
0974 C3F1
0975 C3F1
0976 C3F1 C9 16      CHK_FUNC10 CMP #16H
0977 C3F3 D0 03      BNE CHK_FUNC11
0978 C3F5 4C 03 C4    JMP KEY_INS
0979 C3F8
0980 C3F8 C9 17      CHK_FUNC11 CMP #17H
0981 C3FA D0 04      BNE CHK_FUNC12
0982 C3FC 4C 02 C4    JMP KEY_DEL
0983 C3FF 60      RTS
0984 C400
0985 C400      CHK_FUNC12
0986 C400
0987 C400 60      RTS
0988 C401
```

```

0989 C401 ;-----
0990 C401 NO_RESPONSE
0991 C401 60 RTS
0992 C402
0993 C402
0994 C402 60 KEY_DEL RTS
0995 C403
0996 C403 ;-----
0997 C403 ; insert byte to current display+1
0998 C403 ; shift down 1kB, 256 bytes.
0999 C403
1000 C403 A5 94 KEY_INS LDA STATE
1001 C405 C9 01 CMP #1
1002 C407 F0 08 BEQ KEY_INS1
1003 C409 C9 02 CMP #2
1004 C40B F0 04 BEQ KEY_INS1
1005 C40D
1006 C40D 20 01 C4 JSR NO_RESPONSE
1007 C410 60 RTS
1008 C411
1009 C411 A5 96 KEY_INS1 LDA DISPLAY
1010 C413 85 86 STA DE
1011 C415 A5 97 LDA DISPLAY+1
1012 C417 85 87 STA DE+1
1013 C419
1014 C419 18 CLC
1015 C41A A5 87 LDA DE+1
1016 C41C 69 28 ADC #40 ; DE=DE+$400
1017 C41E 85 87 STA DE+1
1018 C420
1019 C420
1020 C420
1021 C420
1022 C420 60 RTS
1023 C421
1024 C421
1025 C421
1026 C421
1027 C421
1028 C421
1029 C421
1030 C421 ;-----
1031 C421 KEY_SEND_HEX
1032 C421 A9 07 LDA #7
1033 C423 85 94 STA STATE ; STATE = 7 FOR SENDING HEX FILE
1034 C425
1035 C425 A9 00 LDA #0
1036 C427 85 95 STA ZERO_FLAG
1037 C429 20 81 C4 JSR STILL_ADDRESS
1038 C42C A9 AE LDA #0AEH
1039 C42E 85 8C STA BUFFER
1040 C430 A9 02 LDA #2
1041 C432 85 8D STA BUFFER+1
1042 C434 60 RTS
1043 C435
1044 C435 60 RTS
1045 C436
1046 C436
1047 C436 ;-----
1048 C436 KEY_DOWNLOAD_HEX
1049 C436
1050 C436 A9 B3 LDA #0B3H ; PRINT LOAD
1051 C438 85 91 STA BUFFER+5
1052 C43A A9 85 LDA #85H
1053 C43C 85 91 STA BUFFER+5
1054 C43E A9 A3 LDA #0A3H
1055 C440 85 90 STA BUFFER+4
1056 C442 A9 3F LDA #3FH
1057 C444 85 8F STA BUFFER+3
1058 C446 A9 B3 LDA #0B3H
1059 C448 85 8E STA BUFFER+2
1060 C44A A9 00 LDA #0
1061 C44C 85 8D STA BUFFER+1
1062 C44E 85 8C STA BUFFER
1063 C450
1064 C450 ; JSR NEW_LINE

```

```

1065 C450          ; JSR NEW_LINE
1066 C450          ; JSR NEW_LINE
1067 C450 A9 0A          LDA #10
1068 C452 85 94          STA STATE
1069 C454 60            RTS
1070 C455
1071 C455 A9 55          GO_STATE10  LDA #55H
1072 C457 8D 00 80        STA GPIO1
1073 C45A
1074 C45A
1075 C45A 20 5F C1        JSR GET_RECORD ; GET INTEL HEX FILE
1076 C45D
1077 C45D A9 02          LDA #2
1078 C45F 85 94          STA STATE
1079 C461 20 B1 C4        JSR STILL_DATA
1080 C464
1081 C464 60            RTS
1082 C465
1083 C465
1084 C465
1085 C465 A9 05          ;-----
KEY_REL      LDA #5
1086 C467 85 94          STA STATE ; STATE = 5 FOR RELATIVE BYTE CALCULATION
1087 C469
1088 C469 A9 00          LDA #0
1089 C46B 85 95          STA ZERO_FLAG
1090 C46D 20 81 C4        JSR STILL_ADDRESS
1091 C470 A9 AE          LDA #0AEH
1092 C472 85 8C          STA BUFFER
1093 C474 A9 02          LDA #2
1094 C476 85 8D          STA BUFFER+1
1095 C478 60            RTS
1096 C479
1097 C479
1098 C479
1099 C479 A9 01          ;-----
KEY_ADDR    LDA #1
1100 C47B 85 94          STA STATE ; STATE =1 FOR ADDRESS MODE
1101 C47D
1102 C47D A9 00          LDA #0
1103 C47F 85 95          STA ZERO_FLAG
1104 C481
1105 C481          STILL_ADDRESS
1106 C481 20 D9 C4        JSR READ_MEMORY
1107 C484
1108 C484 A5 91          LDA BUFFER+5
1109 C486 09 40          ORA #40H
1110 C488 85 91          STA BUFFER+5
1111 C48A
1112 C48A A5 90          LDA BUFFER+4
1113 C48C 09 40          ORA #40H
1114 C48E 85 90          STA BUFFER+4
1115 C490
1116 C490 A5 8F          LDA BUFFER+3
1117 C492 09 40          ORA #40H
1118 C494 85 8F          STA BUFFER+3
1119 C496
1120 C496 A5 8E          LDA BUFFER+2
1121 C498 09 40          ORA #40H
1122 C49A 85 8E          STA BUFFER+2
1123 C49C
1124 C49C A5 8D          LDA BUFFER+1
1125 C49E 29 BF          AND #~40H
1126 C4A0 85 8D          STA BUFFER+1
1127 C4A2
1128 C4A2 A5 8C          LDA BUFFER
1129 C4A4 29 BF          AND #~40H
1130 C4A6 85 8C          STA BUFFER
1131 C4A8
1132 C4A8 60            RTS
1133 C4A9
1134 C4A9 A9 02          ;-----
KEY_DATA    LDA #2
1135 C4AB 85 94          STA STATE ; STATE =2 FOR DATA MODE
1136 C4AD
1137 C4AD A9 00          LDA #0
1138 C4AF 85 95          STA ZERO_FLAG
1139 C4B1
1140 C4B1 20 D9 C4        STILL_DATA JSR READ_MEMORY

```

```
1141 C4B4
1142 C4B4 A5 91 LDA BUFFER+5
1143 C4B6 29 BF AND #~40H
1144 C4B8 85 91 STA BUFFER+5
1145 C4BA
1146 C4BA A5 90 LDA BUFFER+4
1147 C4BC 29 BF AND #~40H
1148 C4BE 85 90 STA BUFFER+4
1149 C4C0
1150 C4C0 A5 8F LDA BUFFER+3
1151 C4C2 29 BF AND #~40H
1152 C4C4 85 8F STA BUFFER+3
1153 C4C6
1154 C4C6 A5 8E LDA BUFFER+2
1155 C4C8 29 BF AND #~40H
1156 C4CA 85 8E STA BUFFER+2
1157 C4CC
1158 C4CC A5 8D LDA BUFFER+1
1159 C4CE 09 40 ORA #40H
1160 C4D0 85 8D STA BUFFER+1
1161 C4D2
1162 C4D2 A5 8C LDA BUFFER
1163 C4D4 09 40 ORA #40H
1164 C4D6 85 8C STA BUFFER
1165 C4D8
1166 C4D8 60 RTS
1167 C4D9
1168 C4D9
1169 C4D9 ; READ MEMORY
1170 C4D9
1171 C4D9 READ_MEMORY
1172 C4D9
1173 C4D9 A5 96 LDA DISPLAY
1174 C4DB 85 84 STA HL
1175 C4DD A5 97 LDA DISPLAY+1
1176 C4DF 85 85 STA HL+1
1177 C4E1 20 4B C3 JSR ADDRESS_DISPLAY
1178 C4E4 A0 00 LDY #0
1179 C4E6 B1 84 LDA (HL),Y
1180 C4E8
1181 C4E8 ;STA GPIO1
1182 C4E8
1183 C4E8 20 3D C3 JSR DATA_DISPLAY
1184 C4EB 60 RTS
1185 C4EC ;-----
1186 C4EC
1187 C4EC 20 1C C5 HEX_SEND_FILE JSR HEX_ADDR
1188 C4EF A9 AE LDA #0AEH
1189 C4F1 85 8C STA BUFFER
1190 C4F3 A9 02 LDA #2
1191 C4F5 85 8D STA BUFFER+1
1192 C4F7 60 RTS
1193 C4F8
1194 C4F8 20 1C C5 HEX_SEND_FILE2 JSR HEX_ADDR
1195 C4FB A9 8F LDA #08FH
1196 C4FD 85 8C STA BUFFER
1197 C4FF A9 02 LDA #2
1198 C501 85 8D STA BUFFER+1
1199 C503 60 RTS
1200 C504
1201 C504 ;-----
1202 C504
1203 C504 20 1C C5 HEX_REL JSR HEX_ADDR
1204 C507 A9 AE LDA #0AEH
1205 C509 85 8C STA BUFFER
1206 C50B A9 02 LDA #2
1207 C50D 85 8D STA BUFFER+1
1208 C50F 60 RTS
1209 C510
1210 C510
1211 C510
1212 C510 20 1C C5 HEX_REL6 JSR HEX_ADDR
1213 C513 A9 B3 LDA #0B3H
1214 C515 85 8C STA BUFFER
1215 C517 A9 02 LDA #2
1216 C519 85 8D STA BUFFER+1
```

```

1217 C51B 60          RTS
1218 C51C
1219 C51C
1220 C51C          ;----- HEX KEY FOR ADDRESS -----
1221 C51C
1222 C51C A5 95     HEX_ADDR      LDA ZERO_FLAG
1223 C51E C9 00          CMP #0
1224 C520 D0 0A          BNE SHIFT_ADDRESS
1225 C522
1226 C522 A9 01          LDA #1
1227 C524 85 95          STA ZERO_FLAG
1228 C526 A9 00          LDA #0
1229 C528 85 96          STA DISPLAY
1230 C52A 85 97          STA DISPLAY+1
1231 C52C
1232 C52C 18          SHIFT_ADDRESS CLC
1233 C52D 26 96          ROL DISPLAY
1234 C52F 26 97          ROL DISPLAY+1
1235 C531
1236 C531 18          CLC
1237 C532 26 96          ROL DISPLAY
1238 C534 26 97          ROL DISPLAY+1
1239 C536
1240 C536 18          CLC
1241 C537 26 96          ROL DISPLAY
1242 C539 26 97          ROL DISPLAY+1
1243 C53B
1244 C53B 18          CLC
1245 C53C 26 96          ROL DISPLAY
1246 C53E 26 97          ROL DISPLAY+1
1247 C540
1248 C540 A5 96          LDA DISPLAY
1249 C542 05 83          ORA REG_C
1250 C544 85 96          STA DISPLAY
1251 C546
1252 C546          ; JSR READ_MEMORY
1253 C546
1254 C546 20 81 C4     JSR STILL_ADDRESS
1255 C549
1256 C549 60          RTS
1257 C54A          ;----- HEX KEY FOR DATA MODE -----
1258 C54A
1259 C54A
1260 C54A A5 95     HEX_DATA      LDA ZERO_FLAG
1261 C54C C9 00          CMP #0
1262 C54E D0 0A          BNE SHIFT_DATA
1263 C550
1264 C550 A9 01          LDA #1
1265 C552 85 95          STA ZERO_FLAG
1266 C554
1267 C554 A9 00          LDA #0
1268 C556 A0 00          LDY #0
1269 C558 91 96          STA (DISPLAY),Y
1270 C55A
1271 C55A A0 00     SHIFT_DATA   LDY #0
1272 C55C B1 96          LDA (DISPLAY),Y
1273 C55E 0A          ASL A
1274 C55F 0A          ASL A
1275 C560 0A          ASL A
1276 C561 0A          ASL A
1277 C562 05 83          ORA REG_C
1278 C564 91 96          STA (DISPLAY),Y
1279 C566
1280 C566          ; JSR READ_MEMORY
1281 C566 20 B1 C4     JSR STILL_DATA
1282 C569 60          RTS
1283 C56A
1284 C56A          ; INCREMENT CURRENT ADDRESS BY ONE
1285 C56A          ;
1286 C56A
1287 C56A A5 94     KEY_INC      LDA STATE
1288 C56C C9 05          CMP #5
1289 C56E F0 1D          BEQ REL_KEY_PRESSED
1290 C570
1291 C570 C9 07          CMP #7
1292 C572 F0 35          BEQ SEND_INC1

```



```
1293 C574
1294 C574 A9 02          LDA #2
1295 C576 85 94          STA STATE      ; STATE =2 FOR DATA MODE
1296 C578
1297 C578 A9 00          LDA #0
1298 C57A 85 95          STA ZERO_FLAG
1299 C57C
1300 C57C
1301 C57C 18             CLC
1302 C57D A5 96             LDA DISPLAY
1303 C57F 69 01             ADC #1
1304 C581 85 96             STA DISPLAY
1305 C583 A5 97             LDA DISPLAY+1
1306 C585 69 00             ADC #0
1307 C587 85 97             STA DISPLAY+1
1308 C589                  ; JSR READ_MEMORY
1309 C589 20 B1 C4         JSR STILL_DATA
1310 C58C 60              RTS
1311 C58D
1312 C58D          REL_KEY_PRESSED
1313 C58D
1314 C58D          ; Save start address
1315 C58D
1316 C58D A5 96             LDA DISPLAY
1317 C58F 85 A0             STA START_ADDRESS
1318 C591 A5 97             LDA DISPLAY+1
1319 C593 85 A1             STA START_ADDRESS+1
1320 C595
1321 C595 A9 06             LDA #6
1322 C597 85 94             STA STATE
1323 C599 A9 00             LDA #0
1324 C59B 85 95             STA ZERO_FLAG
1325 C59D
1326 C59D 20 81 C4         JSR STILL_ADDRESS
1327 C5A0 A9 B3             LDA #0B3H
1328 C5A2 85 8C             STA BUFFER
1329 C5A4 A9 02             LDA #2
1330 C5A6 85 8D             STA BUFFER+1
1331 C5A8 60              RTS
1332 C5A9
1333 C5A9
1334 C5A9          SEND_INC1  ; Save start address
1335 C5A9
1336 C5A9 A5 96             LDA DISPLAY
1337 C5AB 85 A0             STA START_ADDRESS
1338 C5AD A5 97             LDA DISPLAY+1
1339 C5AF 85 A1             STA START_ADDRESS+1
1340 C5B1
1341 C5B1 A9 08             LDA #8
1342 C5B3 85 94             STA STATE
1343 C5B5 A9 00             LDA #0
1344 C5B7 85 95             STA ZERO_FLAG
1345 C5B9
1346 C5B9 20 81 C4         JSR STILL_ADDRESS
1347 C5BC A9 8F             LDA #08FH
1348 C5BE 85 8C             STA BUFFER
1349 C5C0 A9 02             LDA #2
1350 C5C2 85 8D             STA BUFFER+1
1351 C5C4 60              RTS
1352 C5C5
1353 C5C5
1354 C5C5
1355 C5C5
1356 C5C5
1357 C5C5
1358 C5C5
1359 C5C5          ; DECREMENT CURRENT ADDRESS BY ONE
1360 C5C5          ;
1361 C5C5
1362 C5C5 A9 02          KEY_DEC  LDA #2
1363 C5C7 85 94          STA STATE      ; STATE =2 FOR DATA MODE
1364 C5C9
1365 C5C9 A9 00          LDA #0
1366 C5CB 85 95          STA ZERO_FLAG
1367 C5CD
1368 C5CD
```

```

1369 C5CD 38 SEC
1370 C5CE A5 96 LDA DISPLAY
1371 C5D0 E9 01 SBC #1
1372 C5D2 85 96 STA DISPLAY
1373 C5D4 A5 97 LDA DISPLAY+1
1374 C5D6 E9 00 SBC #0
1375 C5D8 85 97 STA DISPLAY+1
1376 C5DA ; JSR READ_MEMORY
1377 C5DA 20 B1 C4 JSR STILL_DATA
1378 C5DD 60 RTS
1379 C5DE
1380 C5DE ; KEY PC, SET CURRENT USER ADDRESS
1381 C5DE
1382 C5DE A9 02 KEY_PC LDA #2
1383 C5E0 85 94 STA STATE ; STATE =2 FOR DATA MODE
1384 C5E2
1385 C5E2 A9 00 LDA #0
1386 C5E4 85 95 STA ZERO_FLAG
1387 C5E6
1388 C5E6 A5 98 LDA PC_USER
1389 C5E8 85 96 STA DISPLAY
1390 C5EA A5 99 LDA PC_USER+1
1391 C5EC 85 97 STA DISPLAY+1
1392 C5EE ; JSR READ_MEMORY
1393 C5EE 20 B1 C4 JSR STILL_DATA
1394 C5F1 60 RTS
1395 C5F2
1396 C5F2 ; KEY REGSITER
1397 C5F2 ; SET STATE TO 3 FOR REGISTER INPUT WITH HEX KEY
1398 C5F2
1399 C5F2 A9 03 KEY_REG LDA #3
1400 C5F4 85 94 STA STATE ; STATE = 3 FOR REGISTER DISPLAY
1401 C5F6
1402 C5F6 A9 03 LDA #3
1403 C5F8 85 91 STA BUFFER+5
1404 C5FA A9 8F LDA #8FH
1405 C5FC 85 90 STA BUFFER+4
1406 C5FE A9 BE LDA #0BEH
1407 C600 85 8F STA BUFFER+3
1408 C602 A9 02 LDA #2
1409 C604 85 8E STA BUFFER+2
1410 C606 A9 00 LDA #0
1411 C608 85 8D STA BUFFER+1
1412 C60A 85 8C STA BUFFER
1413 C60C
1414 C60C 60 RTS
1415 C60D
1416 C60D ;-----
1417 C60D GO_STATE8
1418 C60D
1419 C60D A5 96 LDA DISPLAY
1420 C60F 85 A2 STA DESTINATION ; DESTINATION IS NOW ENDING ADDRESS
1421 C611 A5 97 LDA DISPLAY+1
1422 C613 85 A3 STA DESTINATION+1
1423 C615
1424 C615 ; NOW COMPUTE NUMBER OF BYTE = DESTINATION - START_ADDRESS
1425 C615 A5 A0 LDA START_ADDRESS
1426 C617 85 84 STA HL
1427 C619 A5 A1 LDA START_ADDRESS+1
1428 C61B 85 85 STA HL+1
1429 C61D
1430 C61D 38 SEC
1431 C61E A5 A2 LDA DESTINATION
1432 C620 E5 84 SBC HL
1433 C622 85 A4 STA OFFSET_BYTE
1434 C624
1435 C624 A5 A3 LDA DESTINATION+1
1436 C626 E5 85 SBC HL+1
1437 C628 85 A5 STA OFFSET_BYTE+1 ; OFFSET = NUMBER OF BYTE
1438 C62A
1439 C62A ; DEVIDE NUMBER OF BYTE WITH 16 TO GET NUMBER OF RECORD TO BE SENT
1440 C62A
1441 C62A 46 A5 LSR OFFSET_BYTE+1
1442 C62C 66 A4 ROR OFFSET_BYTE
1443 C62E
1444 C62E 46 A5 LSR OFFSET_BYTE+1

```

```

1445 C630 66 A4      ROR OFFSET_BYTE
1446 C632
1447 C632 46 A5      LSR OFFSET_BYTE+1
1448 C634 66 A4      ROR OFFSET_BYTE
1449 C636
1450 C636 46 A5      LSR OFFSET_BYTE+1
1451 C638 66 A4      ROR OFFSET_BYTE
1452 C63A
1453 C63A A5 A4      LDA OFFSET_BYTE ; CHECK RESULT
1454 C63C 8D 00 80    STA GPIO1
1455 C63F
1456 C63F 60          RTS
1457 C640
1458 C640             SHORT_GO_STATE10
1459 C640 4C 55 C4    JMP GO_STATE10
1460 C643
1461 C643             ; KEY GO WRITE USER REGISTERS TO STACK AND USE RTI TO JUMP TO USER PRC
1462 C643             ;
1463 C643
1464 C643 A5 94      KEY_GO LDA STATE
1465 C645 C9 06      CMP #6
1466 C647 F0 46      BEQ GO_STATE6
1467 C649
1468 C649 C9 08      CMP #8
1469 C64B F0 C0      BEQ GO_STATE8
1470 C64D
1471 C64D C9 0A      CMP #10
1472 C64F F0 EF      BEQ SHORT_GO_STATE10
1473 C651
1474 C651 BA          TSX
1475 C652 86 9F      STX SAVE_SP ; SAVE SYSTEM STACK
1476 C654
1477 C654             ; NOW SWITCH TO USER STACK
1478 C654
1479 C654 A6 9D      LDX USER_S
1480 C656 9A          TXS
1481 C657
1482 C657 A5 97      LDA DISPLAY+1
1483 C659 48          PHA
1484 C65A A5 96      LDA DISPLAY
1485 C65C 48          PHA
1486 C65D A5 9E      LDA USER_P
1487 C65F 48          PHA
1488 C660 A6 9B      LDX USER_X
1489 C662 A4 9C      LDY USER_Y
1490 C664 A5 9A      LDA USER_A
1491 C666 40          RTI
1492 C667             ;----- SINGLE STEP -----
1493 C667
1494 C667             KEY_STEP
1495 C667
1496 C667 BA          TSX
1497 C668 86 9F      STX SAVE_SP ; SAVE SYSTEM STACK
1498 C66A
1499 C66A             ; NOW SWITCH TO USER STACK
1500 C66A
1501 C66A A6 9D      LDX USER_S
1502 C66C 9A          TXS
1503 C66D
1504 C66D             ; LOAD CURRENT PC TO DISPLAY
1505 C66D
1506 C66D A5 98      LDA PC_USER
1507 C66F 85 96      STA DISPLAY
1508 C671 A5 99      LDA PC_USER+1
1509 C673 85 97      STA DISPLAY+1
1510 C675
1511 C675
1512 C675
1513 C675 A5 97      LDA DISPLAY+1
1514 C677 48          PHA
1515 C678 A5 96      LDA DISPLAY
1516 C67A 48          PHA
1517 C67B A5 9E      LDA USER_P
1518 C67D 48          PHA
1519 C67E A6 9B      LDX USER_X
1520 C680 A4 9C      LDY USER_Y

```

```

1521 C682
1522 C682 A9 FF      LDA #$FF          ; BREAK MUST BE LOGIC HIGH TO ENABLE IT
1523 C684 8D 02 80   STA PORT1
1524 C687
1525 C687 EA          NOP
1526 C688 EA          NOP
1527 C689 EA          NOP
1528 C68A EA          NOP
1529 C68B EA          NOP
1530 C68C A5 9A      LDA USER_A      ;
1531 C68E 40          RTI              ;
1532 C68F
1533 C68F              ; USER INSTRUCTION IS 8TH FETCHING, IT WILL JUMP TO NMI SERVICE
1534 C68F
1535 C68F
1536 C68F
1537 C68F              ; KEY GO WITH RELATIVE CALCULATION
1538 C68F              ; FIND OFFSET BYTE
1539 C68F
1540 C68F              GO_STATE6
1541 C68F
1542 C68F A5 96          LDA DISPLAY
1543 C691 85 A2          STA DESTINATION
1544 C693 A5 97          LDA DISPLAY+1
1545 C695 85 A3          STA DESTINATION+1
1546 C697
1547 C697              ; NOW COMPUTE OFFSET_BYTE = DESTINATION - START_ADDRESS
1548 C697
1549 C697              ; THE REAL PC WILL BE NEXT INSTRUCTION ADDRESS (+2 FROM BRANCH INSTRU
1550 C697
1551 C697 A5 A0          LDA START_ADDRESS
1552 C699 85 84          STA HL
1553 C69B A5 A1          LDA START_ADDRESS+1
1554 C69D 85 85          STA HL+1
1555 C69F 20 C5 C0      JSR INC_HL
1556 C6A2 20 C5 C0      JSR INC_HL
1557 C6A5
1558 C6A5 38            SEC
1559 C6A6 A5 A2          LDA DESTINATION
1560 C6A8 E5 84          SBC HL
1561 C6AA 85 A4          STA OFFSET_BYTE
1562 C6AC
1563 C6AC A5 A3          LDA DESTINATION+1
1564 C6AE E5 85          SBC HL+1
1565 C6B0 85 A5          STA OFFSET_BYTE+1
1566 C6B2
1567 C6B2              ; CHECK IF THE OFFSET BYTE WAS BETWEEN -128 (FF80) TO +127 (007F)
1568 C6B2              ; IF BIT 7 OF THE OFFSET BYTE IS 0, THE HIGH BYTE MUST BE ZERO
1569 C6B2              ; IF BIT 7 OF THE OFFSET BYTE IS 1, THE HIGH BYTE MUST BE FF
1570 C6B2              ; OTHERWISE, THE OFFSET BYTE WAS OUT OF RANGE, SHOW ERROR THEN
1571 C6B2
1572 C6B2 A5 A4          LDA OFFSET_BYTE
1573 C6B4 29 80          AND #80H
1574 C6B6 F0 09          BEQ CHK_OFFSET_HIGH
1575 C6B8
1576 C6B8              ; CHECK HIGH BYTE MUST BE FF (-1)
1577 C6B8
1578 C6B8 A5 A5          LDA OFFSET_BYTE+1
1579 C6BA C9 FF          CMP #0FFH
1580 C6BC D0 28          BNE OUT_OFF_RANGE
1581 C6BE
1582 C6BE 4C C5 C6      JMP IN_RANGE
1583 C6C1
1584 C6C1              CHK_OFFSET_HIGH
1585 C6C1 A5 A5          LDA OFFSET_BYTE+1
1586 C6C3 D0 21          BNE OUT_OFF_RANGE
1587 C6C5
1588 C6C5              ; STORE OFFSET TO THE 2ND BYTE OF BRANCH INSTRUCTION
1589 C6C5
1590 C6C5 A5 A0          IN_RANGE LDA START_ADDRESS
1591 C6C7 85 84          STA HL
1592 C6C9 A5 A1          LDA START_ADDRESS+1
1593 C6CB 85 85          STA HL+1
1594 C6CD 20 C5 C0      JSR INC_HL
1595 C6D0
1596 C6D0 A5 A4          LDA OFFSET_BYTE

```

```

1597 C6D2 A0 00 LDY #0
1598 C6D4 91 84 STA (HL),Y
1599 C6D6
1600 C6D6 A5 84 LDA HL ; DISPLAY LOCATION OF OFFSET BYTE
1601 C6D8 85 96 STA DISPLAY
1602 C6DA A5 85 LDA HL+1
1603 C6DC 85 97 STA DISPLAY+1
1604 C6DE
1605 C6DE
1606 C6DE 20 B1 C4 JSR STILL_DATA
1607 C6E1
1608 C6E1 A9 02 LDA #2
1609 C6E3 85 94 STA STATE
1610 C6E5 60 RTS
1611 C6E6
1612 C6E6 OUT_OFF_RANGE
1613 C6E6
1614 C6E6 A9 02 LDA #2
1615 C6E8 85 91 STA BUFFER+5
1616 C6EA A9 8F LDA #8FH
1617 C6EC 85 90 STA BUFFER+4
1618 C6EE A9 03 LDA #3
1619 C6F0 85 8F STA BUFFER+3
1620 C6F2 A9 03 LDA #3
1621 C6F4 85 8E STA BUFFER+2
1622 C6F6 A9 00 LDA #0
1623 C6F8 85 8D STA BUFFER+1
1624 C6FA 85 8C STA BUFFER
1625 C6FC
1626 C6FC A9 02 LDA #2
1627 C6FE 85 94 STA STATE
1628 C700
1629 C700 60 RTS
1630 C701
1631 C701
1632 C701
1633 C701
1634 C701
1635 C701
1636 C701
1637 C701 ; NMI SERVICE ROUTINE
1638 C701 ; SAVE CPU REGISTERS TO USER REGISTERS FOR PROGRAM DEBUGGING
1639 C701
1640 C701 NMI_SERVICE
1641 C701
1642 C701 85 9A STA USER_A
1643 C703 ; STA GPIO1 ; 8-BIT DISPLAY WILL SHOW CONTENT OF ACCUMULAT
1644 C703
1645 C703 A9 BF LDA #$BF
1646 C705 8D 02 80 STA PORT1 ; TURN OFF BRK SIGNAL
1647 C708
1648 C708 ; STILL WITH USER STACK
1649 C708
1650 C708 68 PLA
1651 C709 85 9E STA USER_P
1652 C70B
1653 C70B 68 PLA
1654 C70C 85 96 STA DISPLAY
1655 C70E 85 98 STA PC_USER
1656 C710 68 PLA
1657 C711 85 97 STA DISPLAY+1
1658 C713 85 99 STA PC_USER+1
1659 C715 84 9C STY USER_Y
1660 C717 86 9B STX USER_X
1661 C719
1662 C719 BA TSX
1663 C71A 86 9D STX USER_S
1664 C71C
1665 C71C 20 79 C4 JSR KEY_ADDR ; DISPLAY LOCATION THAT BROKED
1666 C71F
1667 C71F ; RESTORE SYSTEM STACK
1668 C71F
1669 C71F A6 9F LDX SAVE_SP
1670 C721 9A TXS
1671 C722
1672 C722 60 RTS

```

```

1673 C723
1674 C723 ; DISPLAY USER REGSITERS
1675 C723
1676 C723 A5 83 HEX_REG LDA REG_C
1677 C725 C9 00 CMP #0
1678 C727 D0 14 BNE CHK_REG1
1679 C729
1680 C729 A5 9A LDA USER_A
1681 C72B ; STA GPIO1
1682 C72B 20 3D C3 JSR DATA_DISPLAY
1683 C72E A9 82 LDA #82H
1684 C730 85 8E STA BUFFER+2
1685 C732 A9 3F LDA #3FH ; REGISTER A
1686 C734 85 8F STA BUFFER+3
1687 C736 A9 00 LDA #0
1688 C738 85 90 STA BUFFER+4
1689 C73A 85 91 STA BUFFER+5
1690 C73C 60 RTS
1691 C73D
1692 C73D CHK_REG1
1693 C73D C9 01 CMP #1
1694 C73F D0 14 BNE CHK_REG2
1695 C741
1696 C741 A5 9B LDA USER_X
1697 C743 ; STA GPIO1
1698 C743
1699 C743 20 3D C3 JSR DATA_DISPLAY
1700 C746 A9 82 LDA #82H
1701 C748 85 8E STA BUFFER+2
1702 C74A A9 07 LDA #7 ; REGISTER X
1703 C74C 85 8F STA BUFFER+3
1704 C74E A9 00 LDA #0
1705 C750 85 90 STA BUFFER+4
1706 C752 85 91 STA BUFFER+5
1707 C754 60 RTS
1708 C755
1709 C755 C9 02 CHK_REG2 CMP #2
1710 C757 D0 14 BNE CHK_REG3
1711 C759
1712 C759 A5 9C LDA USER_Y
1713 C75B ; STA GPIO1
1714 C75B
1715 C75B 20 3D C3 JSR DATA_DISPLAY
1716 C75E A9 82 LDA #82H
1717 C760 85 8E STA BUFFER+2
1718 C762 A9 B6 LDA #0B6H ; REGISTER Y
1719 C764 85 8F STA BUFFER+3
1720 C766 A9 00 LDA #0
1721 C768 85 90 STA BUFFER+4
1722 C76A 85 91 STA BUFFER+5
1723 C76C 60 RTS
1724 C76D
1725 C76D
1726 C76D C9 03 CHK_REG3 CMP #3
1727 C76F D0 14 BNE CHK_REG4
1728 C771
1729 C771 A5 9D LDA USER_S
1730 C773 ; STA GPIO1
1731 C773
1732 C773 20 3D C3 JSR DATA_DISPLAY
1733 C776 A9 82 LDA #82H
1734 C778 85 8E STA BUFFER+2
1735 C77A A9 AE LDA #0AEH ; REGISTER S
1736 C77C 85 8F STA BUFFER+3
1737 C77E A9 00 LDA #0
1738 C780 85 90 STA BUFFER+4
1739 C782 85 91 STA BUFFER+5
1740 C784 60 RTS
1741 C785
1742 C785 C9 05 CHK_REG4 CMP #5
1743 C787 D0 42 BNE CHK_REG5
1744 C789
1745 C789 A9 00 LDA #0 ; RESET HL TO 0000
1746 C78B 85 84 STA HL
1747 C78D 85 85 STA HL+1
1748 C78F

```

```
1749 C78F A5 9E LDA USER_P
1750 C791 ; STA GPIO1
1751 C791 29 01 AND #1
1752 C793 F0 06 BEQ NEXT_BIT1
1753 C795 A5 84 LDA HL
1754 C797 09 01 ORA #1
1755 C799 85 84 STA HL
1756 C79B
1757 C79B A5 9E NEXT_BIT1 LDA USER_P
1758 C79D 29 02 AND #2
1759 C79F F0 06 BEQ NEXT_BIT2
1760 C7A1
1761 C7A1 A5 84 LDA HL
1762 C7A3 09 10 ORA #10H
1763 C7A5 85 84 STA HL
1764 C7A7
1765 C7A7 A5 9E NEXT_BIT2 LDA USER_P
1766 C7A9 29 04 AND #4
1767 C7AB F0 06 BEQ NEXT_BIT3
1768 C7AD
1769 C7AD A5 85 LDA HL+1
1770 C7AF 09 01 ORA #1
1771 C7B1 85 85 STA HL+1
1772 C7B3
1773 C7B3 A5 9E NEXT_BIT3 LDA USER_P
1774 C7B5 29 08 AND #8
1775 C7B7 F0 06 BEQ OK1
1776 C7B9
1777 C7B9 A5 85 LDA HL+1
1778 C7BB 09 10 ORA #10H
1779 C7BD 85 85 STA HL+1
1780 C7BF 20 4B C3 OK1 JSR ADDRESS_DISPLAY
1781 C7C2
1782 C7C2 A9 1F LDA #1FH
1783 C7C4 85 8D STA BUFFER+1
1784 C7C6 A9 85 LDA #085H
1785 C7C8 85 8C STA BUFFER
1786 C7CA 60 RTS
1787 C7CB
1788 C7CB
1789 C7CB C9 04 CHK_REG5 CMP #4
1790 C7CD D0 42 BNE CHK_REG6
1791 C7CF
1792 C7CF A9 00 LDA #0 ; RESET HL TO 0000
1793 C7D1 85 84 STA HL
1794 C7D3 85 85 STA HL+1
1795 C7D5
1796 C7D5 A5 9E LDA USER_P
1797 C7D7 ; STA GPIO1
1798 C7D7
1799 C7D7 29 10 AND #10H
1800 C7D9 F0 06 BEQ NEXT_BIT4
1801 C7DB A5 84 LDA HL
1802 C7DD 09 01 ORA #1
1803 C7DF 85 84 STA HL
1804 C7E1
1805 C7E1 A5 9E NEXT_BIT4 LDA USER_P
1806 C7E3 29 20 AND #20H
1807 C7E5 F0 06 BEQ NEXT_BIT5
1808 C7E7 A5 84 LDA HL
1809 C7E9 09 10 ORA #10H
1810 C7EB 85 84 STA HL
1811 C7ED
1812 C7ED A5 9E NEXT_BIT5 LDA USER_P
1813 C7EF 29 40 AND #40H
1814 C7F1 F0 06 BEQ NEXT_BIT6
1815 C7F3
1816 C7F3 A5 85 LDA HL+1
1817 C7F5 09 01 ORA #1
1818 C7F7 85 85 STA HL+1
1819 C7F9
1820 C7F9 A5 9E NEXT_BIT6 LDA USER_P
1821 C7FB 29 80 AND #80H
1822 C7FD F0 06 BEQ OK2
1823 C7FF
1824 C7FF A5 85 LDA HL+1
```

```

1825 C801 09 10          ORA #10H
1826 C803 85 85          STA HL+1
1827 C805
1828 C805 20 4B C3      OK2          JSR ADDRESS_DISPLAY
1829 C808
1830 C808 A9 1F          LDA #1FH
1831 C80A 85 8D          STA BUFFER+1
1832 C80C A9 37          LDA #37H
1833 C80E 85 8C          STA BUFFER
1834 C810 60              RTS
1835 C811
1836 C811 C9 10          CHK_REG6  CMP #10H
1837 C813 B0 1C          BCS NOT_HEX
1838 C815
1839 C815                ; NOW DISPLAY PAGE ZERO BYTE FROM 0 TO 9
1840 C815
1841 C815 38              SEC
1842 C816 E9 06          SBC #6
1843 C818
1844 C818                ; NOW A IS LOCATION IS PAGE ZERO 0-9
1845 C818 AA              TAX
1846 C819 B5 00          LDA 0,X
1847 C81B 86 A8          STX SAVE_X
1848 C81D
1849 C81D 20 3D C3      JSR DATA_DISPLAY
1850 C820
1851 C820 A6 A8          LDX SAVE_X
1852 C822
1853 C822 8A              TXA
1854 C823 85 85          STA HL+1
1855 C825 20 4B C3      JSR ADDRESS_DISPLAY
1856 C828
1857 C828 A9 82          LDA #82H
1858 C82A 85 8F          STA BUFFER+3
1859 C82C A9 00          LDA #0
1860 C82E 85 8E          STA BUFFER+2
1861 C830 60              RTS
1862 C831
1863 C831                NOT_HEX
1864 C831 60              RTS
1865 C832
1866 C832                ; PRODUCE BEEP WHEN KEY PRESSED
1867 C832                ; CALIBRATED TO 523Hz
1868 C832
1869 C832 AD 01 80      BEEP      LDA PORT0
1870 C835 29 40          AND #40H
1871 C837 F0 15          BEQ NO_BEEP      ; CHECK IF REPEAT KEY IS PRESSED, THEN NO BEEP
1872 C839
1873 C839 A2 40          LDX #40H
1874 C83B A9 3F          BEEP2     LDA #3FH
1875 C83D 8D 02 80      STA PORT1
1876 C840 20 4F C8      JSR BEEP_DELAY
1877 C843 A9 BF          LDA #0BFH
1878 C845 8D 02 80      STA PORT1
1879 C848 20 4F C8      JSR BEEP_DELAY
1880 C84B
1881 C84B CA              DEX
1882 C84C D0 ED          BNE BEEP2
1883 C84E
1884 C84E 60              NO_BEEP  RTS
1885 C84F
1886 C84F A0 BB          BEEP_DELAY LDY #0BBH      ;
1887 C851 88              BEEP_LOOP DEY
1888 C852 D0 FD          BNE BEEP_LOOP
1889 C854 60              RTS
1890 C855
1891 C855                ; DISPLAY COLD BOOT MESSAGE
1892 C855                ;
1893 C855
1894 C855                COLD_MESSAGE
1895 C855
1896 C855 A9 0A          LDA #10
1897 C857 85 81          STA REG_D
1898 C859
1899 C859 A9 08          LDA #8
1900 C85B 85 82          STA REG_B

```



```

1901 C85D
1902 C85D A2 07          LDX #7
1903 C85F
1904 C85F          DISPLAY2
1905 C85F 20 66 C2      JSR SCAN2
1906 C862
1907 C862 C6 81          DEC REG_D
1908 C864 D0 F9          BNE DISPLAY2
1909 C866
1910 C866 CA            DEX
1911 C867
1912 C867 C6 82          DEC REG_B
1913 C869 D0 F4          BNE DISPLAY2
1914 C86B 60            RTS
1915 C86C
1916 C86C
1917 C86C
1918 C86C
1919 C86C          ; NMI and IRQ are called via RAM-vector. This enables the programmer
1920 C86C          ; to insert his own routines.
1921 C86C
1922 C86C
1923 C86C 6C FA 00      NMI    JMP    ($FA)
1924 C86F 6C FE 00      IRQ    JMP    ($FE)
1925 C872
1926 C872
1927 C872          ;-----
1928 C872
1929 C872 A9 00          MAIN    LDA #0
1930 C874 85 8C          STA BUFFER
1931 C876 85 8D          STA BUFFER+1
1932 C878 85 92          STA INVALID ; CLEAR INVALID FLAG
1933 C87A
1934 C87A          ; INSERT 6502 TEXT
1935 C87A
1936 C87A A9 AF          LDA #0AFH
1937 C87C 85 91          STA BUFFER+5
1938 C87E A9 AE          LDA #0AEH
1939 C880 85 90          STA BUFFER+4
1940 C882 A9 BD          LDA #0BDH
1941 C884 85 8F          STA BUFFER+3
1942 C886 A9 9B          LDA #9BH
1943 C888 85 8E          STA BUFFER+2
1944 C88A
1945 C88A
1946 C88A
1947 C88A          ; STORE VECTOR INTERRUPT
1948 C88A
1949 C88A A9 01          LDA #NMI_SERVICE&0FFH ; NMI MUST BE SET BEFORE USING SINGLE STEP
1950 C88C 85 FA          STA $FA
1951 C88E 85 FE          STA $FE
1952 C890
1953 C890 A9 C7          LDA #(NMI_SERVICE>>8)
1954 C892 85 FB          STA $FB
1955 C894 85 FF          STA $FF
1956 C896
1957 C896 A2 FF          LDX #$FF
1958 C898 9A          TXS          ; SET SYSTEM STACK TO 1FFH
1959 C899 A9 7F          LDA #$7F    ; AND USER STACK TO 17FH
1960 C89B 85 9D          STA USER_S
1961 C89D
1962 C89D D8          CLD
1963 C89E 78          SEI          ; DISABLE IRQ
1964 C89F
1965 C89F A9 00          LDA #0
1966 C8A1 85 94          STA STATE  ; INITIAL STATE
1967 C8A3 85 95          STA ZERO_FLAG
1968 C8A5
1969 C8A5 A9 00          LDA #0
1970 C8A7 85 96          STA DISPLAY
1971 C8A9 85 98          STA PC_USER
1972 C8AB A9 02          LDA #02H
1973 C8AD 85 97          STA DISPLAY+1
1974 C8AF 85 99          STA PC_USER+1
1975 C8B1
1976 C8B1

```

```

1977 C8B1 A5 96          LDA DISPLAY
1978 C8B3 85 84          STA HL
1979 C8B5 A5 97          LDA DISPLAY+1
1980 C8B7 85 85          STA HL+1
1981 C8B9
1982 C8B9
1983 C8B9
1984 C8B9          ;JSR ADDRESS_DISPLAY
1985 C8B9 A0 00          LDY #0
1986 C8BB B1 84          LDA (HL),Y
1987 C8BD          ;JSR DATA_DISPLAY
1988 C8BD
1989 C8BD A5 A6          LDA COLD
1990 C8BF C9 99          CMP #99H
1991 C8C1 F0 0F          BEQ WARM_BOOT
1992 C8C3
1993 C8C3 A9 99          LDA #99H
1994 C8C5 85 A6          STA COLD
1995 C8C7
1996 C8C7
1997 C8C7 A9 FF          LDA #$FF
1998 C8C9 8D 00 80       STA GPIO1      ; TEST GPIO1
1999 C8CC
2000 C8CC 20 55 C8       JSR COLD_MESSAGE
2001 C8CF 20 32 C8       JSR BEEP
2002 C8D2
2003 C8D2          WARM_BOOT
2004 C8D2 A9 00          LDA #0
2005 C8D4 8D 00 80       STA GPIO1
2006 C8D7
2007 C8D7 20 ED C2       LOOP3JSR SCANKEY
2008 C8DA 20 66 C3       JSR KEYEXE
2009 C8DD 20 32 C8       JSR BEEP
2010 C8E0 4C D7 C8       JMP LOOP3
2011 C8E3
2012 C8E3
2013 C8E3
2014 C8E3
2015 C8E3
2016 C8E3
2017 C8E3
2018 C8E3          ;-----
2019 C8E3 00          START_MSG .BYTE 0
2020 C8E4 00          .BYTE 0
2021 C8E5 9B          .BYTE 9BH
2022 C8E6 BD          .BYTE 0BDH
2023 C8E7 AE          .BYTE 0AEH
2024 C8E8 AF          .BYTE 0AFH
2025 C8E9 00          .BYTE 0
2026 C8EA 00          .BYTE 0
2027 C8EB 00          .BYTE 0
2028 C8EC 00          .BYTE 0
2029 C8ED 00          .BYTE 0
2030 C8EE 00          .BYTE 0
2031 C8EF
2032 C8EF
2033 C8EF BD          SEGTAB .BYTE 0BDH      ;'0'
2034 C8F0 30          .BYTE 030H      ;'1'
2035 C8F1 9B          .BYTE 09BH      ;'2'
2036 C8F2 BA          .BYTE 0BAH      ;'3'
2037 C8F3 36          .BYTE 036H      ;'4'
2038 C8F4 AE          .BYTE 0AEH      ;'5'
2039 C8F5 AF          .BYTE 0AFH      ;'6'
2040 C8F6 38          .BYTE 038H      ;'7'
2041 C8F7 BF          .BYTE 0BFH      ;'8'
2042 C8F8 BE          .BYTE 0BEH      ;'9'
2043 C8F9 3F          .BYTE 03FH      ;'A'
2044 C8FA A7          .BYTE 0A7H      ;'B'
2045 C8FB 8D          .BYTE 08DH      ;'C'
2046 C8FC B3          .BYTE 0B3H      ;'D'
2047 C8FD 8F          .BYTE 08FH      ;'E'
2048 C8FE 0F          .BYTE 00FH      ;'F'
2049 C8FF
2050 C8FF
2051 C8FF
2052 C8FF          ; Key-position-code to key-internal-code conversion table.

```

```

2053 C8FF
2054 C8FF KEYTAB:
2055 C8FF 03 K0 .BYTE 03H ;HEX_3
2056 C900 07 K1 .BYTE 07H ;HEX_7
2057 C901 0B K2 .BYTE 0BH ;HEX_B
2058 C902 0F K3 .BYTE 0FH ;HEX_F
2059 C903 20 K4 .BYTE 20H ;NOT USED
2060 C904 21 K5 .BYTE 21H ;NOT USED
2061 C905 02 K6 .BYTE 02H ;HEX_2
2062 C906 06 K7 .BYTE 06H ;HEX_6
2063 C907 0A K8 .BYTE 0AH ;HEX_A
2064 C908 0E K9 .BYTE 0EH ;HEX_E
2065 C909 22 K0A .BYTE 22H ;NOT USED
2066 C90A 23 K0B .BYTE 23H ;NOT USED
2067 C90B 01 K0C .BYTE 01H ;HEX_1
2068 C90C 05 K0D .BYTE 05H ;HEX_5
2069 C90D 09 K0E .BYTE 09H ;HEX_9
2070 C90E 0D K0F .BYTE 0DH ;HEX_D
2071 C90F 13 K10 .BYTE 13H ;STEP
2072 C910 1F K11 .BYTE 1FH ;TAPERD
2073 C911 00 K12 .BYTE 00H ;HEX_0
2074 C912 04 K13 .BYTE 04H ;HEX_4
2075 C913 08 K14 .BYTE 08H ;HEX_8
2076 C914 0C K15 .BYTE 0CH ;HEX_C
2077 C915 12 K16 .BYTE 12H ;GO
2078 C916 1E K17 .BYTE 1EH ;TAPEWR
2079 C917 1A K18 .BYTE 1AH ;CBR
2080 C918 18 K19 .BYTE 18H ;PC
2081 C919 1B K1A .BYTE 1BH ;REG
2082 C91A 19 K1B .BYTE 19H ;ADDR
2083 C91B 17 K1C .BYTE 17H ;DEL
2084 C91C 1D K1D .BYTE 1DH ;RELA
2085 C91D 15 K1E .BYTE 15H ;SBR
2086 C91E 11 K1F .BYTE 11H ;-
2087 C91F 14 K20 .BYTE 14H ;DATA
2088 C920 10 K21 .BYTE 10H ;+
2089 C921 16 K22 .BYTE 16H ;INS
2090 C922 1C K23 .BYTE 1CH ;MOVE
2091 C923
2092 C923 ; PAGE FOR CONSTANT STRINGS AREA
2093 C923
2094 EF00 .ORG 0EF00H ; ROM MONITOR
2095 EF00 ; .ORG 06F00H ; RAM TEST
2096 EF00
2097 EF00
2098 EF00 ;TEXT1 .BYTE "6502 TRAINER KIT V1.0 ROM", 10, 13, 0
2099 EF00 363530322054TEXT1 .BYTE "6502 TRAINER KIT V1.0 RAM", 10, 13, 0
2099 EF06 5241494E4552204B49542056312E302052414D0A0D00
2100 EF1C
2101 EF1C
2102 EF1C 3E 3E 00 PROMPT .BYTE ">>", 0
2103 EF1F
2104 EF1F
2105 EF1F
2106 EF1F
2107 EF1F ; VECTOR NMI,RESET AND IRQ
2108 EF1F
2109 EF1F
2110 FFFA .ORG 0FFFAH
2111 FFFA
2112 FFFA 6C C8 .WORD NMI
2113 FFFC 00 C0 .WORD 0C000H ; RESET VECTOR
2114 FFFE 6F C8 .WORD IRQ ; IRQ VECTOR
2115 0000
2116 0000
2117 0000
2118 0000
2119 0000 .END
2120 0000
2121 0000
2122 0000
tasm: Number of errors = 0

```

## NOTE