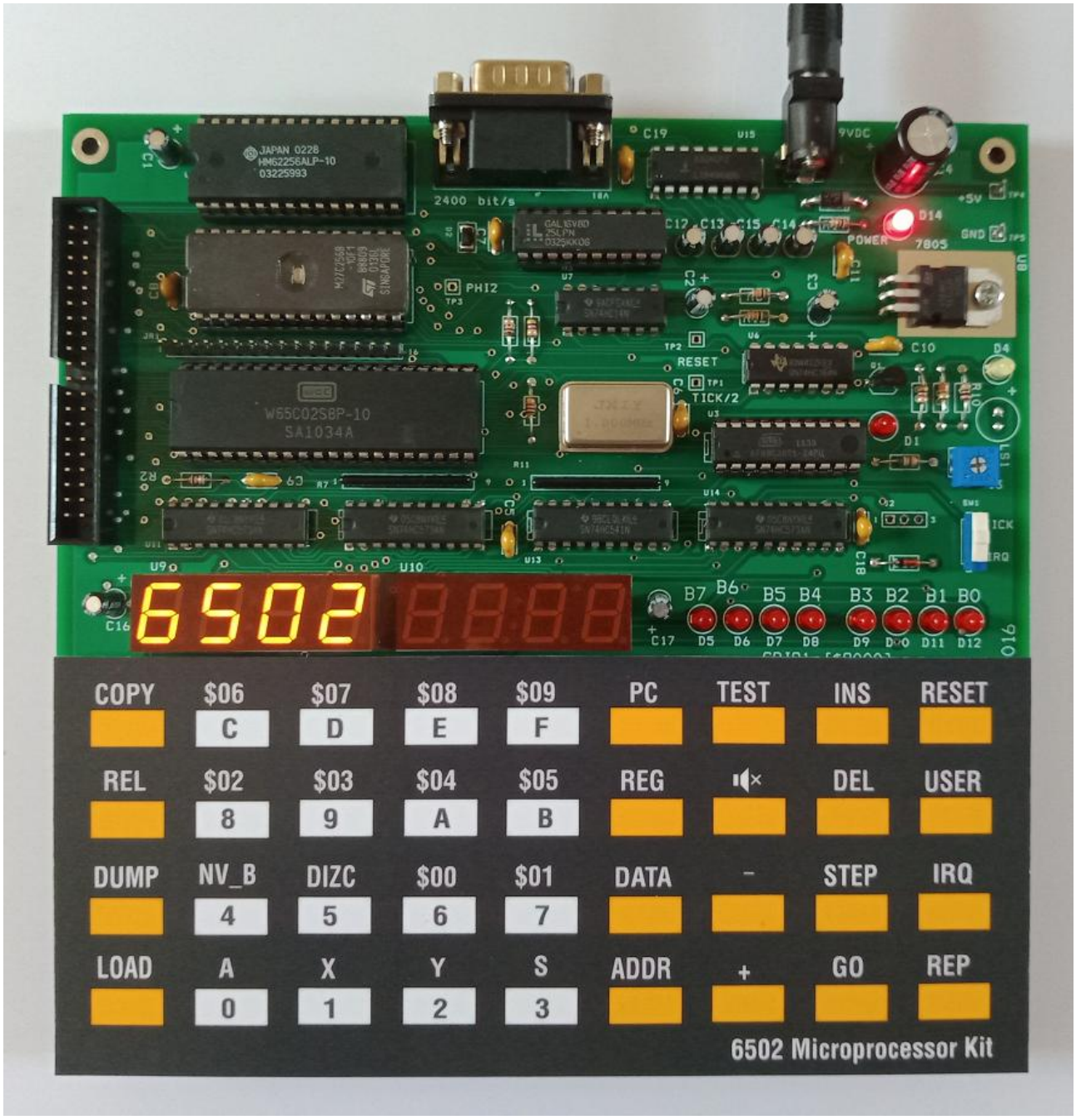
6502 Microprocessor Kit
User's Manual



Rev.2.0 June, 2021

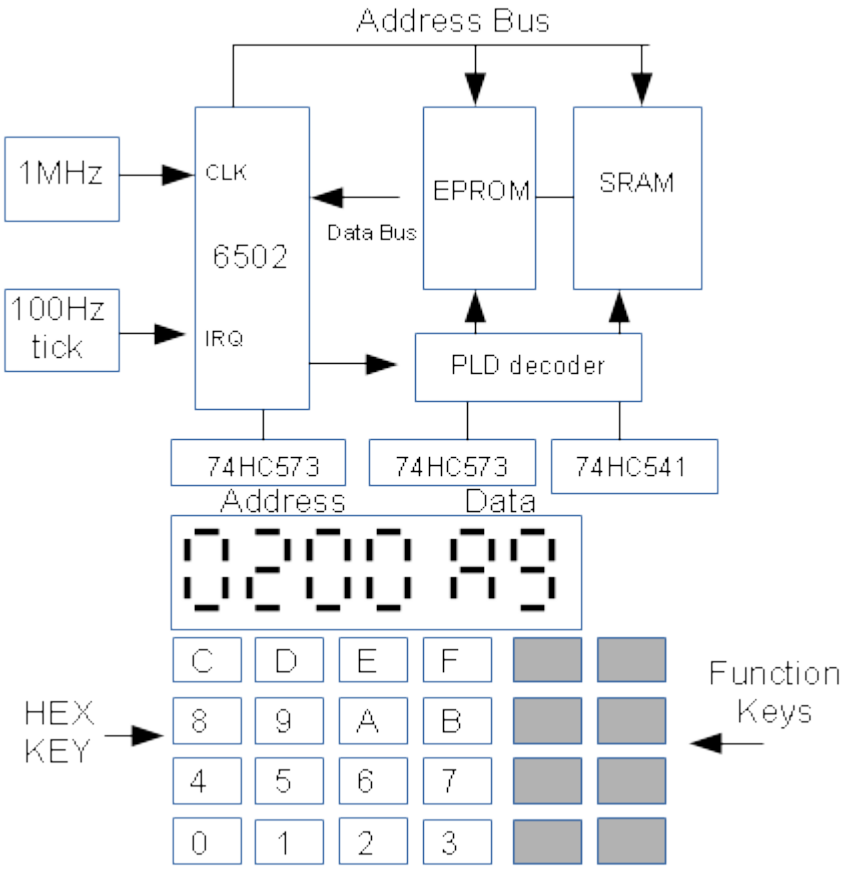
6502 MICROPROCESSOR KIT

CONTENTS

OVERVIEW

The Kit is a single board computer based on the 8-bit 6502 microprocessor. Kit provides 32kB RAM for testing the 6502 instruction directly. The monitor program provides functions for memory examining, user registers for saving CPU status, insert/delete byte, and single step running. The 6-digit display shows the memory address and 8-bit content. Students will learn basic of the 6502 operation 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.

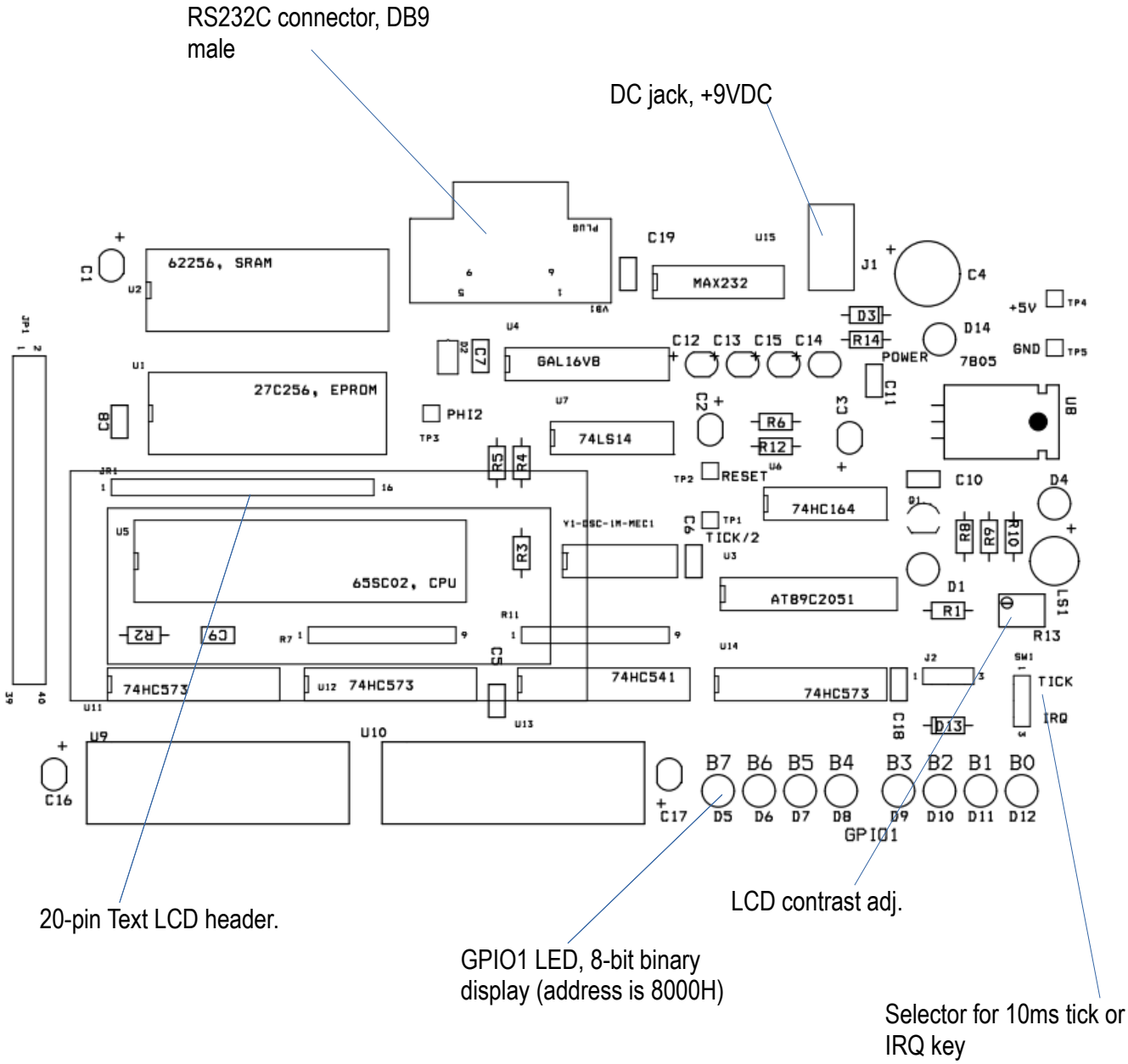
2. The kit provides 8-bit LCD module using CPU bus interfacing.

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



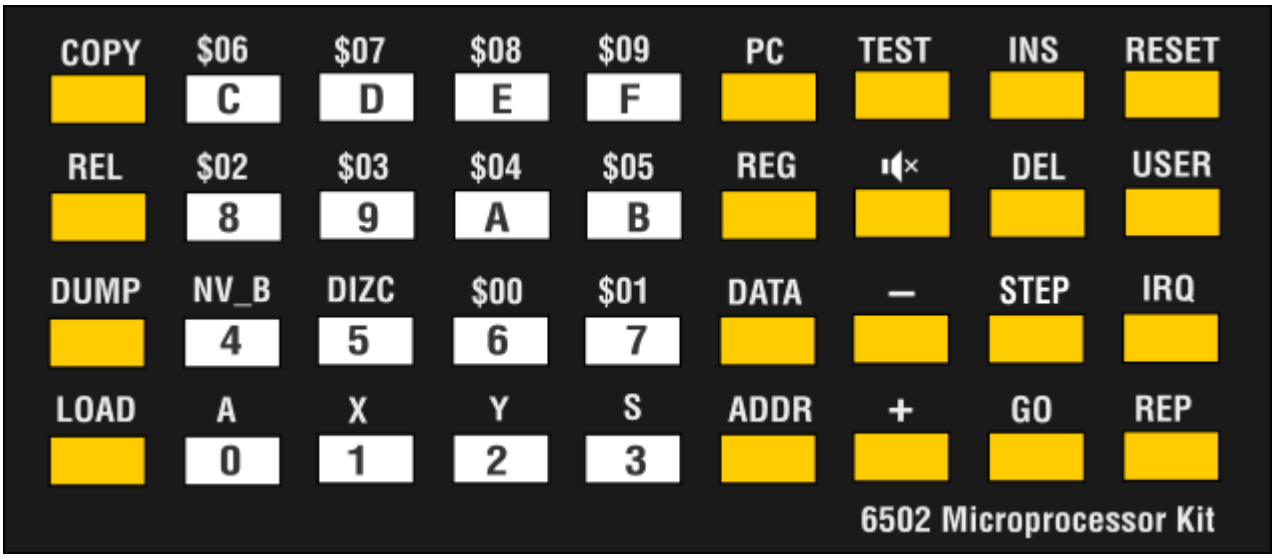
Important Notes

1. Insert/Remove 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.

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 when pressed.

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

Monitor function keys

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

INS Insert one byte at display address+1, shift 256 bytes down..

DEL Delete one byte at current display, shift 256 bytes up.

STEP Execute user code only single instruction and return to save CPU registers to user registers. We can check the results in a given user registers with key REG.

GO Jump from monitor program to user code

TEST Test 10ms interrupt with SW1 set to 10ms tick

 BEEP on/off

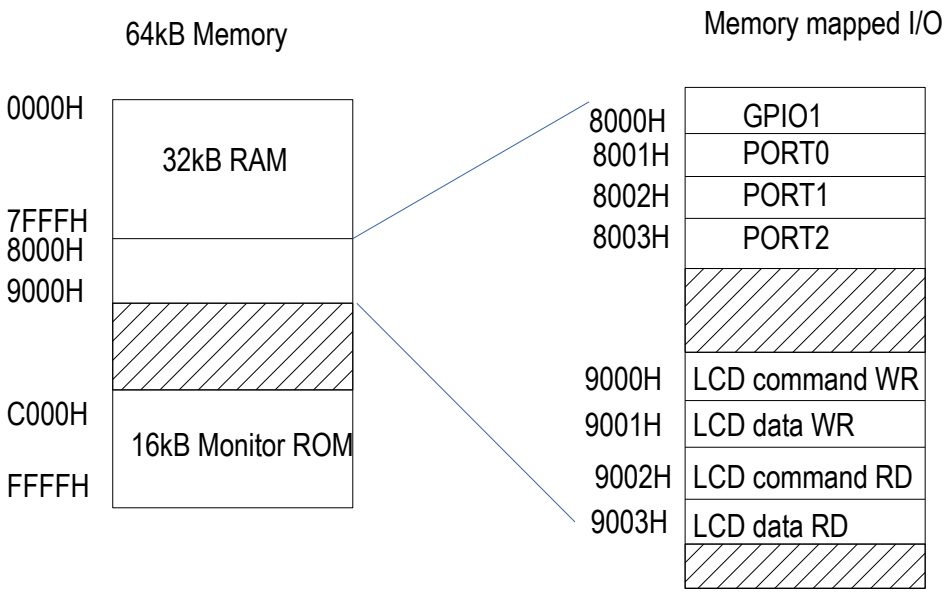
-	Decrement address by one
+	Increment address by one
PC	Set address with user Program Counter
REG	Display user registers, flags or page zero $00 to $09, used with HEX key.
DATA	Set entry mode for Data field
ADDR	Set entry mode for Address field
COPY	Copy block of memory, enter START address, + enter END address, + enter Destination address, then key GO to make a copy.
REL	Compute relative byte, used with key + for Start, Destination and key GO
DUMP	Dump memory on 2400 Terminal
LOAD	Load Intel or MOS hex file sent from 2400 Terminal

HARDWARE FEATURES

- CPU: 6502, 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

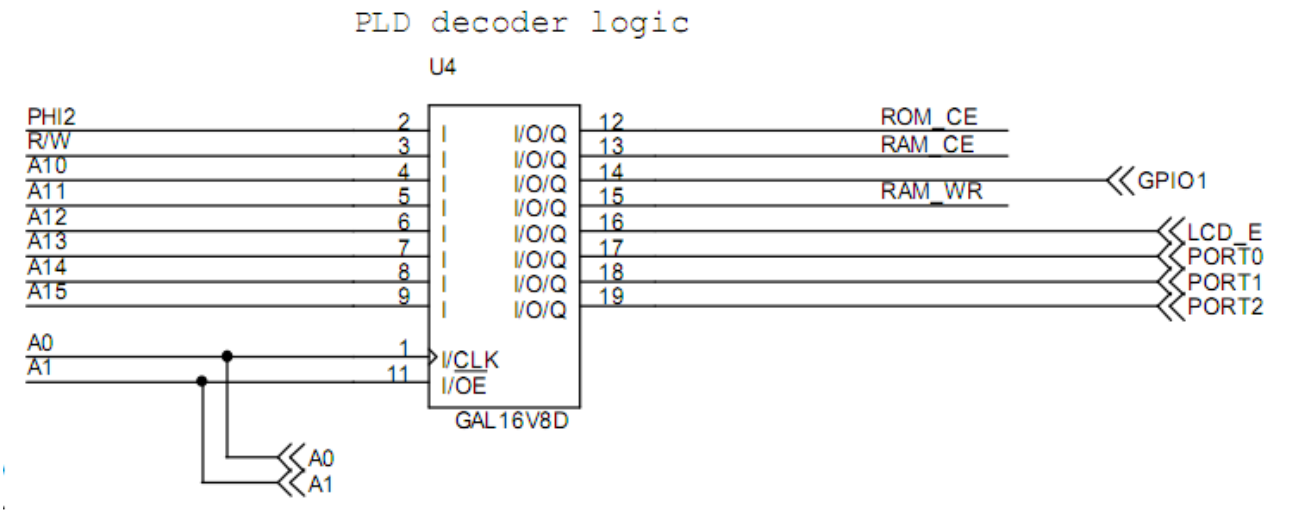
MEMORY AND I/O MAPS

The first 32kB is RAM space from 0000-7FFFH. Zero page is location 00-FFH. Stack space is 100-1FFH. User space is from 0200H to 7FFFH. The 6502 CPU uses memory space from 8000H-BFFFH for I/O port. The monitor ROM is located at C000H-FFFFH



PLD DECODER

The address decoder for memory and I/O devices is made with the programmable logic device, GAL16V8D. Input signals are A0,A1, A10-A15, R/W and clock Phase 2. The output signals are produced from the logic combination of these input signals using preprogrammed logic equation.



PLD logic equations listing.

```
/* *************** INPUT PINS *********************/
PIN 2     =      PHI2;
PIN 3     =      RW;
PIN 4     =      A6;
PIN 5     =      A7;
PIN 6     =      A12;
PIN 7     =      A13;
PIN 8     =      A14;
PIN 9     =      A15;
PIN 1     =      A0;
PIN 11     =     A1;

FIELD ADDRESS = [A15..A12];

/* *************** OUTPUT PINS *********************/
PIN   12    =    ROMCE;
PIN   13    =    RAMCE;
PIN   14    =    GPIO1;
PIN   15    =    RAMWR;
PIN   16    =    LCD_E;
PIN   17    =    PORT0;
PIN   18    =    PORT1;
```

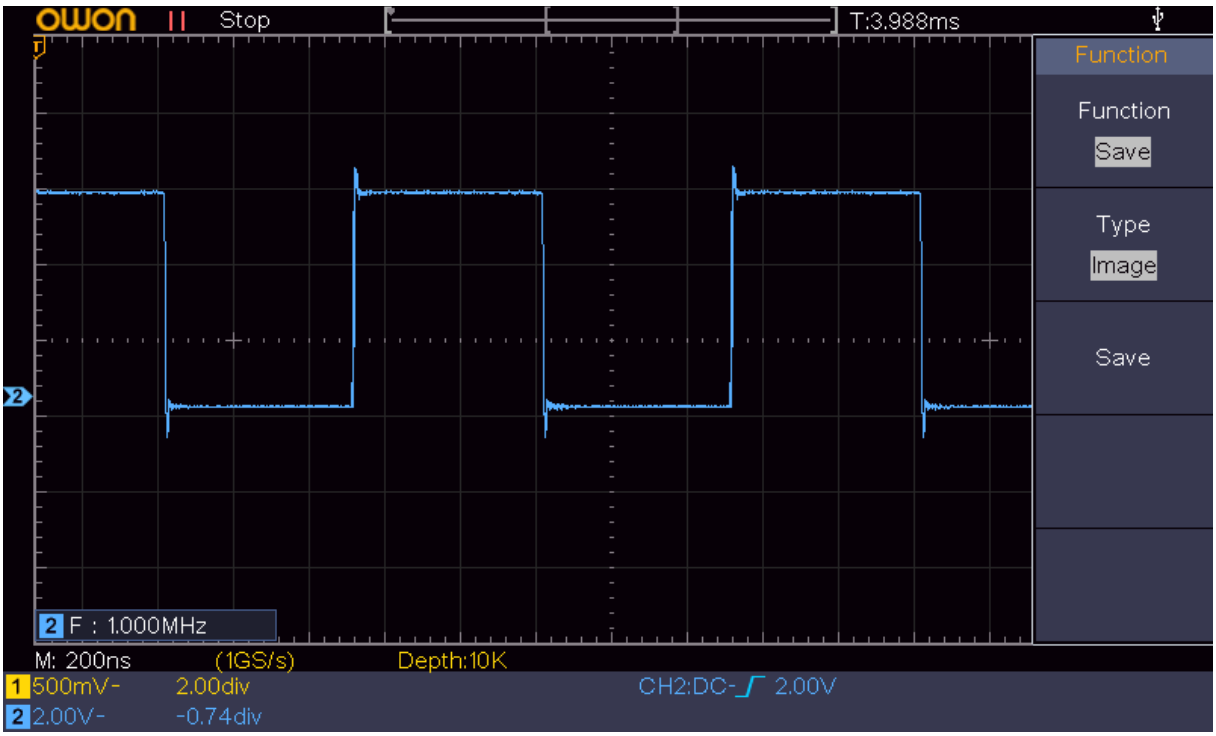
```
PIN  19   =   PORT2;


!ROMCE = RW & PHI2 & ADDRESS:[C000..FFFF];

!RAMCE = ADDRESS:[0000..7FFF];

!RAMWR = !RW & PHI2 & ADDRESS:[0000..7FFF];

!LCD_E = !PHI2 # !A15 # A14 # A13 # !A12;

!GPIO1 = RW # !PHI2 # !A15 # A14 # A13 # A12 # A1 # A0;

PORT0 =  !RW # !PHI2 # !A15 # A14 # A13 # A12 # A1 # !A0;

!PORT1 = RW # !PHI2 # !A15 # A14 # A13 # A12 # !A1 # A0;

!PORT2 = RW # !PHI2 # !A15 # A14 # A13 # A12 # !A1 # !A0;
```

Note

1. Logic symbols are # logical OR, & logical AND, ! Logical NOT
2. WinCupl compiler version 5.30.4 is used to produce JEDEC file.

CPU OSCILLATOR

The 6502 CPU is supplied with 1MHz clock signal from the 14-pin oscillator module. The frequency of clock signal measured at pin 37, CLK0 is 1.000MHz.


OWON
Stop
T:3.988ms
Function
Function
Save
Type
Image
Save
2 F : 1.000MHz
M: 200ns (1GS/s) Depth:10K
1 500mV- 2.00div
2 2.00V- -0.74div
CH2:DC- 2.00V

GETTING STARTED

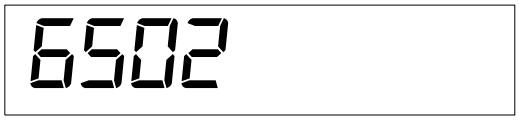
Kit accepts DC power supply with minimum voltage of +7.5V. It draws DC current approx. 70mA. However we can use +9VDC from any AC adapter. The example of AC adapter is shown below.

The center pin is positive. The outer is GND.

If your adapter is adjustable output voltage, try with approx. +9V. Higher voltage will make higher power loss at the voltage regulator, 7805. Dropping voltage across 7805 is approx. +2V. To get +5VDC for the kit, we thus need DC input >+7.5V.

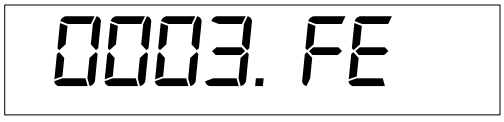
Another power source is the Li-ion power bank used for cell phone. We can use DC-to-DC converter cable to convert +5V to +9V. This power source is suitable for lab work by young student and for remote area. The cable can connect to the power bank and provide +9VDC.

When power up, we will see the cold boot message 6502 on the seven segment display.



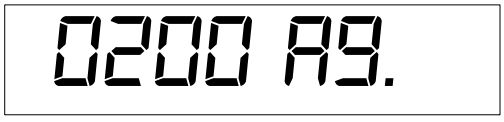
Press PC, the display will show HOME location at 0200. The data field will show its content.

Press ADDR, sets entry mode to ADDRESS field, the dot indicates Address Mode.



Any hex key entry will set the new location.

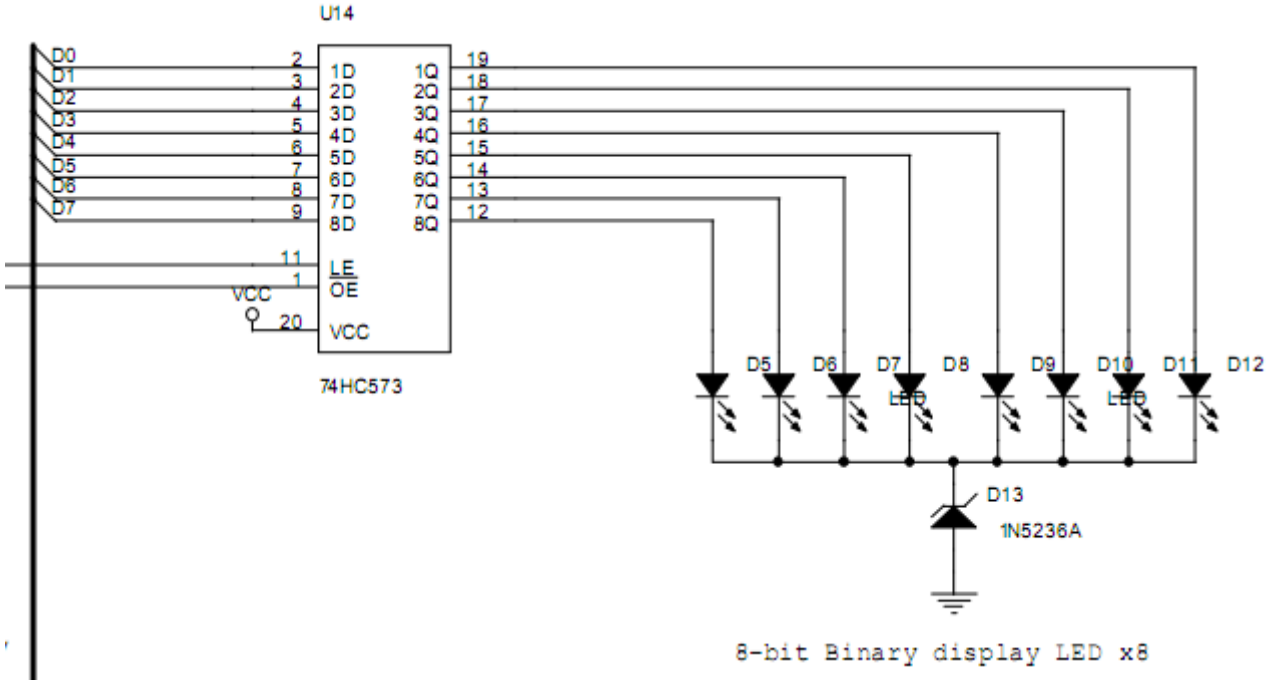
Press + for next location. Press DATA will set entry mode to data field.



Any hex key entry will write to memory directly.

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.



This output port location is decoded in memory space at 8000H. We can use STA $8000 to write the accumulator content to the LED easily.

TEST THE CODE WITH SINGLE STEP

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 2nd 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 increment, 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?

You can press STEP with REP key to make auto repeating.

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

USING BREAK POINT

We can test the operation of 6502 instructions with BREAK POINT setting easily. The break point returns to monitor program and saves the CPU registers to user registers. We can examine them with key REG.

Address	Hex code	Label	Instruction	comment
0200	A9 01	MAIN	LDA #1	Load A with 1
0202	49 FF		EOR #$FF	Exclusive OR A with FF
0204	00		BRK	Break, return monitor

The test code will make Exclusive Or accumulator content 0000 0001 with 1111 1111.

We put the BRK instruction, 00 at location 204.

What is the result by hand computing? Keep it.

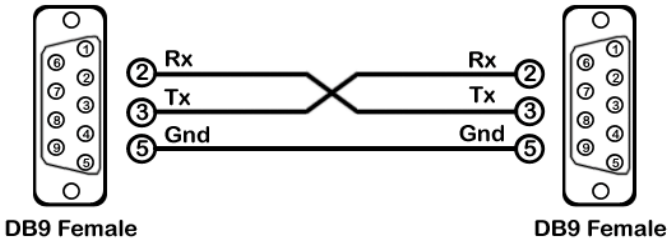
Now after enter the code, A9, 01, 49, FF, and 00. Press PC then GO.

Press REG, 0, what is the result in A?

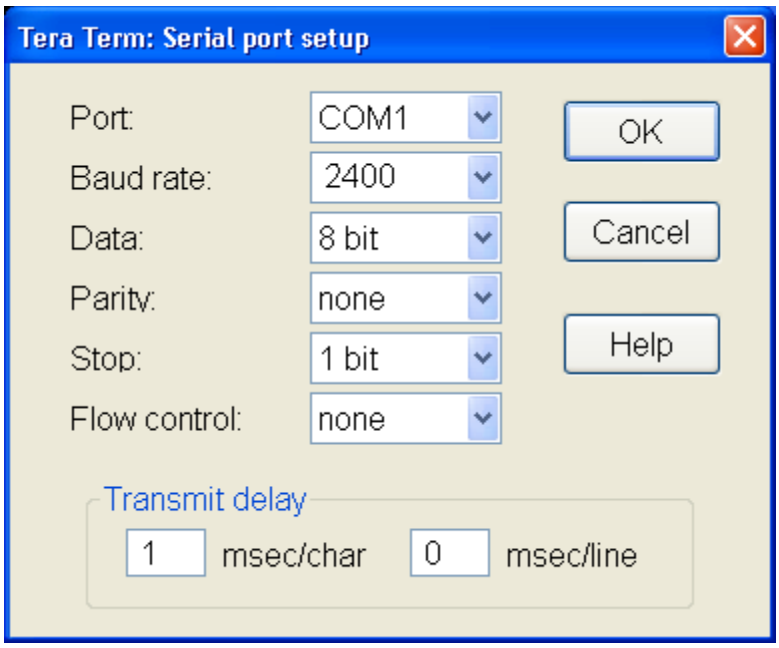
CONNECTING TERMINAL

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://ttssh2.sourceforge.jp/index.html.en

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



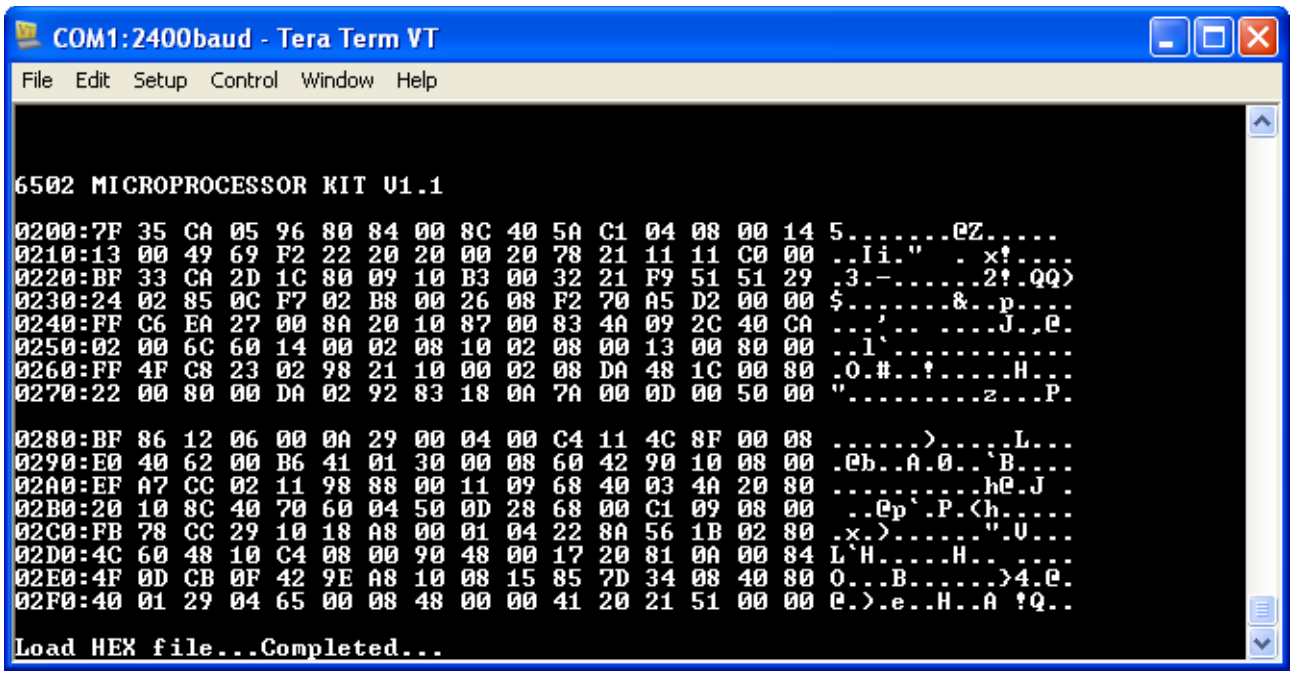
To download Intel or MOS hex file that generated from the assembler or c compiler, **set serial port speed to 2400 bit/s, 8-data bit, no parity, no flow control, one stop bit.**



Press key LOAD, the kit will wait for the data stream from terminal.

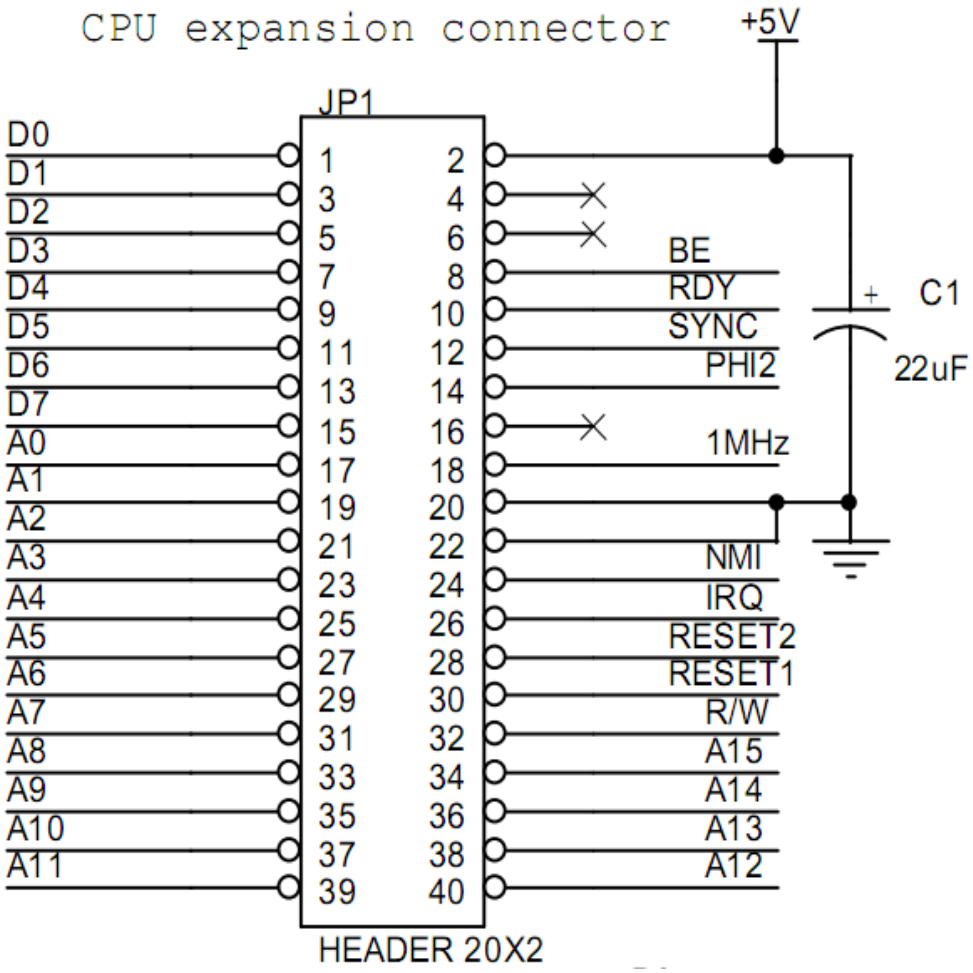
On PC, Click file>Send File>LED.HEX. The kit will read the hex file, write to memory.

Kit accepts both Intel or MOS hex files.


COM1:2400baud - Tera Term VT
File Edit Setup Control Window Help
6502 MICROPROCESSOR KIT V1.1
0200:7F 35 CA 05 96 80 84 00 8C 40 5A C1 04 08 00 14 5.......@Z.....
0210:13 00 49 69 F2 22 20 20 00 20 78 21 11 11 C0 00 ..Ii." . x!....
0220:BF 33 CA 2D 1C 80 09 10 B3 00 32 21 F9 51 51 29 .3.-......2!.QQ)
0230:24 02 85 0C F7 02 B8 00 26 08 F2 70 A5 D2 00 00 $.......&..p....
0240:FF C6 EA 27 00 8A 20 10 87 00 83 4A 09 2C 40 CA ...'.. ....J.,@.
0250:02 00 6C 60 14 00 02 08 10 02 08 00 13 00 80 00 ..l`............
0260:FF 4F C8 23 02 98 21 10 00 02 08 DA 48 1C 00 80 .O.#..!.....H...
0270:22 00 80 00 DA 02 92 83 18 0A 7A 00 0D 00 50 00 ".........z...P.
0280:BF 86 12 06 00 0A 29 00 04 00 C4 11 4C 8F 00 08 ......).....L...
0290:E0 40 62 00 B6 41 01 30 00 08 60 42 90 10 08 00 .@b..A.0..`B....
02A0:EF A7 CC 02 11 98 88 00 11 09 68 40 03 4A 20 80 ....h@.J .
02B0:20 10 8C 40 70 60 04 50 0D 28 68 00 C1 09 08 00 ..@p`.P.(h.....
02C0:FB 78 CC 29 10 18 A8 00 01 04 22 8A 56 1B 02 80 .x.)......".V...
02D0:4C 60 48 10 C4 08 00 90 48 00 17 20 81 0A 00 84 L`H.....H.. ....
02E0:4F 0D CB 0F 42 9E A8 10 08 15 85 7D 34 08 40 80 O...B......}4.@.
02F0:40 01 29 04 65 00 08 48 00 00 41 20 21 51 00 00 @.).e..H..A !Q..
Load HEX file...Completed...

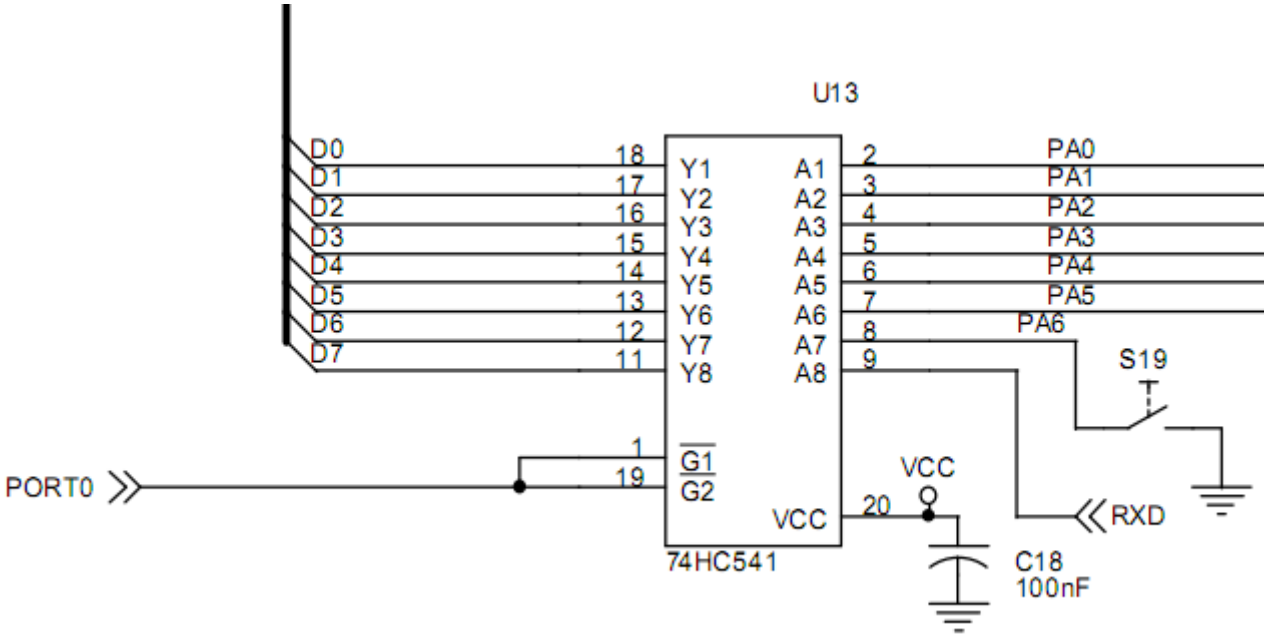
EXPANSION BUS HEADER

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.



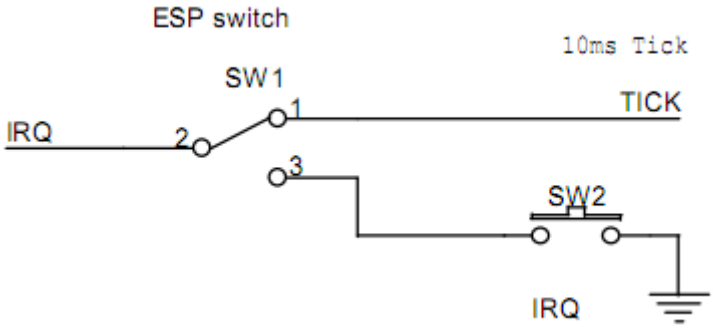
USER KEY

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.



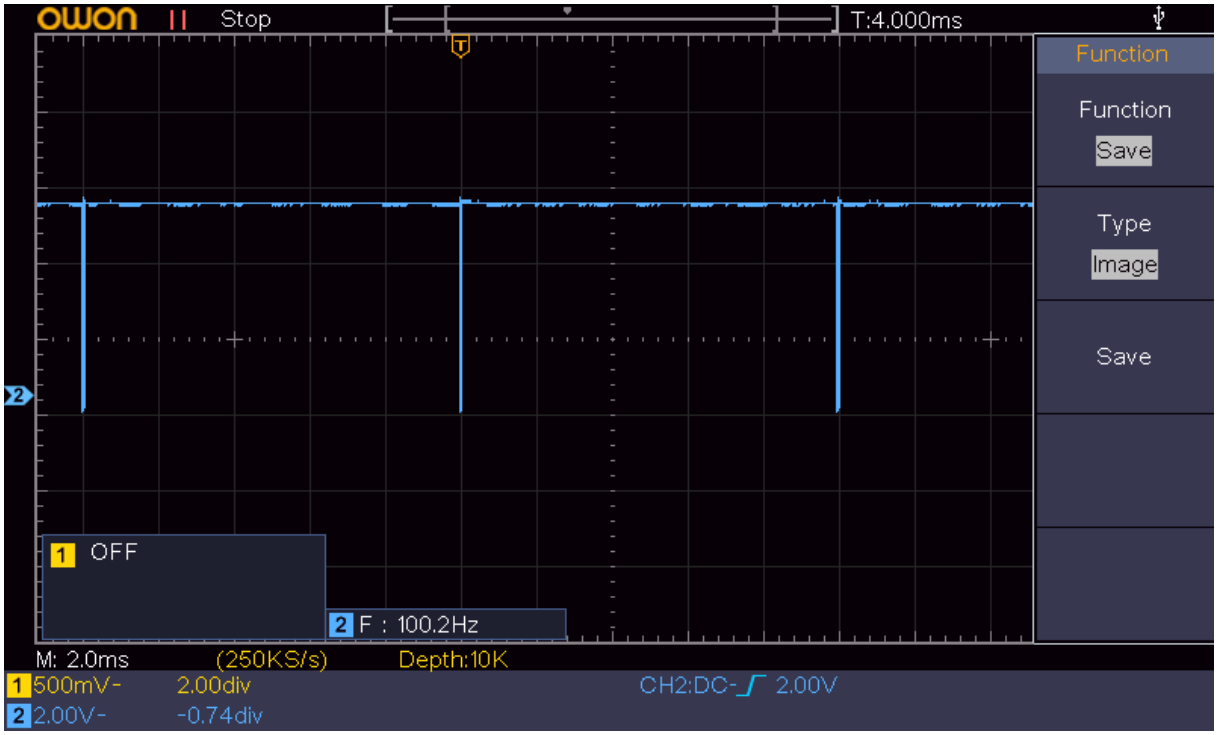
10ms TICK GENERATOR

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.

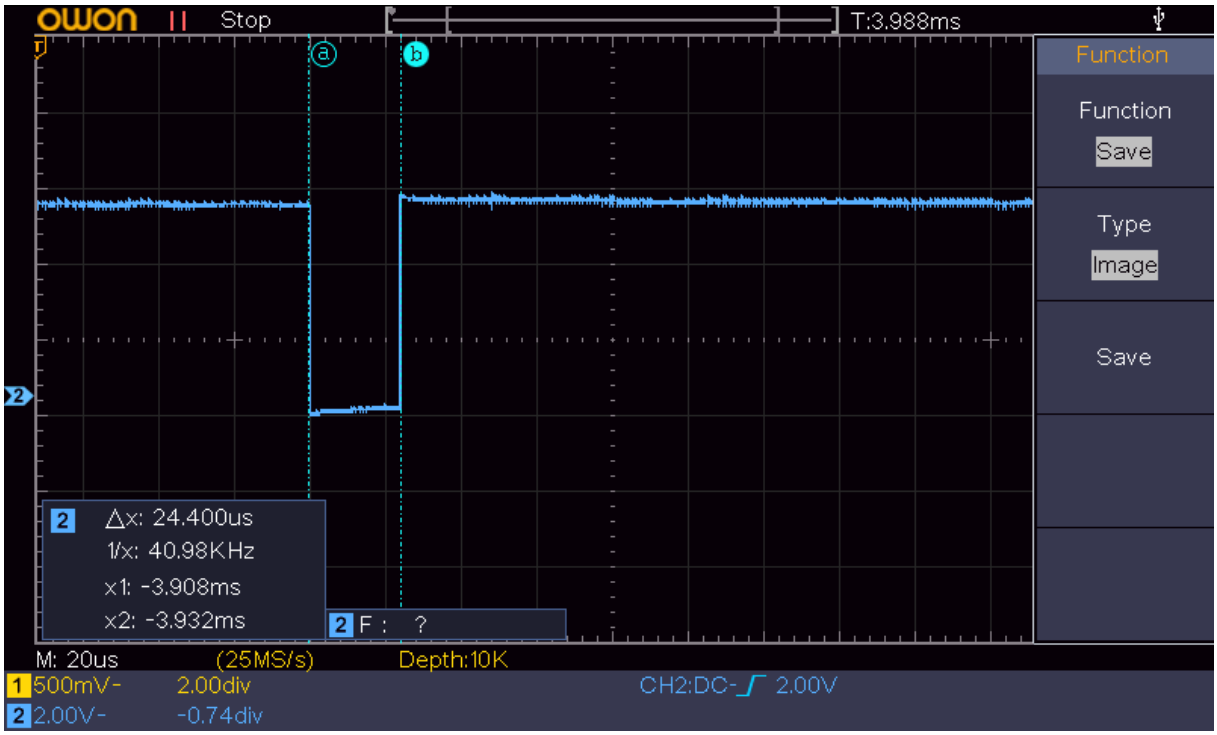


Tick is a 10ms periodic signal for triggering the 6502 IRQ pin. When select SW1 to Tick, the 6502 CPU can be triggered by a maskable interrupt. The 100Hz tick or 10ms tick can be used to produce tasks that executed with multiple of tick. The 6502 kit lab book will show how to use 10ms tick to make a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

The 10ms tick signal measured at P3.7 pin.

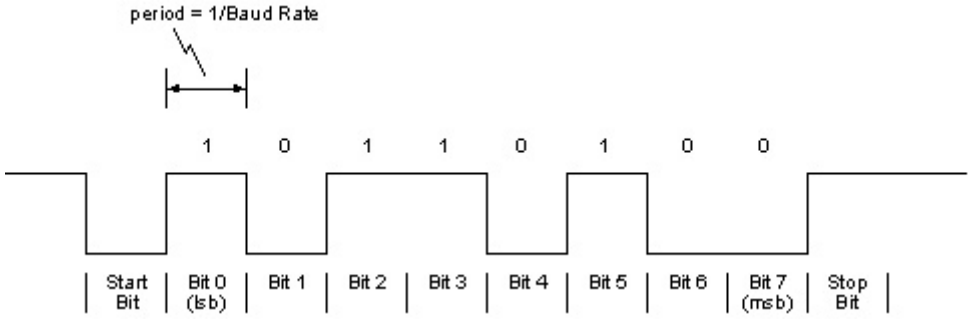


The width of negative pulse is approx. 24µs.



DATA FRAME for UART COMMUNICATION

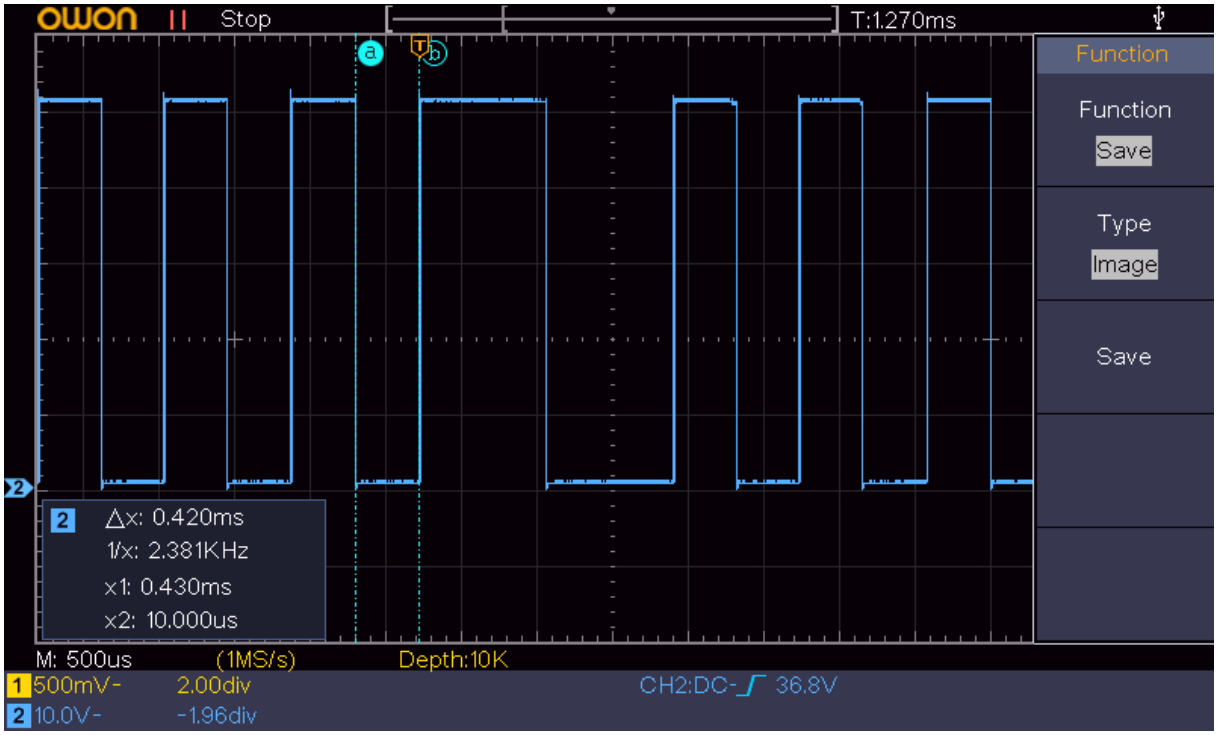
Serial data communicated between kit and terminal is asynchronous format. The 6502 kit has no UART chip, instead it uses software controlled to produce bit rate of 2400 bit/s. The data frame is composed of start bit, 8-data bit and stop bit. For our kit, period = 1/2400 = 417 microseconds.



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

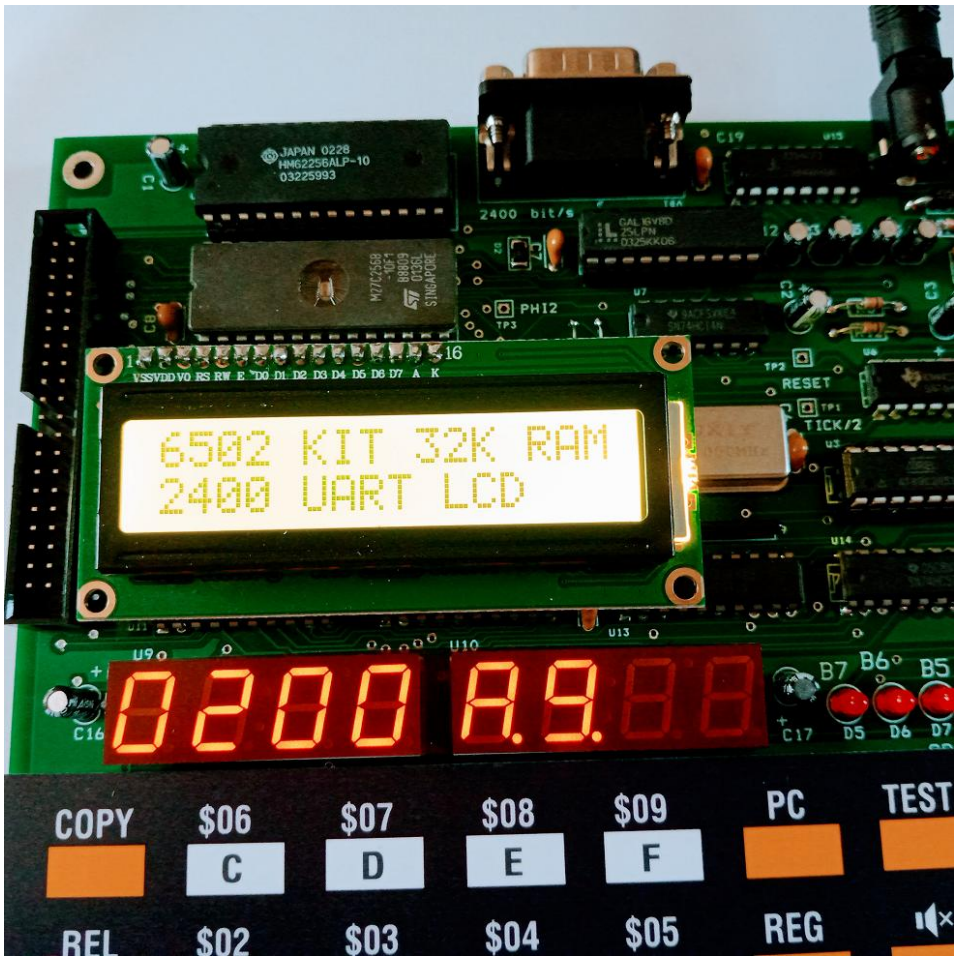
Sample UART waveform with $35 sending, 00110101. Measured period at D3, Δx is 420 microseconds.

STR D0 D1 D2 D3 D4 D5 D6 D7 STP



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.



Note: Insert/Remove the LCD module must be done when the kit is powered off.

The monitor program tests the LCD display on power up. If it was connected, the monitor program will send the cold boot text to the LCD automatically.

The LCD driver subroutines are provided in the monitor program. See the monitor listing for location of the LCD driver.

LOGIC PROBE POWER SUPPLY

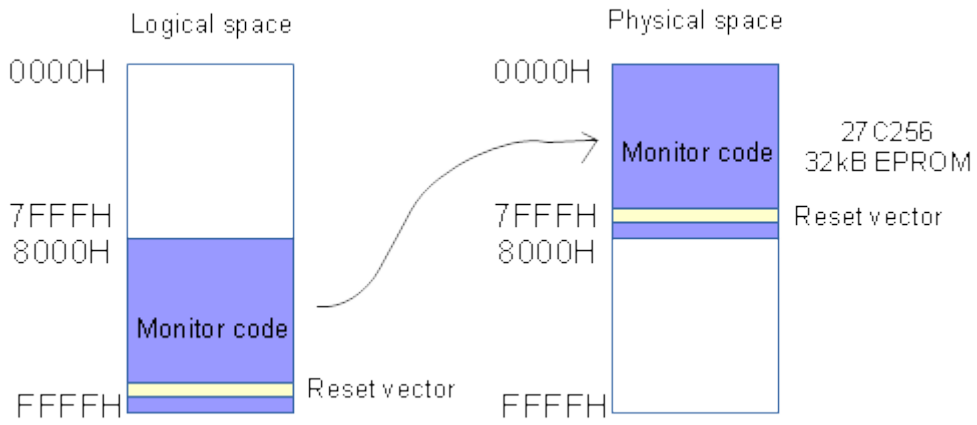
The kit provides test points TP4(+5V) and TP5(GND) for using the logic probe. Students may learn digital logic signals with logic probe easily. The important signals are RESET (TP2) and PHI2 clock (TP3). Tick signal, however indicated by D1 LED blinking. Logic probe can test it at P3.7 of the 89C2051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

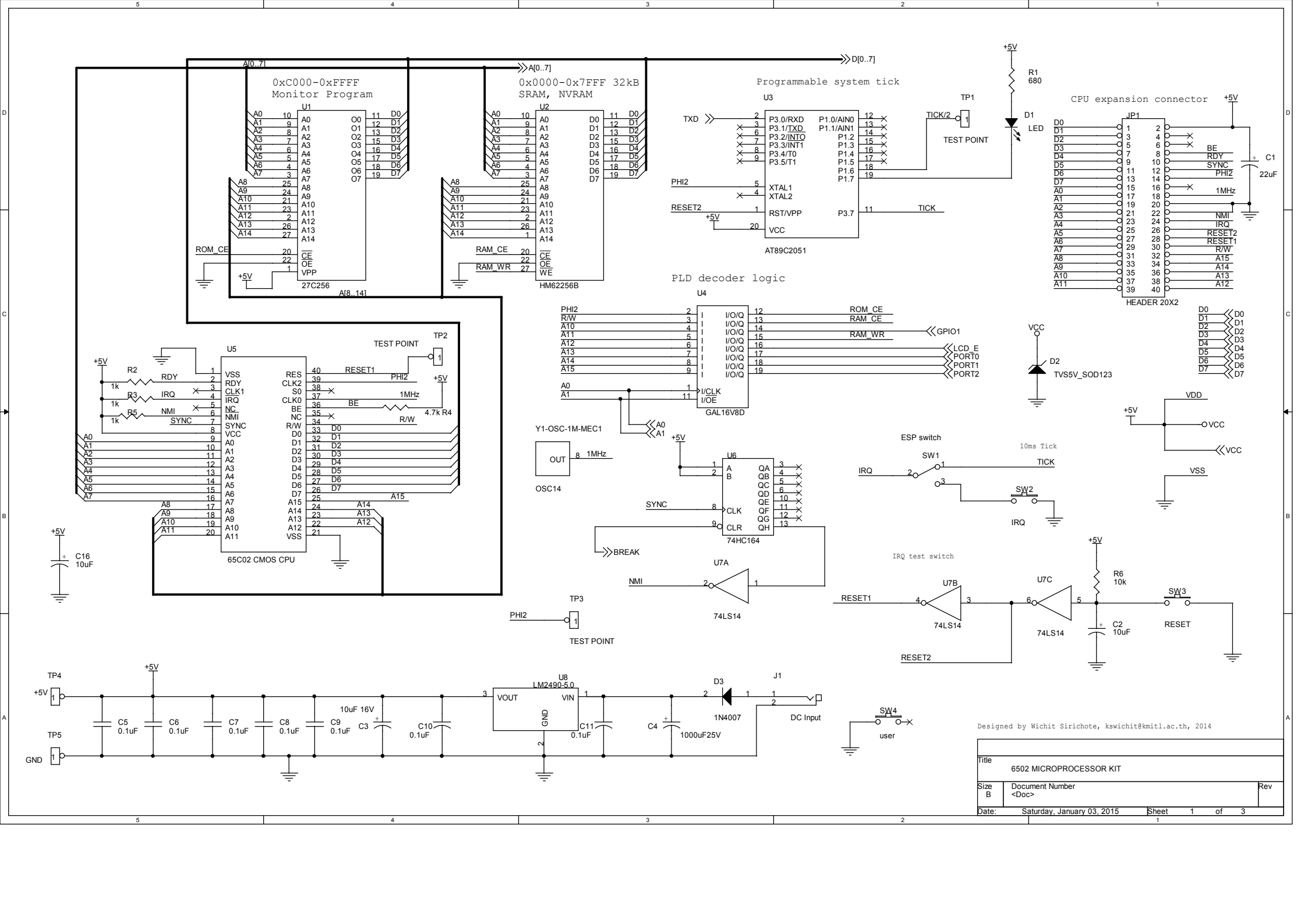


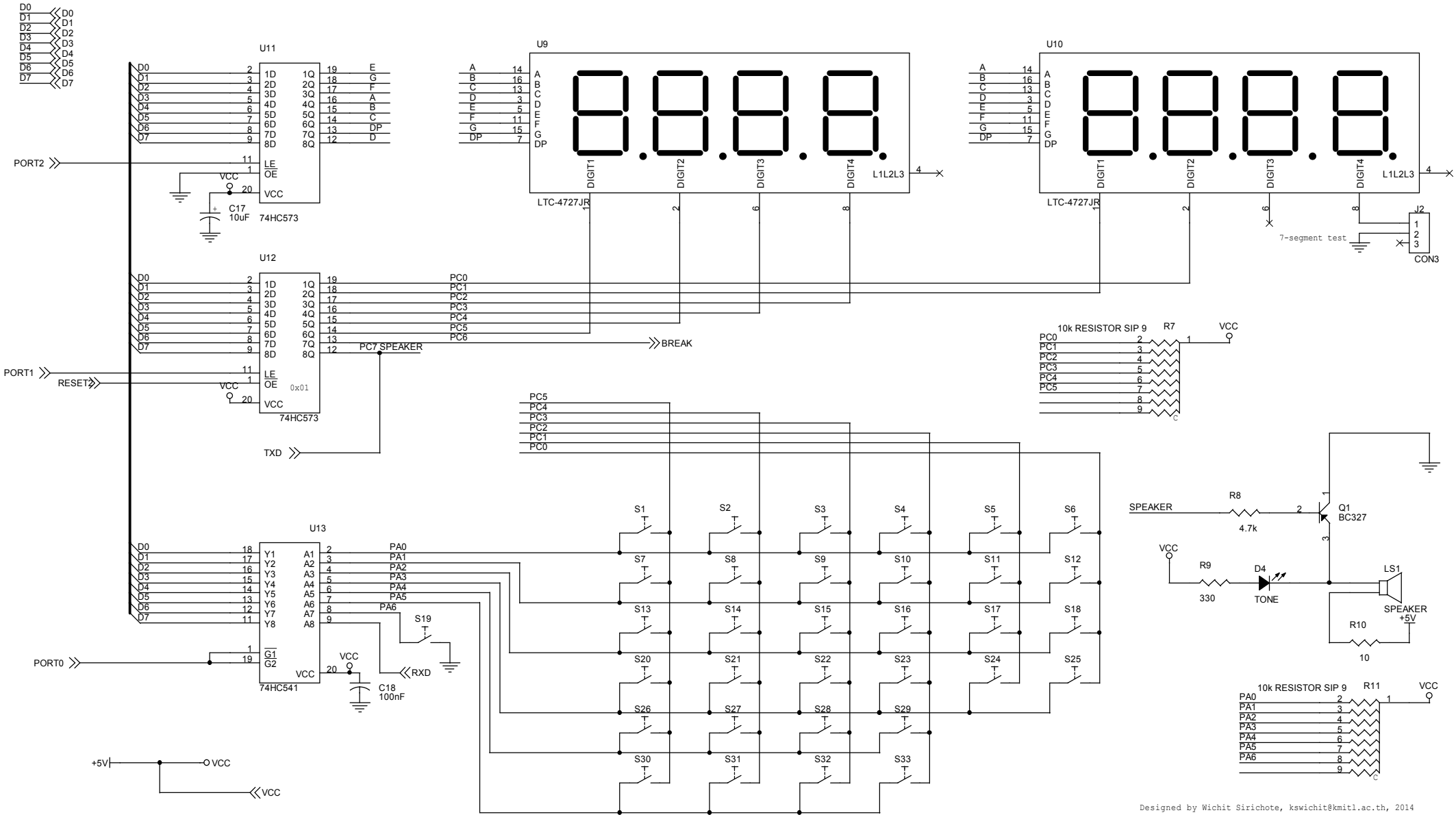
Note:

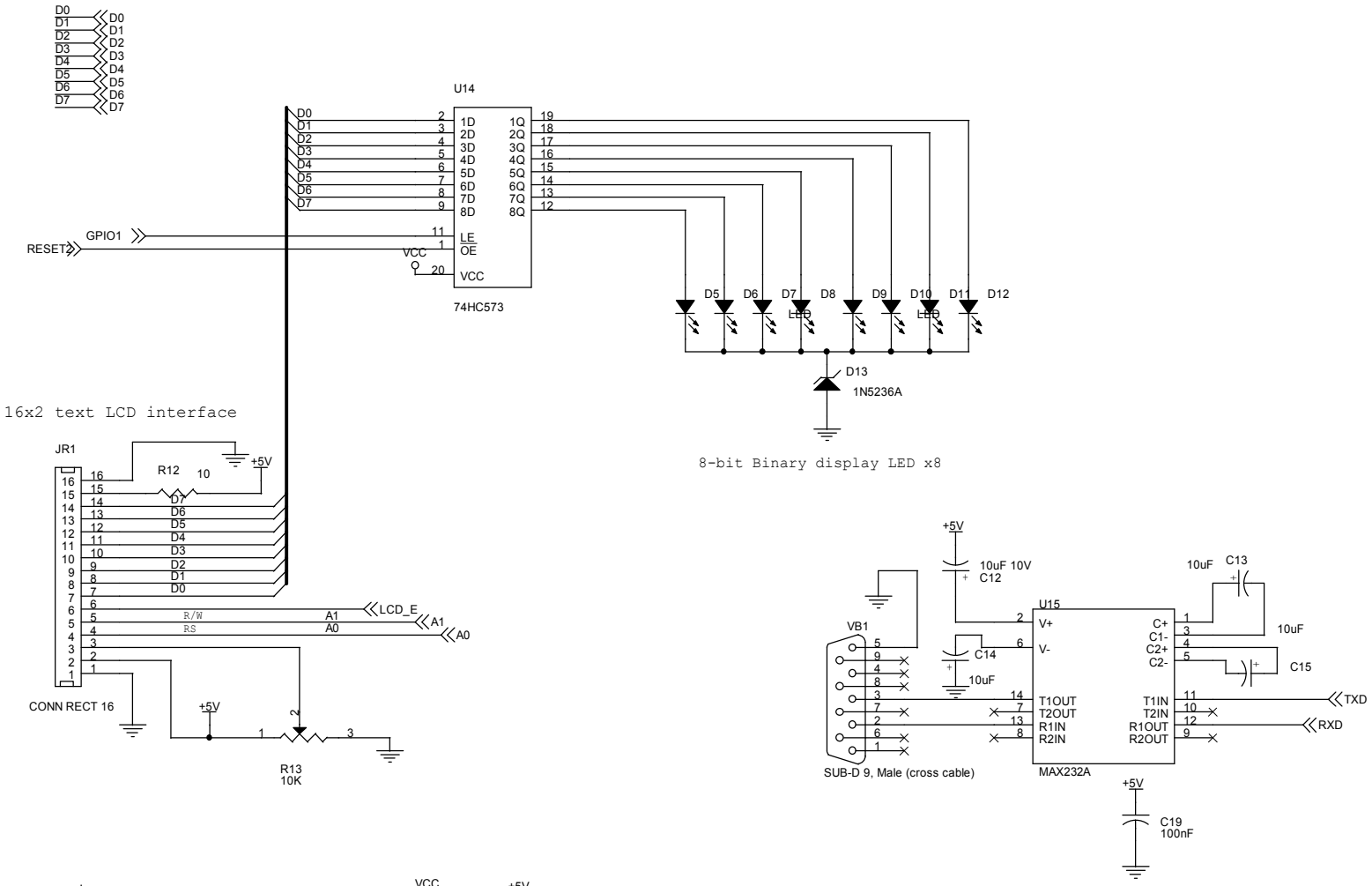
1. The Ceramic EPROM with window will need UV light to erase the memory cell.

2. We can use Flash Memory, SST27SF256 for developing the monitor program. The Flash EPROM, is Flash erasing, no need UV light. The MiniPro TL866CS can program the Flash EPROM directly.

HARDWARE SCHEMATIC and PARTS LIST


0xC000-0xFFFF
Monitor Program
U1
27C256
0x0000-0x7FFF 32kB
SRAM, NVRAM
U2
HM62256B
Programmable system tick
U3
AT89C2051
TP1
TEST POINT
TICK/2
R1
680
D1
LED
CPU expansion connector
JP1
HEADER 20X2
C1
22uF
PLD decoder logic
U4
GAL16V8D
D2
TVS5V_SOD123
U5
65C02 CMOS CPU
TP2
TEST POINT
R2
R3
R5
4.7k R4
Y1-OSC-1M-MEC1
OSC14
U6
74HC164
ESP switch
SW1
SW2
10ms Tick
IRQ test switch
U7A
U7B
U7C
74LS14
R6
10k
SW3
RESET
C2
10uF
TP3
TEST POINT
C16
10uF
U8
LM2490-5.0
D3
1N4007
J1
DC Input
SW4
user
C3
10uF 16V
C4
1000uF25V
C5
0.1uF
C6
0.1uF
C7
0.1uF
C8
0.1uF
C9
0.1uF
C10
0.1uF
C11
0.1uF
TP4
TP5
Designed by Wichit Sirichote, kswichit@kmitl.ac.th, 2014
Title
6502 MICROPROCESSOR KIT
Size
B
Document Number
<Doc>
Rev
Date:
Saturday, January 03, 2015
Sheet
1
of
3


U11
74HC573
C17
10uF
U9
LTC-4727JR
DIGIT1
DIGIT2
DIGIT3
DIGIT4
L1L2L3
U10
LTC-4727JR
J2
7-segment test
CON3
U12
74HC573
PC7 SPEAKER
BREAK
TXD
PORT1
RESET
PORT2
10k RESISTOR SIP 9
R7
VCC
U13
74HC541
PORT0
RXD
C18
100nF
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
SPEAKER
R8
4.7k
Q1
BC327
R9
330
D4
TONE
LS1
SPEAKER
+5V
R10
10
R11
+5V
Designed by Wichit Sirichote, kswichit@kmitl.ac.th, 2014
Title
6502 MICROPROCESSOR KIT
Size
B
Document Number
<Doc>
Rev
1
Date:
Saturday, January 03, 2015
Sheet
2
of
3



Designed by Wichit Sirichote, kswichit@kmitl.ac.th, 2014

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

PARTS LIST

Semiconductors

U1 27C256, 32kB EPROM
U2 HM62256B, 32kB Static RAM
U3 AT89C2051, 20-pin DIP microcontroller
U4 GAL16V8D, programmable logic device
U5 65C02 CMOS CPU
U6 74HC164, shift register
U7 74LS14, inverter
U8 LM7805, voltage regulator
U10,U9 LTC-4727, 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 1N4001 rectifying diode
D4 3mm LED
D13 1N5227A
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

JP1 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, wichit.sirichote@gmail.com
0004   0000             ; Copyright (c) 2016
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 pro
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 bloc
0014   0000             ;
0015   0000             ; The source code is for study and freely modifications.
0016   0000             ;
0017   0000             ;
0018   0000             ; 26 DECEMBER 2014
0019   0000             ; 27 DECEMBER 2014
0020   0000             ;   -ADD OUT OF RANGE CHECK FOR RELATIVE BYTE CALCULATION
0021   0000             ;   -ADD DOWNLOAD HEX FILE TO MONITOR SOURCE
0022   0000             ; 29 DECEMBER 2014
0023   0000             ;   -ADD START MESSAGE ON COLD BOOT
0024   0000             ; 30 DECEMBER 2014
0025   0000             ;   -TEST SINGLE STEP WITH 74LS164
0026   0000             ;   - ADD REPEAT KEY
0027   0000             ;   - LOWER REPEAT SPEED
0028   0000             ;   - REMOVE ACCUMALATOR DISPLAY ON BREAK
0029   0000             ;  2 JANUARY 2015
0030   0000             ;  - REMOVE BINARY DISPLAY IN REGISTER MODE
0031   0000             ;  - ADD REGISTER MODE DISPLAY FOR 10 BYTES ZERO PAGE, $00 T
0032   0000             ;  3 JANUARY 2015
0033   0000             ;  - PROVIDE PROGRAM COUNTER SAVING FOR SINGLE STEP RUNNING
0034   0000             ;    now user may change display address, data, to get back
0035   0000             ;    press PC key to restore it, then press step
0036   0000             ;
0037   0000             ;  27 FEBRUARY 2015
0038   0000             ; LOWER BRIGHTNESS OF THE 7-SEGMENT
0039   0000             ; CALIBRATE BEEP FREQUENCY TO 523HZ
0040   0000             ; 2 MARCH 2015
0041   0000             ; ADD HEX FILE DOWNLOAD USING MOS HEX FORMAT
0042   0000             ; NOW THE BOARD CAN ACCEPT BOTH INTEL AND MOS HEX FILE FORMA
0043   0000             ; NO ERROR, THE GPIO1 WILL DISPLAY 0D (CR), IF ERROR IT WILL
0044   0000             ;
0045   0000             ; 21 MARCH 2016 FIX COLD BOOT MESSAGE, MAKE BEEP ON COLD BOO
0046   0000             ; 6 AUGUST 2016 ADJUST BRIGHTNESS CONTROL FOR NUMBER 1 AND 1
0047   0000             ;               ADJUST BEEP FREQUENCY
0048   0000
0049   0000             ; 14 JANUARY 2016 ADD SEI INSTRUCTION
0050   0000
0051   0000             ; 14 JUNE 2021 ADD INS AND DEL KEYS
0052   0000
0053   0000             ; 24 JUNE 2021 ADD COPY, DUMP, TEST KEYS
0054   0000
0055   0000
0056   0000
0057   0000
0058   0000             ; address of the I/O ports
0059   0000
0060   0000            GPIO1   .EQU 8000H
0061   0000            PORT0   .EQU 8001H
0062   0000            PORT1   .EQU 8002H
0063   0000            PORT2   .EQU 8003H
0064   0000
```

```
  65  0065  0000            DIGIT   .EQU 8002H
  66  0066  0000            SEG7    .EQU 8003H
  67  0067  0000            KIN     .EQU 8001H
  68  0068  0000
  69  0069  0000            BUSY              .EQU 80H
  70  0070  0000
  71  0071  0000            ; BELOW LCD'S REGISTERS ARE MAPPED INTO MEMORY SPACE
  72  0072  0000
  73  0073  0000            COMMAND_WRITE     .EQU 9000H
  74  0074  0000            DATA_WRITE        .EQU 9001H
  75  0075  0000            COMMAND_READ      .EQU 9002H
  76  0076  0000            DATA_READ         .EQU 9003H
  77  0077  0000
  78  0078  0000
  79  0079  0000
  80  0080  0000
  81  0081  0000
  82  0082  0000             ; page zero register definition
  83  0083  0000             ; LOCATION $00 TO $7F ARE 128 BYTES FOR USER PROGRAM TESTIN
  84  0084  0000
  85  0085  0000               .DSEG
  86  0086  0080               .ORG 80H
  87  0087  0080
  88  0088  0080            ; zero page memory definitions for monitor use
  89  0089  0080            REG_E   .BLOCK 1
  90  0090  0081            REG_D   .BLOCK 1
  91  0091  0082            REG_B   .BLOCK 1
  92  0092  0083            REG_C   .BLOCK 1
  93  0093  0084            HL      .BLOCK 2           ; 84H = L 85H = H
  94  0094  0086            DE      .BLOCK 2
  95  0095  0088            REG_A   .BLOCK 1
  96  0096  0089
  97  0097  0089            _ERROR  .BLOCK 1           ; ERROR FLAG FOR INTEL HEX FILE D
  98  0098  008A            BCC     .BLOCK 2           ; BYTE CHECK SUM
  99  0099  008C            BUFFER  .BLOCK 6          ; 8BH - 90H PAGE ZERO DISPLAY BUFF
 100  0100  0092            INVALID .BLOCK 1          ; INVALID KEY HAS BEEN PRESSED FLA
 101  0101  0093                                      ; 0 VALID
 102  0102  0093                     ; 1 INVALID
 103  0103  0093
 104  0104  0093            KEY     .BLOCK 1
 105  0105  0094            STATE   .BLOCK 1
 106  0106  0095            ZERO_FLAG .BLOCK 1          ; ZERO WHEN HEX KEY PRESSED FOR
 107  0107  0096
 108  0108  0096            DISPLAY .BLOCK 2         ; display address
 109  0109  0098
 110  0110  0098            PC_USER  .BLOCK 2       ; FOR SAVING CURRENT PC, ON RESET, I
 111  0111  009A            USER_A  .BLOCK 1
 112  0112  009B            USER_X  .BLOCK 1
 113  0113  009C            USER_Y  .BLOCK 1
 114  0114  009D            USER_S  .BLOCK 1          ; USER STACK POINTER
 115  0115  009E            USER_P  .BLOCK 1          ; PROGRAM STATUS REGISTER
 116  0116  009F            SAVE_SP .BLOCK 1          ; SAVE SYSTEM STACK
 117  0117  00A0
 118  0118  00A0            START_ADDRESS .BLOCK 2    ; START ADDRESS
 119  0119  00A2            END_ADDRESS   .BLOCK 2    ; END ADDRESS
 120  0120  00A4            DESTINATION  .BLOCK 2     ; DESTINATION FOR OFFSET BYTE CALCU
 121  0121  00A6            OFFSET_BYTE .BLOCK 2      ; OFFSET BYTE = DESTINATION - START
 122  0122  00A8            COLD    .BLOCK 1          ; COLD BOOT OR WARM BOOT
 123  0123  00A9
 124  0124  00A9            REPDELAY .BLOCK 1
 125  0125  00AA            SAVE_X   .BLOCK 1
 126  0126  00AB            SAVE_Y   .BLOCK 1
 127  0127  00AC
 128  0128  00AC            DEBUG    .BLOCK 2      ; FOR PROGRAM DEBUGGING
```

```
0129   00AE
0130   00AE              MUTE     .BLOCK 1      ; BEEP/NO BEEP
0131   00AF              SEC      .BLOCK 1
0132   00B0              SEC100   .BLOCK 1
0133   00B1
0134   00B1
0135   00B1
0136   00B1                        .CSEG
0137   00B1
0138   C000                        .ORG 0C000H    ; START ADDRESS FOR ROM
0139   C000                      ;  .ORG 1000H     ; START ADDRESS FOR CODE TESTING I
0140   C000
0141   C000 A9 BF                        LDA #$BF ; turn off break signal
0142   C002 8D 02 80          STA PORT1
0143   C005 A9 00             LDA #0
0144   C007 8D 03 80          STA PORT2 ; turn of 7-segment
0145   C00A
0146   C00A                 ; power up delay
0147   C00A
0148   C00A A2 00             LDX #0
0149   C00C CA           POWER_UP_DELAY DEX
0150   C00D D0 FD                  BNE POWER_UP_DELAY
0151   C00F
0152   C00F 78                SEI      ; disable interrupt
0153   C010
0154   C010                         ; jump to main code
0155   C010
0156   C010 4C 9E CB                JMP MAIN
0157   C013
0158   C013              ;----------------------- 2400 BIT/S SOFTWARE UART ----------
0159   C013              ; one bit delay for 2400 bit/s UART
0160   C013
0161   C013 A0 4C        BIT_DELAY LDY #76      ; 1190 Hz TEST AT 1MHZ OSCILLATOR
0162   C015 88           LOOP      DEY
0163   C016 D0 FD                  BNE LOOP
0164   C018 60              RTS
0165   C019
0166   C019              ; 1.5 bit delay
0167   C019
0168   C019 A0 72        BIT1_5_DELAY LDY #114      ; DELAY 1.5 BIT
0169   C01B 88           LOOP1      DEY
0170   C01C D0 FD                  BNE LOOP1
0171   C01E 60              RTS
0172   C01F
0173   C01F              ; SEND ASCII LETTER TO TERMINAL
0174   C01F              ; ENTRY: A
0175   C01F
0176   C01F 85 80        SEND_BYTE: STA REG_E  ; SAVE ACCUMULATOR
0177   C021
0178   C021 A9 3F                  LDA #3FH    ; start bit is zero
0179   C023 8D 02 80           STA PORT1
0180   C026 20 13 C0               JSR BIT_DELAY    ; delay one bit
0181   C029
0182   C029 A9 08                  LDA #8      ; 8-data bit wil be sent
0183   C02B 85 81              STA REG_D
0184   C02D
0185   C02D A5 80        CHK_BIT:   LDA REG_E
0186   C02F 29 01                   AND #1
0187   C031 F0 08            BEQ SEND_ZERO
0188   C033
0189   C033 A9 BF            LDA #0BFH
0190   C035 8D 02 80         STA PORT1
0191   C038
0192   C038 4C 43 C0                JMP NEXT_BIT
```

```
0193   C03B
0194   C03B
0195   C03B A9 3F       SEND_ZERO: LDA #3FH
0196   C03D 8D 02 80               STA PORT1
0197   C040 4C 43 C0         JMP NEXT_BIT
0198   C043
0199   C043 20 13 C0    NEXT_BIT:  JSR BIT_DELAY
0200   C046
0201   C046 46 80                  LSR REG_E
0202   C048 C6 81            DEC REG_D
0203   C04A D0 E1                  BNE CHK_BIT
0204   C04C
0205   C04C A9 BF                  LDA #0BFH
0206   C04E 8D 02 80         STA PORT1
0207   C051 20 13 C0         JSR BIT_DELAY
0208   C054 60               RTS
0209   C055
0210   C055
0211   C055             ; RECEIVE BYTE FROM 2400 BIT/S TERMINAL
0212   C055             ; EXIT: A
0213   C055
0214   C055 AD 01 80    CIN  LDA PORT0
0215   C058 29 80                AND #80H
0216   C05A D0 F9          BNE CIN
0217   C05C
0218   C05C 20 19 C0       JSR BIT1_5_DELAY
0219   C05F
0220   C05F A9 07          LDA #7
0221   C061 85 81          STA REG_D
0222   C063 A9 00          LDA #0
0223   C065 85 80          STA REG_E
0224   C067
0225   C067
0226   C067
0227   C067 AD 01 80    CHK_BIT_RX   LDA PORT0
0228   C06A 29 80                  AND #80H
0229   C06C D0 09                  BNE BIT_IS_ONE
0230   C06E
0231   C06E A5 80                  LDA REG_E
0232   C070 29 7F            AND #7FH
0233   C072 85 80            STA REG_E
0234   C074 4C 80 C0         JMP NEXT_BIT_RX
0235   C077
0236   C077 A5 80       BIT_IS_ONE LDA REG_E
0237   C079 09 80                  ORA #80H
0238   C07B 85 80            STA REG_E
0239   C07D 4C 80 C0               JMP NEXT_BIT_RX
0240   C080
0241   C080 20 13 C0    NEXT_BIT_RX  JSR BIT_DELAY
0242   C083
0243   C083 46 80                  LSR REG_E
0244   C085
0245   C085 C6 81                  DEC REG_D
0246   C087 D0 DE            BNE CHK_BIT_RX
0247   C089
0248   C089 20 13 C0         JSR BIT_DELAY    ; CENTER OF STOP BIT
0249   C08C
0250   C08C A5 80            LDA REG_E
0251   C08E
0252   C08E 60               RTS
0253   C08F
0254   C08F
0255   C08F             ; PRINT TEXT FROM STRING AREA
0256   C08F             ; ENTRY: X POINTED TO OFFSET
```

```
0257   C08F
0258   C08F BD 00 EF    PSTRING   LDA TEXT1,X
0259   C092 C9 00                 CMP #0
0260   C094 D0 01           BNE PRINT_IT
0261   C096 60              RTS
0262   C097
0263   C097 20 1F C0    PRINT_IT  JSR SEND_BYTE
0264   C09A E8                    INX
0265   C09B 4C 8F C0        JMP PSTRING
0266   C09E
0267   C09E             CR        .EQU 0DH
0268   C09E             LF        .EQU 0AH
0269   C09E             EOS       .EQU 0
0270   C09E
0271   C09E             ;NEW LINE
0272   C09E             ; PRINT CR, LF
0273   C09E
0274   C09E A9 0D       NEW_LINE     LDA #0DH
0275   C0A0 20 1F C0           JSR SEND_BYTE
0276   C0A3 A9 0A              LDA #0AH
0277   C0A5 20 1F C0           JSR SEND_BYTE
0278   C0A8 60                 RTS
0279   C0A9
0280   C0A9
0281   C0A9             ; WRITE NIBBLE TO TERMINAL
0282   C0A9 29 0F       OUT1X        AND #0FH
0283   C0AB 18                       CLC
0284   C0AC 69 30              ADC #30H
0285   C0AE C9 3A                    CMP  #3AH
0286   C0B0 90 03              BCC OUT1X1
0287   C0B2 18                 CLC
0288   C0B3 69 07              ADC #7
0289   C0B5 20 1F C0    OUT1X1      JSR SEND_BYTE
0290   C0B8 60                 RTS
0291   C0B9
0292   C0B9
0293   C0B9 48          OUT2X        PHA
0294   C0BA
0295   C0BA 4A                 LSR A
0296   C0BB 4A                       LSR A
0297   C0BC 4A                 LSR A
0298   C0BD 4A                 LSR A
0299   C0BE
0300   C0BE                  ;  STA GPIO1
0301   C0BE
0302   C0BE 20 A9 C0           JSR OUT1X
0303   C0C1 68                 PLA
0304   C0C2 20 A9 C0           JSR OUT1X
0305   C0C5 60                 RTS
0306   C0C6
0307   C0C6
0308   C0C6             ; INCREMENT HL
0309   C0C6             ; INCREMENT 16-BIT POINTER FOR 16-BIT MEMORY ACCESS
0310   C0C6
0311   C0C6 18          INC_HL       CLC
0312   C0C7 A5 84                    LDA HL
0313   C0C9 69 01              ADC #1
0314   C0CB 85 84              STA HL
0315   C0CD A5 85              LDA HL+1
0316   C0CF 69 00              ADC #0
0317   C0D1 85 85              STA HL+1
0318   C0D3 60                 RTS
0319   C0D4
0320   C0D4             ; INCREMENT DE
```

```
0321   C0D4             ; INCREMENT 16-BIT POINTER FOR 16-BIT MEMORY ACCESS
0322   C0D4
0323   C0D4 18          INC_DE       CLC
0324   C0D5 A5 86                    LDA DE
0325   C0D7 69 01              ADC #1
0326   C0D9 85 86              STA DE
0327   C0DB A5 87              LDA DE+1
0328   C0DD 69 00              ADC #0
0329   C0DF 85 87              STA DE+1
0330   C0E1 60                 RTS
0331   C0E2
0332   C0E2             ; DECREMENT HL
0333   C0E2             ; DECREMENT 16-BIT POINTER FOR 16-BIT MEMORY ACCESS
0334   C0E2
0335   C0E2 18          DEC_HL       CLC
0336   C0E3 A5 84                    LDA HL
0337   C0E5 69 FF              ADC #0FFH
0338   C0E7 85 84              STA HL
0339   C0E9 A5 85              LDA HL+1
0340   C0EB 69 FF              ADC #0FFH
0341   C0ED 85 85              STA HL+1
0342   C0EF 60                 RTS
0343   C0F0
0344   C0F0             ; DECREMENT DE
0345   C0F0             ; DECREMENT 16-BIT POINTER FOR 16-BIT MEMORY ACCESS
0346   C0F0
0347   C0F0 18          DEC_DE       CLC
0348   C0F1 A5 86                    LDA DE
0349   C0F3 69 FF              ADC #0FFH
0350   C0F5 85 86              STA DE
0351   C0F7 A5 87              LDA DE+1
0352   C0F9 69 FF              ADC #0FFH
0353   C0FB 85 87              STA DE+1
0354   C0FD 60                 RTS
0355   C0FE
0356   C0FE             ; PRINT LINE OF MEMORY POINTED TO HL
0357   C0FE             ; ENTRY: HL
0358   C0FE             ; exit: display PC
0359   C0FE
0360   C0FE 20 9E C0    PRINT_LINE   JSR NEW_LINE
0361   C101 A9 10                    LDA #16
0362   C103 85 83                    STA REG_C
0363   C105
0364   C105
0365   C105
0366   C105 A5 85                    LDA HL+1
0367   C107 20 B9 C0           JSR OUT2X
0368   C10A A5 84              LDA HL
0369   C10C 20 B9 C0           JSR OUT2X
0370   C10F
0371   C10F A9 3A              LDA #':'
0372   C111 20 1F C0           JSR SEND_BYTE
0373   C114
0374   C114 A0 00       PRINT_LINE2  LDY #0
0375   C116 B1 84                    LDA (HL),Y
0376   C118
0377   C118 20 B9 C0           JSR OUT2X
0378   C11B
0379   C11B A9 20              LDA #' '
0380   C11D 20 1F C0           JSR SEND_BYTE
0381   C120
0382   C120 20 C6 C0           JSR INC_HL
0383   C123
0384   C123 C6 83              DEC REG_C
```

```
0385  C125
0386  C125 D0 ED            BNE PRINT_LINE2
0387  C127
0388  C127 60               RTS
0389  C128
0390  C128              ; PRINT ASCII FOR MEMORY DUMP
0391  C128              ; ENTRY: DISPLAY ADDRESS
0392  C128              ;
0393  C128
0394  C128 A9 10        PRINT_ASCII  LDA #16
0395  C12A 85 82                     STA REG_B
0396  C12C
0397  C12C              PRINT_ASCII2
0398  C12C A0 00              LDY #0
0399  C12E B1 84                    LDA (HL),Y
0400  C130
0401  C130 C9 20              CMP #32
0402  C132 10 05                    BPL CHECK_80H
0403  C134 A9 2E              LDA #'.'
0404  C136 4C 3F C1           JMP PRINT_ASCII3
0405  C139
0406  C139 C9 80        CHECK_80H    CMP #$80
0407  C13B 30 02                     BMI PRINT_ASCII3
0408  C13D A9 2E              LDA #'.'
0409  C13F
0410  C13F
0411  C13F              PRINT_ASCII3
0412  C13F 20 1F C0           JSR SEND_BYTE
0413  C142
0414  C142 20 C6 C0           JSR INC_HL
0415  C145
0416  C145 C6 82              DEC REG_B
0417  C147
0418  C147 D0 E3              BNE PRINT_ASCII2
0419  C149 60                 RTS
0420  C14A
0421  C14A
0422  C14A
0423  C14A
0424  C14A
0425  C14A              ; CONVERT ASCII TO HEX
0426  C14A              ; ENTRY: A
0427  C14A
0428  C14A 38           TO_HEX     SEC
0429  C14B E9 30                   SBC #30H
0430  C14D C9 10           CMP #10H
0431  C14F 90 05           BCC ZERO_NINE
0432  C151 29 DF           AND #11011111B
0433  C153 38              SEC
0434  C154 E9 07           SBC #7
0435  C156
0436  C156 60           ZERO_NINE  RTS
0437  C157
0438  C157              ; CONVERT TWO ASCII LETTERS  TO SINGLE BYTE
0439  C157              ; EXIT: A
0440  C157
0441  C157 20 55 C0     GET_HEX    JSR CIN
0442  C15A 20 4A C1          JSR TO_HEX
0443  C15D 0A                      ASL A
0444  C15E 0A              ASL A
0445  C15F 0A              ASL A
0446  C160 0A              ASL A
0447  C161
0448  C161 8D 00 80        STA GPIO1
```

```
0449   C164
0450   C164 85 88           STA REG_A
0451   C166
0452   C166 20 55 C0        JSR CIN
0453   C169 20 4A C1        JSR TO_HEX
0454   C16C 18              CLC
0455   C16D 65 88           ADC REG_A
0456   C16F
0457   C16F 60              RTS
0458   C170
0459   C170             ; CONVERT TWO ASCII LETTERS  TO SINGLE BYTE
0460   C170             ; EXIT: A
0461   C170
0462   C170 20 55 C0    GET_HEX2   JSR CIN
0463   C173 48                     PHA
0464   C174 20 1F C0        JSR SEND_BYTE   ; ECHO TO TERMINAL
0465   C177 68              PLA
0466   C178 20 4A C1        JSR TO_HEX
0467   C17B 0A                    ASL A
0468   C17C 0A              ASL A
0469   C17D 0A              ASL A
0470   C17E 0A              ASL A
0471   C17F
0472   C17F 8D 00 80        STA GPIO1
0473   C182
0474   C182 85 88           STA REG_A
0475   C184
0476   C184 20 55 C0        JSR CIN
0477   C187 48              PHA
0478   C188 20 1F C0        JSR SEND_BYTE
0479   C18B 68              PLA
0480   C18C 20 4A C1        JSR TO_HEX
0481   C18F 18              CLC
0482   C190 65 88           ADC REG_A
0483   C192
0484   C192 60              RTS
0485   C193
0486   C193             ;-------------------------------------------------------
0487   C193             SET_NEW_ADDRESS
0488   C193
0489   C193 20 1F C0              JSR SEND_BYTE
0490   C196 A2 3F             LDX #PROMPT&00FFH
0491   C198 20 8F C0              JSR PSTRING
0492   C19B 20 70 C1          JSR GET_HEX2
0493   C19E 85 85             STA HL+1
0494   C1A0 20 70 C1          JSR GET_HEX2
0495   C1A3 85 84             STA HL
0496   C1A5 60                RTS
0497   C1A6
0498   C1A6 18          ADD_BCC   CLC
0499   C1A7 65 8A                 ADC BCC
0500   C1A9 85 8A             STA BCC
0501   C1AB 60                RTS
0502   C1AC
0503   C1AC             ;-------------------------------------------------------
0504   C1AC             ; GET_RECORD READS INTEL HEX FILE AND SAVE TO MEMORY
0505   C1AC
0506   C1AC A9 00       GET_RECORD LDA #0
0507   C1AE 85 89           STA _ERROR
0508   C1B0
0509   C1B0 20 55 C0    GET_RECORD1 JSR CIN
0510   C1B3 C9 3A           CMP #':'
0511   C1B5 F0 07           BEQ GET_RECORD2
0512   C1B7
```

```
0513   C1B7 C9 3B           CMP #$3B          ; ';'
0514   C1B9 D0 F5           BNE GET_RECORD1
0515   C1BB
0516   C1BB 4C 3A C2        JMP GET_MOS2
0517   C1BE
0518   C1BE
0519   C1BE             GET_RECORD2
0520   C1BE
0521   C1BE A9 00           LDA #0
0522   C1C0 85 8A           STA BCC
0523   C1C2
0524   C1C2 20 57 C1        JSR GET_HEX
0525   C1C5 85 83           STA REG_C     ; GET NUMBER OF BYTE
0526   C1C7
0527   C1C7 20 A6 C1        JSR ADD_BCC
0528   C1CA
0529   C1CA 20 57 C1        JSR GET_HEX
0530   C1CD 85 85                 STA HL+1
0531   C1CF
0532   C1CF 20 A6 C1        JSR ADD_BCC
0533   C1D2
0534   C1D2 20 57 C1        JSR GET_HEX
0535   C1D5 85 84           STA HL        ; GET LOAD ADDRESS
0536   C1D7
0537   C1D7 20 A6 C1        JSR ADD_BCC
0538   C1DA
0539   C1DA 20 57 C1        JSR GET_HEX
0540   C1DD
0541   C1DD C9 00           CMP #0
0542   C1DF
0543   C1DF F0 14           BEQ DATA_RECORD
0544   C1E1
0545   C1E1 20 55 C0    WAIT_CR     JSR CIN
0546   C1E4 C9 0D           CMP #0DH
0547   C1E6 D0 F9           BNE WAIT_CR
0548   C1E8
0549   C1E8 8D 00 80        STA GPIO1
0550   C1EB
0551   C1EB A5 89           LDA _ERROR
0552   C1ED C9 01           CMP #1
0553   C1EF D0 03           BNE NOERROR
0554   C1F1
0555   C1F1             ; SHOW ERROR ON LED
0556   C1F1               ;  JSR OUT_OFF_RANGE
0557   C1F1
0558   C1F1 8D 00 80        STA GPIO1
0559   C1F4
0560   C1F4             NOERROR
0561   C1F4 60                    RTS
0562   C1F5
0563   C1F5             DATA_RECORD
0564   C1F5
0565   C1F5 20 57 C1        JSR GET_HEX
0566   C1F8 A0 00           LDY #0
0567   C1FA 91 84           STA (HL),Y   ; WRITE TO MEMORY
0568   C1FC
0569   C1FC 20 A6 C1        JSR ADD_BCC
0570   C1FF
0571   C1FF 8D 00 80        STA GPIO1
0572   C202
0573   C202 20 C6 C0        JSR INC_HL
0574   C205
0575   C205 C6 83           DEC REG_C
0576   C207 D0 EC           BNE DATA_RECORD ; UNTIL C=0
```

```
0577   C209
0578   C209 A5 8A                LDA BCC
0579   C20B 49 FF           EOR #0FFH    ; ONE'S COMPLEMENT
0580   C20D 18              CLC
0581   C20E 69 01           ADC #1       ; TWO'S COMPLEMENT
0582   C210 85 8A           STA BCC
0583   C212
0584   C212
0585   C212 20 57 C1        JSR GET_HEX    ; GET BYTE CHECK SUM
0586   C215
0587   C215 C5 8A           CMP BCC; COMPARE WITH BYTE CHECK SUM
0588   C217 F0 04           BEQ SKIP11
0589   C219
0590   C219 A9 01           LDA #1
0591   C21B 85 89           STA _ERROR    ; ERROR FLAG =1
0592   C21D
0593   C21D
0594   C21D             SKIP11
0595   C21D
0596   C21D 4C B0 C1        JMP GET_RECORD1     ; NEXT LINE
0597   C220
0598   C220
0599   C220
0600   C220             SEND_PROMPT
0601   C220
0602   C220 20 9E C0             JSR NEW_LINE
0603   C223 A5 85                 LDA HL+1
0604   C225 20 B9 C0        JSR OUT2X
0605   C228 A5 84           LDA HL
0606   C22A 20 B9 C0        JSR OUT2X
0607   C22D
0608   C22D
0609   C22D A2 3F          LDX #PROMPT&00FFH
0610   C22F 20 8F C0              JSR PSTRING
0611   C232 60                    RTS
0612   C233
0613   C233             ;=============================================================
0614   C233             ; get MOS record
0615   C233             ; sample MOS record
0616   C233             ;
0617   C233             ;18 0200 A9018500182600A5008D0080A200A00088D0FDCAD0F84C05 09
0618   C233             ;01 0218 02 001D
0619   C233             ;00
0620   C233             ;
0621   C233             ; 18 is number of byte
0622   C233             ; 0200 is load address
0623   C233             ; A9, 01, 85.. data byte
0624   C233             ; 09B3 is 16-bit check sum
0625   C233
0626   C233
0627   C233
0628   C233             GET_MOS1
0629   C233 20 55 C0              JSR CIN
0630   C236 C9 3B           CMP #$3B         ; ';'
0631   C238 D0 F9           BNE GET_MOS1
0632   C23A
0633   C23A             GET_MOS2
0634   C23A
0635   C23A A9 00           LDA #0
0636   C23C 85 8A           STA BCC
0637   C23E 85 8B           STA BCC+1  ; MOS uses 16-bit checksum
0638   C240
0639   C240
0640   C240 20 57 C1        JSR GET_HEX
```

```
0641   C243 85 83           STA REG_C     ; GET NUMBER OF BYTE
0642   C245
0643   C245 C9 00           CMP #0
0644   C247 F0 16           BEQ END_RECORD
0645   C249
0646   C249 20 A7 C2        JSR ADD_BCC_MOS
0647   C24C
0648   C24C 20 57 C1        JSR GET_HEX
0649   C24F 85 85                 STA HL+1
0650   C251
0651   C251 20 A7 C2        JSR ADD_BCC_MOS
0652   C254
0653   C254 20 57 C1        JSR GET_HEX
0654   C257 85 84           STA HL        ; GET LOAD ADDRESS
0655   C259
0656   C259 20 A7 C2        JSR ADD_BCC_MOS
0657   C25C
0658   C25C 4C 73 C2        JMP DATA_RECORD2
0659   C25F
0660   C25F 20 55 C0    END_RECORD JSR CIN
0661   C262 C9 0D           CMP #0DH
0662   C264 D0 F9           BNE END_RECORD
0663   C266
0664   C266 8D 00 80        STA GPIO1
0665   C269
0666   C269 A5 89           LDA _ERROR
0667   C26B C9 01           CMP #1
0668   C26D D0 03           BNE NOERROR2
0669   C26F
0670   C26F             ; SHOW ERROR ON LED
0671   C26F
0672   C26F 8D 00 80        STA GPIO1
0673   C272
0674   C272
0675   C272             NOERROR2
0676   C272 60                    RTS
0677   C273
0678   C273             DATA_RECORD2
0679   C273
0680   C273 20 57 C1        JSR GET_HEX
0681   C276 A0 00           LDY #0
0682   C278 91 84           STA (HL),Y   ; WRITE TO MEMORY
0683   C27A
0684   C27A 20 A7 C2        JSR ADD_BCC_MOS
0685   C27D
0686   C27D 8D 00 80        STA GPIO1
0687   C280
0688   C280 20 C6 C0        JSR INC_HL
0689   C283
0690   C283 C6 83           DEC REG_C
0691   C285 D0 EC           BNE DATA_RECORD2 ; UNTIL C=0
0692   C287
0693   C287             ; now get 16-bit check sum
0694   C287
0695   C287 20 57 C1        JSR GET_HEX    ; GET 16-bit CHECK SUM
0696   C28A 85 85           STA HL+1
0697   C28C               ;   STA DEBUG+1
0698   C28C
0699   C28C 20 57 C1        JSR GET_HEX
0700   C28F 85 84           STA HL          ; check sum now stored in HL+1 and HL
0701   C291                     ;  STA DEBUG
0702   C291
0703   C291 A5 8B           LDA BCC+1
0704   C293 C5 85           CMP HL+1
```

```
0705   C295 D0 09           BNE error_mos
0706   C297 A5 8A           LDA BCC
0707   C299 C5 84           CMP HL
0708   C29B D0 03           BNE error_mos
0709   C29D
0710   C29D 4C A4 C2        JMP SKIP12
0711   C2A0
0712   C2A0             error_mos
0713   C2A0 A9 01           LDA #1
0714   C2A2 85 89           STA _ERROR    ; ERROR FLAG =1
0715   C2A4
0716   C2A4
0717   C2A4             SKIP12
0718   C2A4
0719   C2A4 4C 33 C2        JMP GET_MOS1     ; NEXT LINE
0720   C2A7
0721   C2A7
0722   C2A7             ; add 16-bit check sum, stores in BCC+1 and BCC
0723   C2A7
0724   C2A7             ADD_BCC_MOS
0725   C2A7
0726   C2A7 18                   CLC
0727   C2A8 65 8A                ADC BCC
0728   C2AA 85 8A           STA BCC
0729   C2AC A9 00           LDA #0
0730   C2AE 65 8B           ADC BCC+1
0731   C2B0 85 8B           STA BCC+1
0732   C2B2 60              RTS
0733   C2B3
0734   C2B3
0735   C2B3             ;--------------------------------- END UART CODE -----------
0736   C2B3
0737   C2B3             ; SCAN DISPLAY ONLY
0738   C2B3             ; ENTRY: X POINTED TO NEXT MESSAGE BYTE
0739   C2B3             ;        FIX_MESSAGE LOCATION
0740   C2B3
0741   C2B3             SCAN2:
0742   C2B3 86 83         STX REG_C
0743   C2B5 A9 01         LDA #1
0744   C2B7 85 80         STA REG_E
0745   C2B9
0746   C2B9 A9 06         LDA #6
0747   C2BB 85 84         STA HL
0748   C2BD
0749   C2BD             ;to the active column.
0750   C2BD A5 80       KCOL2   LDA REG_E
0751   C2BF
0752   C2BF 49 FF                EOR #0FFH               ; COMPLEMENT IT
0753   C2C1
0754   C2C1 29 BF         AND #0BFH               ; BREAK MUST BE LOGIC '0' TO DISAB
0755   C2C3 8D 02 80      STA DIGIT
0756   C2C6
0757   C2C6 BD 1A CC      LDA START_MSG,X
0758   C2C9 8D 03 80      STA SEG7
0759   C2CC
0760   C2CC A0 05         LDY #$5
0761   C2CE
0762   C2CE 88          DELAY5 DEY
0763   C2CF D0 FD         BNE DELAY5
0764   C2D1
0765   C2D1 A9 00         LDA #0                  ; TURN LED OFF
0766   C2D3 8D 03 80      STA SEG7
0767   C2D6
0768   C2D6 A0 14         LDY #20
```

```
0769   C2D8 88          DELAY55  DEY
0770   C2D9 D0 FD         BNE DELAY55
0771   C2DB
0772   C2DB
0773   C2DB E8            INX
0774   C2DC
0775   C2DC A5 80         LDA REG_E
0776   C2DE 0A            ASL A
0777   C2DF 85 80         STA REG_E
0778   C2E1
0779   C2E1 C6 84         DEC  HL
0780   C2E3 D0 D8         BNE  KCOL2
0781   C2E5
0782   C2E5 A6 83         LDX REG_C
0783   C2E7
0784   C2E7 60            RTS
0785   C2E8
0786   C2E8
0787   C2E8             ; SCAN DISPLAY AND KEYBOARD
0788   C2E8             ; ENTRY: DISPLAY BUFFER IN PAGE 0
0789   C2E8             ; EXIT: KEY = -1 NO KEY PRESSED
0790   C2E8             ;       KEY >=0 KEY POSITION
0791   C2E8             ; REGSITERS USED: X,A,Y
0792   C2E8
0793   C2E8             SCAN1:
0794   C2E8
0795   C2E8
0796   C2E8 A2 00         LDX #0
0797   C2EA
0798   C2EA A9 00         LDA #0
0799   C2EC 85 83         STA REG_C
0800   C2EE
0801   C2EE A9 FF         LDA #-1
0802   C2F0 85 93         STA KEY
0803   C2F2
0804   C2F2 A9 01         LDA #1
0805   C2F4 85 80         STA REG_E
0806   C2F6
0807   C2F6 A9 06         LDA #6
0808   C2F8 85 84         STA HL
0809   C2FA
0810   C2FA             ;to the active column.
0811   C2FA A5 80       KCOL    LDA REG_E
0812   C2FC
0813   C2FC 49 FF                EOR #0FFH               ; COMPLEMENT IT
0814   C2FE 29 BF         AND #0BFH               ; MUST BE LOW FOR BREAK
0815   C300
0816   C300 8D 02 80      STA DIGIT
0817   C303
0818   C303 B5 8C         LDA BUFFER,X
0819   C305 8D 03 80      STA SEG7
0820   C308
0821   C308 29 BF               AND #~$40              ; mask off dot
0822   C30A C9 30               CMP #30H ; is it number 1
0823   C30C D0 03         BNE NOT_ONE
0824   C30E 4C 15 C3      JMP IT_IS_ONE
0825   C311
0826   C311
0827   C311 C9 02       NOT_ONE CMP #2
0828   C313 D0 05               BNE NOT_LINE   ; is it line
0829   C315
0830   C315             IT_IS_ONE
0831   C315
0832   C315 A0 05         LDY #$5
```

```
0833   C317 4C 1C C3            JMP DELAY3
0834   C31A
0835   C31A A0 15       NOT_LINE LDY #$15
0836   C31C 88          DELAY3 DEY
0837   C31D D0 FD         BNE DELAY3
0838   C31F
0839   C31F A9 00         LDA #0                  ; TURN LED OFF
0840   C321 8D 03 80      STA SEG7
0841   C324
0842   C324 A0 32         LDY #50
0843   C326 88          DELAY10  DEY
0844   C327 D0 FD         BNE DELAY10
0845   C329
0846   C329
0847   C329 A9 06         LDA #6
0848   C32B 85 82         STA REG_B
0849   C32D
0850   C32D AD 01 80      LDA KIN
0851   C330
0852   C330 85 81         STA  REG_D
0853   C332
0854   C332
0855   C332 46 81       KROWLSR  REG_D   ;Rotate D 1 bit right, bit 0
0856   C334                      ;of D will be rotated into
0857   C334 B0 04         BCS NOKEY      ;carry flag.
0858   C336
0859   C336 A5 83         LDA REG_C
0860   C338 85 93         STA KEY
0861   C33A
0862   C33A E6 83       NOKEY  INC REG_C    ;Increase current key-code by 1.
0863   C33C
0864   C33C C6 82         DEC REG_B
0865   C33E D0 F2         BNE KROW
0866   C340
0867   C340 E8            INX
0868   C341
0869   C341 A5 80         LDA REG_E
0870   C343 0A            ASL A
0871   C344 85 80         STA REG_E
0872   C346
0873   C346
0874   C346 C6 84         DEC  HL
0875   C348 D0 B0         BNE  KCOL
0876   C34A 60            RTS
0877   C34B
0878   C34B
0879   C34B A0 C8       DEBOUNCE LDY #200
0880   C34D 88          DELAY4   DEY
0881   C34E D0 FD           BNE DELAY4
0882   C350 60              RTS
0883   C351
0884   C351             ;-------------------------------------------------------------
0885   C351
0886   C351 20 E8 C2    SCANKEY JSR SCAN1
0887   C354 A5 93               LDA KEY
0888   C356 C9 FF         CMP #-1
0889   C358 F0 16         BEQ KEY_RELEASED
0890   C35A
0891   C35A AD 01 80      LDA PORT0
0892   C35D 29 40         AND #40H
0893   C35F D0 F0         BNE SCANKEY
0894   C361
0895   C361             ; IF REPEAT KEY WAS PRESSED, SLOW DOWN IT
0896   C361 A9 20                 LDA #20H
```

```
0897   C363 85 A9               STA REPDELAY
0898   C365
0899   C365 20 E8 C2    DISPLAY4  JSR SCAN1
0900   C368 C6 A9           DEC REPDELAY
0901   C36A D0 F9           BNE DISPLAY4
0902   C36C
0903   C36C
0904   C36C A2 00               LDX #0         ; THEN REPEAT KEY PRESS
0905   C36E 86 92               STX INVALID    ; RESET INVALID FLAG
0906   C370             KEY_RELEASED
0907   C370
0908   C370 20 4B C3      JSR DEBOUNCE
0909   C373
0910   C373             UNTIL_PRESS
0911   C373
0912   C373 20 E8 C2      JSR SCAN1
0913   C376 A5 93               LDA KEY
0914   C378 C9 FF         CMP #-1
0915   C37A F0 F7         BEQ UNTIL_PRESS
0916   C37C
0917   C37C 20 4B C3      JSR DEBOUNCE
0918   C37F
0919   C37F 20 E8 C2      JSR SCAN1
0920   C382
0921   C382 A5 93         LDA KEY
0922   C384 AA            TAX
0923   C385 BD 36 CC      LDA KEYTAB,X    ; OPEN TABLE
0924   C388
0925   C388             ; STA GPIO1       ; TEST NOW A IS INTERNAL CODE
0926   C388 60            RTS
0927   C389
0928   C389
0929   C389             ; CONVERT LOW NIBBLE IN ACCUMULATOR TO 7-SEGMENT PATTERN
0930   C389             ; ENTRY: A
0931   C389             ; EXIT: A
0932   C389
0933   C389             NIBBLE_7SEG
0934   C389 AA                    TAX
0935   C38A BD 26 CC              LDA SEGTAB,X
0936   C38D 60               RTS
0937   C38E
0938   C38E
0939   C38E             ; CONVERT BYTE TO 7-SEGMENT PATTERN
0940   C38E             ; ENTRY: A
0941   C38E             ; EXIT: DE
0942   C38E
0943   C38E 48          BYTE_7SEG   PHA
0944   C38F 29 0F                 AND #0FH
0945   C391 20 89 C3              JSR NIBBLE_7SEG
0946   C394 85 86            STA DE
0947   C396 68               PLA
0948   C397 4A                    LSR A
0949   C398 4A               LSR A
0950   C399 4A               LSR A
0951   C39A 4A               LSR A
0952   C39B 20 89 C3         JSR NIBBLE_7SEG
0953   C39E 85 87            STA DE+1
0954   C3A0 60               RTS
0955   C3A1
0956   C3A1             ; CONVERT BYTE TO 7-SEGMENT PATTERN AND SAVE TO DISPLAY BUFF
0957   C3A1             ; ENTRY: A
0958   C3A1
0959   C3A1 48          DATA_DISPLAY PHA   ; SAVE ACCUMULATOR
0960   C3A2 20 8E C3                JSR BYTE_7SEG
```

```
 961  0961   C3A5 A5 86              LDA DE
 962  0962   C3A7 85 8C              STA BUFFER
 963  0963   C3A9 A5 87              LDA DE+1
 964  0964   C3AB 85 8D              STA BUFFER+1
 965  0965   C3AD 68                    PLA
 966  0966   C3AE 60                RTS
 967  0967   C3AF
 968  0968   C3AF             ; CONVERT 16-BIT ADDRESS IN HL AND SAVE IT TO ADDRESS FILED
 969  0969   C3AF             ; ENTRY: HL
 970  0970   C3AF
 971  0971   C3AF             ADDRESS_DISPLAY
 972  0972   C3AF
 973  0973   C3AF A5 84                LDA HL
 974  0974   C3B1 20 8E C3          JSR BYTE_7SEG
 975  0975   C3B4 A5 86              LDA DE
 976  0976   C3B6 85 8E              STA BUFFER+2
 977  0977   C3B8 A5 87              LDA DE+1
 978  0978   C3BA 85 8F              STA BUFFER+3
 979  0979   C3BC A5 85              LDA HL+1
 980  0980   C3BE 20 8E C3          JSR BYTE_7SEG
 981  0981   C3C1 A5 86              LDA DE
 982  0982   C3C3 85 90              STA BUFFER+4
 983  0983   C3C5 A5 87              LDA DE+1
 984  0984   C3C7 85 91              STA BUFFER+5
 985  0985   C3C9 60                 RTS
 986  0986   C3CA
 987  0987   C3CA             ;************************************************************
 988  0988   C3CA             ;
 989  0989   C3CA             ; EXECUTE FUNCTIONS OR HEX KEY ENTERED
 990  0990   C3CA             ; CHECK HEX KEY OR FUNCTIONS KEY
 991  0991   C3CA             ; ENTRY: A
 992  0992   C3CA
 993  0993   C3CA C9 10       KEYEXE     CMP #10H
 994  0994   C3CC B0 56            BCS FUNCTION_KEY
 995  0995   C3CE
 996  0996   C3CE
 997  0997   C3CE             ;HHHHHHHHHHHHHHH   KEY HEX ENTERED HHHHHHHHHHHHHHHHHHHHHHHHH
 998  0998   C3CE
 999  0999   C3CE 85 83                STA REG_C     ; SAVE HEX KEY
1000  1000   C3D0 A5 94            LDA STATE
1001  1001   C3D2
1002  1002   C3D2 C9 01            CMP #1
1003  1003   C3D4 D0 05            BNE CHK_STATE2
1004  1004   C3D6
1005  1005   C3D6 A5 83            LDA REG_C
1006  1006   C3D8 4C 54 C6         JMP HEX_ADDR
1007  1007   C3DB
1008  1008   C3DB C9 02       CHK_STATE2 CMP #2
1009  1009   C3DD D0 03                BNE CHK_STATE3
1010  1010   C3DF 4C 82 C6         JMP HEX_DATA
1011  1011   C3E2
1012  1012   C3E2 C9 03       CHK_STATE3 CMP #3
1013  1013   C3E4 D0 03                BNE CHK_STATE5
1014  1014   C3E6 4C DE C8         JMP HEX_REG
1015  1015   C3E9
1016  1016   C3E9 C9 05       CHK_STATE5 CMP #5
1017  1017   C3EB D0 03                BNE CHK_STATE6
1018  1018   C3ED 4C 3C C6         JMP HEX_REL
1019  1019   C3F0
1020  1020   C3F0 C9 06       CHK_STATE6 CMP #6
1021  1021   C3F2 D0 03                BNE CHK_STATE7
1022  1022   C3F4 4C 48 C6         JMP HEX_REL6
1023  1023   C3F7
1024  1024   C3F7 C9 07       CHK_STATE7 CMP #7
```

```
1025  1025   C3F9 D0 03                BNE CHK_STATE8
1026  1026   C3FB 4C EC C5             JMP HEX_SEND_FILE
1027  1027   C3FE
1028  1028   C3FE C9 08       CHK_STATE8  CMP #8
1029  1029   C400 D0 03                BNE CHK_STATE9
1030  1030   C402 4C F8 C5          JMP HEX_SEND_FILE2
1031  1031   C405
1032  1032   C405 C9 0A       CHK_STATE9  CMP #10
1033  1033   C407 D0 03                BNE CHK_STATE10
1034  1034   C409 4C 18 C6          JMP HEX_COPY10
1035  1035   C40C
1036  1036   C40C C9 0B       CHK_STATE10 CMP #11
1037  1037   C40E D0 03                BNE CHK_STATE11
1038  1038   C410 4C 24 C6          JMP HEX_COPY11
1039  1039   C413
1040  1040   C413 C9 0C       CHK_STATE11 CMP #12
1041  1041   C415 D0 03                BNE CHK_STATE12
1042  1042   C417 4C 30 C6          JMP HEX_COPY12
1043  1043   C41A
1044  1044   C41A             CHK_STATE12
1045  1045   C41A
1046  1046   C41A
1047  1047   C41A
1048  1048   C41A
1049  1049   C41A A5 83                lda REG_C
1050  1050   C41C 8D 00 80             sta GPIO1
1051  1051   C41F A9 01                LDA #1          ; INVALID KEY PRESSED
1052  1052   C421 85 92             STA INVALID
1053  1053   C423
1054  1054   C423
1055  1055   C423
1056  1056   C423
1057  1057   C423
1058  1058   C423             ; HEX KEY WAS PRESSED
1059  1059   C423
1060  1060   C423
1061  1061   C423 60              RTS
1062  1062   C424
1063  1063   C424             ;FFFFFFFFFFFFFFFFFFFFF FUNCTION KEY FFFFFFFFFFFFFFFFFFFFF
1064  1064   C424
1065  1065   C424             FUNCTION_KEY
1066  1066   C424
1067  1067   C424 C9 19                CMP #19H    ; KEY ADDR
1068  1068   C426 D0 03             BNE CHK_FUNC1
1069  1069   C428 4C 79 C5          JMP KEY_ADDR
1070  1070   C42B
1071  1071   C42B C9 14       CHK_FUNC1 CMP #14H   ; KEY DATA
1072  1072   C42D D0 03                BNE CHK_FUNC2
1073  1073   C42F 4C A9 C5          JMP KEY_DATA
1074  1074   C432
1075  1075   C432 C9 10       CHK_FUNC2 CMP #10H   ; KEY +
1076  1076   C434 D0 03                BNE CHK_FUNC3
1077  1077   C436 4C A2 C6          JMP KEY_INC
1078  1078   C439
1079  1079   C439 C9 11       CHK_FUNC3 CMP #11H    ; KEY -
1080  1080   C43B D0 03                BNE CHK_FUNC4
1081  1081   C43D 4C 3D C7          JMP KEY_DEC
1082  1082   C440
1083  1083   C440 C9 18       CHK_FUNC4  CMP #18H
1084  1084   C442 D0 03                BNE CHK_FUNC5
1085  1085   C444 4C 56 C7           JMP KEY_PC
1086  1086   C447
1087  1087   C447 C9 1B       CHK_FUNC5  CMP #1BH
1088  1088   C449 D0 03                BNE CHK_FUNC6
```

```
1089  1089   C44B 4C 6A C7         JMP KEY_REG
1090  1090   C44E
1091  1091   C44E C9 12       CHK_FUNC6  CMP #12H
1092  1092   C450 D0 03                  BNE CHK_FUNC7
1093  1093   C452 4C BB C7         JMP KEY_GO
1094  1094   C455
1095  1095   C455 C9 1D       CHK_FUNC7  CMP #1DH
1096  1096   C457 D0 03                  BNE CHK_FUNC8
1097  1097   C459 4C 65 C5         JMP KEY_REL
1098  1098   C45C
1099  1099   C45C C9 1F       CHK_FUNC8  CMP #1FH
1100  1100   C45E D0 03                  BNE CHK_FUNC9
1101  1101   C460 4C 22 C5         JMP KEY_DOWNLOAD_HEX
1102  1102   C463
1103  1103   C463 C9 13       CHK_FUNC9 CMP #13H
1104  1104   C465 D0 03                 BNE CHK_FUNC10
1105  1105   C467 4C E6 C7        JMP KEY_STEP
1106  1106   C46A
1107  1107   C46A
1108  1108   C46A C9 16       CHK_FUNC10 CMP #16H
1109  1109   C46C D0 03                  BNE CHK_FUNC11
1110  1110   C46E 4C A1 C4         JMP KEY_INS
1111  1111   C471
1112  1112   C471 C9 17       CHK_FUNC11 CMP #17H
1113  1113   C473 D0 03                  BNE CHK_FUNC12
1114  1114   C475 4C E3 C4         JMP KEY_DEL
1115  1115   C478
1116  1116   C478
1117  1117   C478 C9 1E       CHK_FUNC12 CMP #1EH
1118  1118   C47A D0 03                  BNE CHK_FUNC13
1119  1119   C47C 4C 4C CA         JMP KEY_DUMP
1120  1120   C47F
1121  1121   C47F C9 1A       CHK_FUNC13 CMP #1AH
1122  1122   C481 D0 03                  BNE CHK_FUNC14
1123  1123   C483 4C 96 C4         JMP KEY_MUTE
1124  1124   C486
1125  1125   C486 C9 15       CHK_FUNC14 CMP #15H
1126  1126   C488 D0 03                  BNE CHK_FUNC15
1127  1127   C48A 4C 9D C4         JMP KEY_TEST
1128  1128   C48D
1129  1129   C48D C9 1C       CHK_FUNC15 CMP #1CH
1130  1130   C48F D0 03                  BNE CHK_FUNC16
1131  1131   C491 4C 04 C6         JMP KEY_COPY
1132  1132   C494             CHK_FUNC16
1133  1133   C494
1134  1134   C494
1135  1135   C494 60                    RTS
1136  1136   C495
1137  1137   C495             ;-----------------------------------------------------------
1138  1138   C495             NO_RESPONSE
1139  1139   C495 60                    RTS
1140  1140   C496
1141  1141   C496
1142  1142   C496
1143  1143   C496 A5 AE       KEY_MUTE   LDA MUTE
1144  1144   C498 49 01                  EOR #1
1145  1145   C49A 85 AE             STA MUTE
1146  1146   C49C 60                RTS
1147  1147   C49D
1148  1148   C49D 20 79 CB    KEY_TEST   JSR TEST_TICK
1149  1149   C4A0 60                    RTS
1150  1150   C4A1
1151  1151   C4A1             ;-----------------------------------------------------------
1152  1152   C4A1             ; insert byte to current display+1
```

```
1153  1153   C4A1             ; shift down 256 bytes.
1154  1154   C4A1
1155  1155   C4A1 A5 94       KEY_INS   LDA STATE
1156  1156   C4A3 C9 01                 CMP #1
1157  1157   C4A5 F0 08                 BEQ KEY_INS1
1158  1158   C4A7 C9 02           CMP #2
1159  1159   C4A9 F0 04           BEQ KEY_INS1
1160  1160   C4AB
1161  1161   C4AB 20 95 C4        JSR NO_RESPONSE
1162  1162   C4AE 60              RTS
1163  1163   C4AF
1164  1164   C4AF A5 96       KEY_INS1  LDA DISPLAY
1165  1165   C4B1 85 84                 STA HL
1166  1166   C4B3 A5 97           LDA DISPLAY+1
1167  1167   C4B5 85 85           STA HL+1
1168  1168   C4B7
1169  1169   C4B7 18              CLC
1170  1170   C4B8 A5 85           LDA HL+1
1171  1171   C4BA 69 01           ADC #1
1172  1172   C4BC 85 85           STA HL+1  ; HL = HL+100H
1173  1173   C4BE
1174  1174   C4BE A2 00           LDX #$0
1175  1175   C4C0
1176  1176   C4C0 A0 00       INSERT2    LDY #0
1177  1177   C4C2 B1 84           LDA (HL),Y
1178  1178   C4C4
1179  1179   C4C4 A0 01           LDY #1
1180  1180   C4C6 91 84           STA (HL),Y
1181  1181   C4C8
1182  1182   C4C8 20 E2 C0              JSR DEC_HL
1183  1183   C4CB
1184  1184   C4CB CA              DEX
1185  1185   C4CC D0 F2           BNE INSERT2
1186  1186   C4CE
1187  1187   C4CE 20 C6 C0        JSR INC_HL
1188  1188   C4D1
1189  1189   C4D1 A9 00                 LDA #0
1190  1190   C4D3 A0 00           LDY #0
1191  1191   C4D5 91 84           STA (HL),Y
1192  1192   C4D7
1193  1193   C4D7 A5 84           LDA HL
1194  1194   C4D9 85 96                 STA DISPLAY
1195  1195   C4DB A5 85           LDA HL+1
1196  1196   C4DD 85 97           STA DISPLAY+1
1197  1197   C4DF
1198  1198   C4DF 20 B1 C5        JSR STILL_DATA
1199  1199   C4E2
1200  1200   C4E2
1201  1201   C4E2 60              RTS
1202  1202   C4E3
1203  1203   C4E3             ;------------------- DELETE KEY ----------------------------
1204  1204   C4E3
1205  1205   C4E3 A5 94       KEY_DEL   LDA STATE
1206  1206   C4E5 C9 01                 CMP #1
1207  1207   C4E7 F0 08                 BEQ KEY_DEL1
1208  1208   C4E9 C9 02           CMP #2
1209  1209   C4EB F0 04           BEQ KEY_DEL1
1210  1210   C4ED
1211  1211   C4ED 20 95 C4        JSR NO_RESPONSE
1212  1212   C4F0 60              RTS
1213  1213   C4F1
1214  1214   C4F1 A5 96       KEY_DEL1  LDA DISPLAY
1215  1215   C4F3 85 84                 STA HL
1216  1216   C4F5 A5 97           LDA DISPLAY+1
```

```
1217  1217   C4F7 85 85          STA HL+1
1218  1218   C4F9
1219  1219   C4F9 A2 00                LDX #$0
1220  1220   C4FB
1221  1221   C4FB A0 01       DELETE2    LDY #1
1222  1222   C4FD B1 84          LDA (HL),Y
1223  1223   C4FF
1224  1224   C4FF A0 00          LDY #0
1225  1225   C501 91 84          STA (HL),Y
1226  1226   C503
1227  1227   C503 20 C6 C0             JSR INC_HL
1228  1228   C506
1229  1229   C506 CA             DEX
1230  1230   C507 D0 F2          BNE DELETE2
1231  1231   C509
1232  1232   C509 20 B1 C5       JSR STILL_DATA
1233  1233   C50C
1234  1234   C50C 60             RTS
1235  1235   C50D
1236  1236   C50D             ;-------------------------------------------------------
1237  1237   C50D             KEY_SEND_HEX
1238  1238   C50D A9 07                  LDA #7
1239  1239   C50F 85 94             STA STATE ; STATE = 7 FOR SENDING HEX FILE
1240  1240   C511
1241  1241   C511 A9 00                  LDA #0
1242  1242   C513 85 95             STA ZERO_FLAG
1243  1243   C515 20 81 C5          JSR STILL_ADDRESS
1244  1244   C518 A9 AE             LDA #0AEH
1245  1245   C51A 85 8C             STA BUFFER
1246  1246   C51C A9 02             LDA #2
1247  1247   C51E 85 8D             STA BUFFER+1
1248  1248   C520 60                RTS
1249  1249   C521
1250  1250   C521 60                 RTS
1251  1251   C522
1252  1252   C522
1253  1253   C522             ;-------------------------------------------------------
1254  1254   C522             KEY_DOWNLOAD_HEX
1255  1255   C522
1256  1256   C522 A9 01              LDA #1
1257  1257   C524 85 AE              STA MUTE
1258  1258   C526
1259  1259   C526 A9 B3              LDA #0B3H    ; PRINT LOAD
1260  1260   C528 85 91              STA BUFFER+5
1261  1261   C52A A9 85              LDA #85H
1262  1262   C52C 85 91              STA BUFFER+5
1263  1263   C52E A9 A3              LDA #0A3H
1264  1264   C530 85 90              STA BUFFER+4
1265  1265   C532 A9 3F              LDA #3FH
1266  1266   C534 85 8F              STA BUFFER+3
1267  1267   C536 A9 B3              LDA #0B3H
1268  1268   C538 85 8E              STA BUFFER+2
1269  1269   C53A A9 00              LDA #0
1270  1270   C53C 85 8D              STA BUFFER+1
1271  1271   C53E 85 8C              STA BUFFER
1272  1272   C540
1273  1273   C540                   ; JSR NEW_LINE
1274  1274   C540                   ; JSR NEW_LINE
1275  1275   C540                   ; JSR NEW_LINE
1276  1276   C540
1277  1277   C540 A2 1F                    LDX #TEXT2-TEXT1
1278  1278   C542 20 8F C0                 JSR PSTRING
1279  1279   C545                ;    JSR NEW_LINE
1280  1280   C545 20 AC C1                 JSR GET_RECORD  ; GET INTEL HEX FILE
```

```
1281  1281   C548
1282  1282   C548 A2 32              LDX #TEXT3-TEXT1
1283  1283   C54A 20 8F C0           JSR PSTRING
1284  1284   C54D
1285  1285   C54D
1286  1286   C54D A9 02              LDA #2
1287  1287   C54F 85 94              STA STATE
1288  1288   C551 20 B1 C5           JSR STILL_DATA
1289  1289   C554
1290  1290   C554 60                 RTS
1291  1291   C555
1292  1292   C555
1293  1293   C555                          ;LDA #10
1294  1294   C555                    ;STA STATE
1295  1295   C555                    ;RTS
1296  1296   C555
1297  1297   C555 A9 55       GO_STATE10    LDA #55H
1298  1298   C557 8D 00 80           STA GPIO1
1299  1299   C55A
1300  1300   C55A
1301  1301   C55A 20 AC C1           JSR GET_RECORD  ; GET INTEL HEX FILE
1302  1302   C55D
1303  1303   C55D A9 02              LDA #2
1304  1304   C55F 85 94              STA STATE
1305  1305   C561 20 B1 C5           JSR STILL_DATA
1306  1306   C564
1307  1307   C564 60                 RTS
1308  1308   C565
1309  1309   C565
1310  1310   C565             ;-------------------------------------------------------
1311  1311   C565 A9 05       KEY_REL      LDA #5
1312  1312   C567 85 94              STA STATE ; STATE = 5 FOR RELATIVE BYTE CALCULATION
1313  1313   C569
1314  1314   C569 A9 00                    LDA #0
1315  1315   C56B 85 95              STA ZERO_FLAG
1316  1316   C56D 20 81 C5           JSR STILL_ADDRESS
1317  1317   C570 A9 AE              LDA #0AEH
1318  1318   C572 85 8C              STA BUFFER
1319  1319   C574 A9 02              LDA #2
1320  1320   C576 85 8D              STA BUFFER+1
1321  1321   C578 60                 RTS
1322  1322   C579
1323  1323   C579
1324  1324   C579             ;-------------------------------------------------------
1325  1325   C579 A9 01       KEY_ADDR     LDA #1
1326  1326   C57B 85 94                    STA STATE     ; STATE =1 FOR ADDRESS MODE
1327  1327   C57D
1328  1328   C57D A9 00              LDA #0
1329  1329   C57F 85 95              STA ZERO_FLAG
1330  1330   C581
1331  1331   C581             STILL_ADDRESS
1332  1332   C581 20 D9 C5                 JSR READ_MEMORY
1333  1333   C584
1334  1334   C584 A5 91              LDA BUFFER+5
1335  1335   C586 09 40              ORA #40H
1336  1336   C588 85 91              STA BUFFER+5
1337  1337   C58A
1338  1338   C58A A5 90              LDA BUFFER+4
1339  1339   C58C 09 40              ORA #40H
1340  1340   C58E 85 90              STA BUFFER+4
1341  1341   C590
1342  1342   C590 A5 8F              LDA BUFFER+3
1343  1343   C592 09 40              ORA #40H
1344  1344   C594 85 8F              STA BUFFER+3
```

```
1345  1345   C596
1346  1346   C596 A5 8E                  LDA BUFFER+2
1347  1347   C598 09 40            ORA #40H
1348  1348   C59A 85 8E            STA BUFFER+2
1349  1349   C59C
1350  1350   C59C A5 8D                  LDA BUFFER+1
1351  1351   C59E 29 BF            AND #~40H
1352  1352   C5A0 85 8D            STA BUFFER+1
1353  1353   C5A2
1354  1354   C5A2 A5 8C            LDA BUFFER
1355  1355   C5A4 29 BF            AND #~40H
1356  1356   C5A6 85 8C            STA BUFFER
1357  1357   C5A8
1358  1358   C5A8 60                     RTS
1359  1359   C5A9             ;-----------------------------------------------------------
1360  1360   C5A9 A9 02       KEY_DATA     LDA #2
1361  1361   C5AB 85 94                   STA STATE     ; STATE =2 FOR DATA MODE
1362  1362   C5AD
1363  1363   C5AD A9 00              LDA #0
1364  1364   C5AF 85 95            STA ZERO_FLAG
1365  1365   C5B1
1366  1366   C5B1 20 D9 C5    STILL_DATA   JSR READ_MEMORY
1367  1367   C5B4
1368  1368   C5B4 A5 91            LDA BUFFER+5
1369  1369   C5B6 29 BF            AND #~40H
1370  1370   C5B8 85 91            STA BUFFER+5
1371  1371   C5BA
1372  1372   C5BA A5 90            LDA BUFFER+4
1373  1373   C5BC 29 BF            AND #~40H
1374  1374   C5BE 85 90            STA BUFFER+4
1375  1375   C5C0
1376  1376   C5C0 A5 8F            LDA BUFFER+3
1377  1377   C5C2 29 BF            AND #~40H
1378  1378   C5C4 85 8F            STA BUFFER+3
1379  1379   C5C6
1380  1380   C5C6 A5 8E                  LDA BUFFER+2
1381  1381   C5C8 29 BF            AND #~40H
1382  1382   C5CA 85 8E            STA BUFFER+2
1383  1383   C5CC
1384  1384   C5CC A5 8D                  LDA BUFFER+1
1385  1385   C5CE 09 40            ORA #40H
1386  1386   C5D0 85 8D            STA BUFFER+1
1387  1387   C5D2
1388  1388   C5D2 A5 8C            LDA BUFFER
1389  1389   C5D4 09 40            ORA #40H
1390  1390   C5D6 85 8C            STA BUFFER
1391  1391   C5D8
1392  1392   C5D8 60                     RTS
1393  1393   C5D9
1394  1394   C5D9
1395  1395   C5D9             ; READ MEMORY
1396  1396   C5D9
1397  1397   C5D9             READ_MEMORY
1398  1398   C5D9
1399  1399   C5D9 A5 96                LDA DISPLAY
1400  1400   C5DB 85 84          STA HL
1401  1401   C5DD A5 97          LDA DISPLAY+1
1402  1402   C5DF 85 85          STA HL+1
1403  1403   C5E1 20 AF C3       JSR ADDRESS_DISPLAY
1404  1404   C5E4 A0 00                LDY #0
1405  1405   C5E6 B1 84          LDA (HL),Y
1406  1406   C5E8
1407  1407   C5E8                ;STA GPIO1
1408  1408   C5E8
```

```
1409  1409   C5E8 20 A1 C3      JSR DATA_DISPLAY
1410  1410   C5EB 60            RTS
1411  1411   C5EC              ;-------------------------------------------------------------
1412  1412   C5EC
1413  1413   C5EC 20 54 C6     HEX_SEND_FILE JSR HEX_ADDR
1414  1414   C5EF A9 AE                      LDA #0AEH
1415  1415   C5F1 85 8C                STA BUFFER
1416  1416   C5F3 A9 02                LDA #2
1417  1417   C5F5 85 8D                STA BUFFER+1
1418  1418   C5F7 60                   RTS
1419  1419   C5F8
1420  1420   C5F8 20 54 C6     HEX_SEND_FILE2 JSR HEX_ADDR
1421  1421   C5FB A9 8F                      LDA #08FH
1422  1422   C5FD 85 8C                STA BUFFER
1423  1423   C5FF A9 02                LDA #2
1424  1424   C601 85 8D                STA BUFFER+1
1425  1425   C603 60                   RTS
1426  1426   C604
1427  1427   C604 A9 0A        KEY_COPY     LDA #10
1428  1428   C606 85 94              STA STATE ; STATE = 10 FOR COPY BLOCK OF MEMORY
1429  1429   C608
1430  1430   C608 A9 00                      LDA #0
1431  1431   C60A 85 95                STA ZERO_FLAG
1432  1432   C60C 20 81 C5             JSR STILL_ADDRESS
1433  1433   C60F A9 AE                LDA #0AEH
1434  1434   C611 85 8C                STA BUFFER
1435  1435   C613 A9 02                LDA #2
1436  1436   C615 85 8D                STA BUFFER+1
1437  1437   C617 60                   RTS
1438  1438   C618
1439  1439   C618
1440  1440   C618 20 54 C6     HEX_COPY10  JSR HEX_ADDR
1441  1441   C61B A9 AE                    LDA #0AEH
1442  1442   C61D 85 8C              STA BUFFER
1443  1443   C61F A9 02              LDA #2
1444  1444   C621 85 8D              STA BUFFER+1
1445  1445   C623 60                 RTS
1446  1446   C624
1447  1447   C624 20 54 C6     HEX_COPY11  JSR HEX_ADDR
1448  1448   C627 A9 8F                    LDA #08FH
1449  1449   C629 85 8C              STA BUFFER
1450  1450   C62B A9 02              LDA #2
1451  1451   C62D 85 8D              STA BUFFER+1
1452  1452   C62F 60                 RTS
1453  1453   C630
1454  1454   C630 20 54 C6     HEX_COPY12  JSR HEX_ADDR
1455  1455   C633 A9 B3                    LDA #0b3H
1456  1456   C635 85 8C              STA BUFFER
1457  1457   C637 A9 02              LDA #2
1458  1458   C639 85 8D              STA BUFFER+1
1459  1459   C63B 60                 RTS
1460  1460   C63C
1461  1461   C63C              ;-------------------------------------------------------------
1462  1462   C63C
1463  1463   C63C 20 54 C6     HEX_REL     JSR HEX_ADDR
1464  1464   C63F A9 AE                    LDA #0AEH
1465  1465   C641 85 8C              STA BUFFER
1466  1466   C643 A9 02              LDA #2
1467  1467   C645 85 8D              STA BUFFER+1
1468  1468   C647 60                 RTS
1469  1469   C648
1470  1470   C648
1471  1471   C648
1472  1472   C648 20 54 C6     HEX_REL6    JSR HEX_ADDR
```

```
1473  1473   C64B A9 B3                  LDA #0B3H
1474  1474   C64D 85 8C            STA BUFFER
1475  1475   C64F A9 02            LDA #2
1476  1476   C651 85 8D            STA BUFFER+1
1477  1477   C653 60               RTS
1478  1478   C654
1479  1479   C654
1480  1480   C654             ;------------------- HEX KEY FOR ADDRESS ------------------
1481  1481   C654
1482  1482   C654 A5 95       HEX_ADDR     LDA ZERO_FLAG
1483  1483   C656 C9 00              CMP #0
1484  1484   C658 D0 0A              BNE SHIFT_ADDRESS
1485  1485   C65A
1486  1486   C65A A9 01              LDA #1
1487  1487   C65C 85 95              STA ZERO_FLAG
1488  1488   C65E A9 00                    LDA #0
1489  1489   C660 85 96              STA DISPLAY
1490  1490   C662 85 97              STA DISPLAY+1
1491  1491   C664
1492  1492   C664 18          SHIFT_ADDRESS CLC
1493  1493   C665 26 96                    ROL DISPLAY
1494  1494   C667 26 97              ROL DISPLAY+1
1495  1495   C669
1496  1496   C669 18                       CLC
1497  1497   C66A 26 96                    ROL DISPLAY
1498  1498   C66C 26 97              ROL DISPLAY+1
1499  1499   C66E
1500  1500   C66E 18                 CLC
1501  1501   C66F 26 96                    ROL DISPLAY
1502  1502   C671 26 97              ROL DISPLAY+1
1503  1503   C673
1504  1504   C673 18                 CLC
1505  1505   C674 26 96                    ROL DISPLAY
1506  1506   C676 26 97              ROL DISPLAY+1
1507  1507   C678
1508  1508   C678 A5 96              LDA DISPLAY
1509  1509   C67A 05 83              ORA REG_C
1510  1510   C67C 85 96              STA DISPLAY
1511  1511   C67E
1512  1512   C67E                         ; JSR READ_MEMORY
1513  1513   C67E
1514  1514   C67E 20 81 C5           JSR STILL_ADDRESS
1515  1515   C681
1516  1516   C681 60                 RTS
1517  1517   C682             ;------------------------- HEX KEY FOR DATA MODE -----------
1518  1518   C682
1519  1519   C682
1520  1520   C682 A5 95       HEX_DATA     LDA ZERO_FLAG
1521  1521   C684 C9 00              CMP #0
1522  1522   C686 D0 0A              BNE SHIFT_DATA
1523  1523   C688
1524  1524   C688 A9 01              LDA #1
1525  1525   C68A 85 95              STA ZERO_FLAG
1526  1526   C68C
1527  1527   C68C A9 00              LDA #0
1528  1528   C68E A0 00              LDY #0
1529  1529   C690 91 96              STA (DISPLAY),Y
1530  1530   C692
1531  1531   C692 A0 00       SHIFT_DATA   LDY #0
1532  1532   C694 B1 96                    LDA (DISPLAY),Y
1533  1533   C696 0A                 ASL A
1534  1534   C697 0A                 ASL A
1535  1535   C698 0A                 ASL A
1536  1536   C699 0A                 ASL A
```

```
1537  1537   C69A 05 83              ORA REG_C
1538  1538   C69C 91 96              STA (DISPLAY),Y
1539  1539   C69E
1540  1540   C69E                   ; JSR READ_MEMORY
1541  1541   C69E 20 B1 C5           JSR STILL_DATA
1542  1542   C6A1 60                 RTS
1543  1543   C6A2
1544  1544   C6A2             ; INCREMENT CURRENT ADDRESS BY ONE
1545  1545   C6A2             ;
1546  1546   C6A2
1547  1547   C6A2 A5 94       KEY_INC     LDA STATE
1548  1548   C6A4 C9 05                   CMP #5
1549  1549   C6A6 F0 25            BEQ REL_KEY_PRESSED
1550  1550   C6A8
1551  1551   C6A8 C9 07            CMP #7
1552  1552   C6AA F0 3D            BEQ SEND_INC1
1553  1553   C6AC
1554  1554   C6AC C9 0A                  CMP #10
1555  1555   C6AE F0 55            BEQ COPY_START
1556  1556   C6B0
1557  1557   C6B0 C9 0B            CMP #11
1558  1558   C6B2 F0 6D            BEQ COPY_END
1559  1559   C6B4
1560  1560   C6B4             ; NORMAL INCREMENT
1561  1561   C6B4
1562  1562   C6B4 A9 02                  LDA #2
1563  1563   C6B6 85 94                  STA STATE     ; STATE =2 FOR DATA MODE
1564  1564   C6B8
1565  1565   C6B8 A9 00            LDA #0
1566  1566   C6BA 85 95            STA ZERO_FLAG
1567  1567   C6BC
1568  1568   C6BC
1569  1569   C6BC 18                    CLC
1570  1570   C6BD A5 96                 LDA DISPLAY
1571  1571   C6BF 69 01             ADC #1
1572  1572   C6C1 85 96             STA DISPLAY
1573  1573   C6C3 A5 97             LDA DISPLAY+1
1574  1574   C6C5 69 00             ADC #0
1575  1575   C6C7 85 97             STA DISPLAY+1
1576  1576   C6C9                        ; JSR READ_MEMORY
1577  1577   C6C9 20 B1 C5          JSR STILL_DATA
1578  1578   C6CC 60                RTS
1579  1579   C6CD
1580  1580   C6CD             REL_KEY_PRESSED
1581  1581   C6CD
1582  1582   C6CD                   ; Save start address
1583  1583   C6CD
1584  1584   C6CD A5 96             LDA DISPLAY
1585  1585   C6CF 85 A0             STA START_ADDRESS
1586  1586   C6D1 A5 97             LDA DISPLAY+1
1587  1587   C6D3 85 A1             STA START_ADDRESS+1
1588  1588   C6D5
1589  1589   C6D5 A9 06                  LDA #6
1590  1590   C6D7 85 94             STA STATE
1591  1591   C6D9 A9 00                  LDA #0
1592  1592   C6DB 85 95             STA ZERO_FLAG
1593  1593   C6DD
1594  1594   C6DD 20 81 C5          JSR STILL_ADDRESS
1595  1595   C6E0 A9 B3             LDA #0B3H
1596  1596   C6E2 85 8C             STA BUFFER
1597  1597   C6E4 A9 02             LDA #2
1598  1598   C6E6 85 8D             STA BUFFER+1
1599  1599   C6E8 60                RTS
1600  1600   C6E9
```

```
1601  1601   C6E9
1602  1602   C6E9              SEND_INC1   ; Save start address
1603  1603   C6E9
1604  1604   C6E9 A5 96              LDA DISPLAY
1605  1605   C6EB 85 A0              STA START_ADDRESS
1606  1606   C6ED A5 97              LDA DISPLAY+1
1607  1607   C6EF 85 A1              STA START_ADDRESS+1
1608  1608   C6F1
1609  1609   C6F1 A9 08                    LDA #8
1610  1610   C6F3 85 94              STA STATE
1611  1611   C6F5 A9 00                    LDA #0
1612  1612   C6F7 85 95              STA ZERO_FLAG
1613  1613   C6F9
1614  1614   C6F9 20 81 C5           JSR STILL_ADDRESS
1615  1615   C6FC A9 8F              LDA #08FH
1616  1616   C6FE 85 8C              STA BUFFER
1617  1617   C700 A9 02              LDA #2
1618  1618   C702 85 8D              STA BUFFER+1
1619  1619   C704 60                 RTS
1620  1620   C705
1621  1621   C705              COPY_START   ; Save start address
1622  1622   C705
1623  1623   C705 A5 96              LDA DISPLAY
1624  1624   C707 85 A0              STA START_ADDRESS
1625  1625   C709 A5 97              LDA DISPLAY+1
1626  1626   C70B 85 A1              STA START_ADDRESS+1
1627  1627   C70D
1628  1628   C70D A9 0B                    LDA #11
1629  1629   C70F 85 94              STA STATE
1630  1630   C711 A9 00                    LDA #0
1631  1631   C713 85 95              STA ZERO_FLAG
1632  1632   C715
1633  1633   C715 20 81 C5           JSR STILL_ADDRESS
1634  1634   C718
1635  1635   C718 A9 8F              LDA #08FH  ; show end address
1636  1636   C71A 85 8C              STA BUFFER
1637  1637   C71C A9 02              LDA #2
1638  1638   C71E 85 8D              STA BUFFER+1
1639  1639   C720 60                 RTS
1640  1640   C721
1641  1641   C721
1642  1642   C721              COPY_END   ; Save start address
1643  1643   C721
1644  1644   C721 A5 96              LDA DISPLAY
1645  1645   C723 85 A2              STA END_ADDRESS
1646  1646   C725 A5 97              LDA DISPLAY+1
1647  1647   C727 85 A3              STA END_ADDRESS+1
1648  1648   C729
1649  1649   C729 A9 0C                    LDA #12
1650  1650   C72B 85 94              STA STATE
1651  1651   C72D A9 00                    LDA #0
1652  1652   C72F 85 95              STA ZERO_FLAG
1653  1653   C731
1654  1654   C731 20 81 C5           JSR STILL_ADDRESS
1655  1655   C734
1656  1656   C734 A9 B3              LDA #0B3H   ; show destination
1657  1657   C736 85 8C              STA BUFFER
1658  1658   C738 A9 02              LDA #2
1659  1659   C73A 85 8D              STA BUFFER+1
1660  1660   C73C 60                 RTS
1661  1661   C73D
1662  1662   C73D
1663  1663   C73D
1664  1664   C73D
```

```
1665  1665   C73D
1666  1666   C73D
1667  1667   C73D                ; DECREMENT CURRENT ADDRESS BY ONE
1668  1668   C73D                ;
1669  1669   C73D
1670  1670   C73D A9 02          KEY_DEC     LDA #2
1671  1671   C73F 85 94                      STA STATE     ; STATE =2 FOR DATA MODE
1672  1672   C741
1673  1673   C741 A9 00               LDA #0
1674  1674   C743 85 95               STA ZERO_FLAG
1675  1675   C745
1676  1676   C745
1677  1677   C745 38                   SEC
1678  1678   C746 A5 96                LDA DISPLAY
1679  1679   C748 E9 01                SBC #1
1680  1680   C74A 85 96                STA DISPLAY
1681  1681   C74C A5 97                LDA DISPLAY+1
1682  1682   C74E E9 00                SBC #0
1683  1683   C750 85 97                STA DISPLAY+1
1684  1684   C752                          ;  JSR READ_MEMORY
1685  1685   C752 20 B1 C5             JSR STILL_DATA
1686  1686   C755 60                   RTS
1687  1687   C756
1688  1688   C756                ; KEY PC, SET CURRENT USER ADDRESS
1689  1689   C756
1690  1690   C756 A9 02          KEY_PC      LDA #2
1691  1691   C758 85 94                      STA STATE     ; STATE =2 FOR DATA MODE
1692  1692   C75A
1693  1693   C75A A9 00               LDA #0
1694  1694   C75C 85 95               STA ZERO_FLAG
1695  1695   C75E
1696  1696   C75E A5 98               LDA PC_USER
1697  1697   C760 85 96               STA DISPLAY
1698  1698   C762 A5 99               LDA PC_USER+1
1699  1699   C764 85 97               STA DISPLAY+1
1700  1700   C766                          ; JSR READ_MEMORY
1701  1701   C766 20 B1 C5             JSR STILL_DATA
1702  1702   C769 60                   RTS
1703  1703   C76A
1704  1704   C76A                ; KEY REGSITER
1705  1705   C76A                ; SET STATE TO 3 FOR REGISTER INPUT WITH HEX KEY
1706  1706   C76A
1707  1707   C76A A9 03          KEY_REG    LDA #3
1708  1708   C76C 85 94                     STA STATE  ; STATE = 3 FOR REGISTER DISPLAY
1709  1709   C76E
1710  1710   C76E A9 03               LDA #3
1711  1711   C770 85 91               STA BUFFER+5
1712  1712   C772 A9 8F               LDA #8FH
1713  1713   C774 85 90               STA BUFFER+4
1714  1714   C776 A9 BE               LDA #0BEH
1715  1715   C778 85 8F               STA BUFFER+3
1716  1716   C77A A9 02               LDA #2
1717  1717   C77C 85 8E               STA BUFFER+2
1718  1718   C77E A9 00               LDA #0
1719  1719   C780 85 8D               STA BUFFER+1
1720  1720   C782 85 8C               STA BUFFER
1721  1721   C784
1722  1722   C784 60                  RTS
1723  1723   C785
1724  1724   C785                ;-------------------------------------------------------
1725  1725   C785                GO_STATE8
1726  1726   C785
1727  1727   C785 A5 96            LDA DISPLAY
1728  1728   C787 85 A4            STA DESTINATION     ; DESTINATION IS NOW ENDING ADDRESS
```

```
1729  1729   C789 A5 97        LDA DISPLAY+1
1730  1730   C78B 85 A5        STA DESTINATION+1
1731  1731   C78D
1732  1732   C78D            ; NOW COMPUTE NUMBER OF BYTE = DESTINATION - START_ADDRESS
1733  1733   C78D A5 A0              LDA START_ADDRESS
1734  1734   C78F 85 84        STA HL
1735  1735   C791 A5 A1        LDA START_ADDRESS+1
1736  1736   C793 85 85        STA HL+1
1737  1737   C795
1738  1738   C795 38                 SEC
1739  1739   C796 A5 A4        LDA DESTINATION
1740  1740   C798 E5 84        SBC HL
1741  1741   C79A 85 A6        STA OFFSET_BYTE
1742  1742   C79C
1743  1743   C79C A5 A5        LDA DESTINATION+1
1744  1744   C79E E5 85        SBC HL+1
1745  1745   C7A0 85 A7        STA OFFSET_BYTE+1   ; OFFSET = NUMBER OF BYTE
1746  1746   C7A2
1747  1747   C7A2            ; DIVIDE NUMBER OF BYTE WITH 16 TO GET NUMBER OF RECORD TO B
1748  1748   C7A2
1749  1749   C7A2 46 A7              LSR OFFSET_BYTE+1
1750  1750   C7A4 66 A6        ROR OFFSET_BYTE
1751  1751   C7A6
1752  1752   C7A6 46 A7        LSR OFFSET_BYTE+1
1753  1753   C7A8 66 A6        ROR OFFSET_BYTE
1754  1754   C7AA
1755  1755   C7AA 46 A7              LSR OFFSET_BYTE+1
1756  1756   C7AC 66 A6        ROR OFFSET_BYTE
1757  1757   C7AE
1758  1758   C7AE 46 A7        LSR OFFSET_BYTE+1
1759  1759   C7B0 66 A6        ROR OFFSET_BYTE
1760  1760   C7B2
1761  1761   C7B2 A5 A6              LDA OFFSET_BYTE ; CHECK RESULT
1762  1762   C7B4 8D 00 80     STA GPIO1
1763  1763   C7B7
1764  1764   C7B7 60           RTS
1765  1765   C7B8
1766  1766   C7B8            SHORT_GO_STATE10
1767  1767   C7B8 4C 55 C5           JMP GO_STATE10
1768  1768   C7BB
1769  1769   C7BB            ; KEY GO WRITE USER REGISTERS TO STACK AND USE RTI TO JUMP T
1770  1770   C7BB            ;
1771  1771   C7BB
1772  1772   C7BB A5 94      KEY_GO LDA STATE
1773  1773   C7BD C9 06             CMP #6
1774  1774   C7BF F0 4D             BEQ GO_STATE6
1775  1775   C7C1
1776  1776   C7C1 C9 08             CMP #8
1777  1777   C7C3 F0 C0             BEQ GO_STATE8
1778  1778   C7C5
1779  1779   C7C5 C9 0A             CMP #10
1780  1780   C7C7 F0 EF             BEQ SHORT_GO_STATE10
1781  1781   C7C9
1782  1782   C7C9 C9 0C             CMP #12
1783  1783   C7CB F0 16             BEQ GO_COPY12
1784  1784   C7CD
1785  1785   C7CD            ; JUMP TO USER CODE
1786  1786   C7CD
1787  1787   C7CD BA                 TSX
1788  1788   C7CE 86 9F        STX SAVE_SP   ; SAVE SYSTEM STACK
1789  1789   C7D0
1790  1790   C7D0            ; NOW SWITCH TO USER STACK
1791  1791   C7D0
1792  1792   C7D0 A6 9D              LDX USER_S
```

```
1793  1793   C7D2 9A           TXS
1794  1794   C7D3
1795  1795   C7D3 A5 97        LDA DISPLAY+1
1796  1796   C7D5 48           PHA
1797  1797   C7D6 A5 96        LDA DISPLAY
1798  1798   C7D8 48           PHA
1799  1799   C7D9 A5 9E        LDA USER_P
1800  1800   C7DB 48           PHA
1801  1801   C7DC A6 9B        LDX USER_X
1802  1802   C7DE A4 9C        LDY USER_Y
1803  1803   C7E0 A5 9A        LDA USER_A
1804  1804   C7E2 40           RTI
1805  1805   C7E3
1806  1806   C7E3 4C 80 C8    GO_COPY12 JMP GO_COPY13
1807  1807   C7E6
1808  1808   C7E6             ;---------------------- SINGLE STEP -----------------------
1809  1809   C7E6
1810  1810   C7E6             KEY_STEP
1811  1811   C7E6
1812  1812   C7E6 BA                  TSX
1813  1813   C7E7 86 9F        STX SAVE_SP   ; SAVE SYSTEM STACK
1814  1814   C7E9
1815  1815   C7E9                       ; NOW SWITCH TO USER STACK
1816  1816   C7E9
1817  1817   C7E9 A6 9D               LDX USER_S
1818  1818   C7EB 9A           TXS
1819  1819   C7EC
1820  1820   C7EC             ; LOAD CURRENT PC TO DISPLAY
1821  1821   C7EC
1822  1822   C7EC A5 98               LDA PC_USER
1823  1823   C7EE 85 96        STA DISPLAY
1824  1824   C7F0 A5 99        LDA PC_USER+1
1825  1825   C7F2 85 97        STA DISPLAY+1
1826  1826   C7F4
1827  1827   C7F4
1828  1828   C7F4
1829  1829   C7F4 A5 97               LDA DISPLAY+1
1830  1830   C7F6 48           PHA
1831  1831   C7F7 A5 96        LDA DISPLAY
1832  1832   C7F9 48           PHA
1833  1833   C7FA A5 9E        LDA USER_P
1834  1834   C7FC 48           PHA
1835  1835   C7FD A6 9B        LDX USER_X
1836  1836   C7FF A4 9C        LDY USER_Y
1837  1837   C801
1838  1838   C801 A9 FF        LDA #$FF       ; BREAK MUST BE LOGIC HIGH TO ENABLE IT
1839  1839   C803 8D 02 80     STA PORT1
1840  1840   C806
1841  1841   C806 EA                  NOP
1842  1842   C807 EA           NOP
1843  1843   C808 EA           NOP
1844  1844   C809 EA           NOP
1845  1845   C80A EA           NOP
1846  1846   C80B A5 9A        LDA USER_A  ;
1847  1847   C80D 40           RTI         ;
1848  1848   C80E
1849  1849   C80E             ; USER INSTRUCTION IS 8TH FETCHING, IT WILL JUMP TO NMI SERV
1850  1850   C80E
1851  1851   C80E
1852  1852   C80E
1853  1853   C80E             ; KEY GO WITH RELATIVE CALCULATION
1854  1854   C80E             ; FIND OFFSET BYTE
1855  1855   C80E
1856  1856   C80E             GO_STATE6
```

```
1857  1857   C80E
1858  1858   C80E A5 96              LDA DISPLAY
1859  1859   C810 85 A4          STA DESTINATION
1860  1860   C812 A5 97          LDA DISPLAY+1
1861  1861   C814 85 A5          STA DESTINATION+1
1862  1862   C816
1863  1863   C816              ; NOW COMPUTE OFFSET_BYTE = DESTINATION - START_ADDRESS
1864  1864   C816
1865  1865   C816              ; THE REAL PC WILL BE NEXT INTSRUCTION ADDRESS (+2 FROM BRAN
1866  1866   C816
1867  1867   C816 A5 A0              LDA START_ADDRESS
1868  1868   C818 85 84          STA HL
1869  1869   C81A A5 A1          LDA START_ADDRESS+1
1870  1870   C81C 85 85          STA HL+1
1871  1871   C81E 20 C6 C0       JSR INC_HL
1872  1872   C821 20 C6 C0       JSR INC_HL
1873  1873   C824
1874  1874   C824 38                 SEC
1875  1875   C825 A5 A4          LDA DESTINATION
1876  1876   C827 E5 84          SBC HL
1877  1877   C829 85 A6          STA OFFSET_BYTE
1878  1878   C82B
1879  1879   C82B A5 A5          LDA DESTINATION+1
1880  1880   C82D E5 85          SBC HL+1
1881  1881   C82F 85 A7          STA OFFSET_BYTE+1
1882  1882   C831
1883  1883   C831              ; CHECK IF THE OFFSET BYTE WAS BETWEEN -128 (FF80) TO +127 (
1884  1884   C831              ; IF BIT 7 OF THE OFFSET BYTE IS 0, THE HIGH BYTE MUST BE ZE
1885  1885   C831              ; IF BIT 7 OF THE OFFSET BYTE IS 1, THE HIGH BYTE MUST BE FF
1886  1886   C831              ; OTHERWISE, THE OFFSET BYTE WAS OUT OF RANGE, SHOW ERROR TH
1887  1887   C831
1888  1888   C831 A5 A6              LDA OFFSET_BYTE
1889  1889   C833 29 80          AND #80H
1890  1890   C835 F0 09          BEQ CHK_OFFSET_HIGH
1891  1891   C837
1892  1892   C837              ; CHECK HIGH BYTE MUST BE FF (-1)
1893  1893   C837
1894  1894   C837 A5 A7          LDA OFFSET_BYTE+1
1895  1895   C839 C9 FF          CMP #0FFH
1896  1896   C83B D0 28          BNE OUT_OFF_RANGE
1897  1897   C83D
1898  1898   C83D 4C 44 C8       JMP IN_RANGE
1899  1899   C840
1900  1900   C840              CHK_OFFSET_HIGH
1901  1901   C840 A5 A7              LDA OFFSET_BYTE+1
1902  1902   C842 D0 21          BNE OUT_OFF_RANGE
1903  1903   C844
1904  1904   C844              ; STORE OFFSET TO THE 2ND BYTE OF BRANCH INSTRUCTION
1905  1905   C844
1906  1906   C844 A5 A0        IN_RANGE LDA START_ADDRESS
1907  1907   C846 85 84          STA HL
1908  1908   C848 A5 A1          LDA START_ADDRESS+1
1909  1909   C84A 85 85          STA HL+1
1910  1910   C84C 20 C6 C0       JSR INC_HL
1911  1911   C84F
1912  1912   C84F A5 A6          LDA OFFSET_BYTE
1913  1913   C851 A0 00          LDY #0
1914  1914   C853 91 84          STA (HL),Y
1915  1915   C855
1916  1916   C855 A5 84          LDA HL          ; DISPLAY LOCATION OF OFFSET BYTE
1917  1917   C857 85 96          STA DISPLAY
1918  1918   C859 A5 85          LDA HL+1
1919  1919   C85B 85 97          STA DISPLAY+1
1920  1920   C85D
```

```
1921  1921   C85D
1922  1922   C85D 20 B1 C5      JSR STILL_DATA
1923  1923   C860
1924  1924   C860 A9 02         LDA #2
1925  1925   C862 85 94         STA STATE
1926  1926   C864 60            RTS
1927  1927   C865
1928  1928   C865              OUT_OFF_RANGE
1929  1929   C865
1930  1930   C865 A9 02                LDA #2
1931  1931   C867 85 91         STA BUFFER+5
1932  1932   C869 A9 8F         LDA #8FH
1933  1933   C86B 85 90         STA BUFFER+4
1934  1934   C86D A9 03         LDA #3
1935  1935   C86F 85 8F         STA BUFFER+3
1936  1936   C871 A9 03         LDA #3
1937  1937   C873 85 8E         STA BUFFER+2
1938  1938   C875 A9 00         LDA #0
1939  1939   C877 85 8D         STA BUFFER+1
1940  1940   C879 85 8C         STA BUFFER
1941  1941   C87B
1942  1942   C87B A9 02         LDA #2
1943  1943   C87D 85 94         STA STATE
1944  1944   C87F
1945  1945   C87F 60            RTS
1946  1946   C880
1947  1947   C880
1948  1948   C880 A5 96        GO_COPY13    LDA DISPLAY
1949  1949   C882 85 A4               STA DESTINATION
1950  1950   C884 A5 97               LDA DISPLAY+1
1951  1951   C886 85 A5               STA DESTINATION+1
1952  1952   C888
1953  1953   C888              ; NOW COPY BLOCK OF MEMORY
1954  1954   C888
1955  1955   C888 A5 A0               LDA START_ADDRESS
1956  1956   C88A 85 84               STA HL
1957  1957   C88C A5 A1               LDA START_ADDRESS+1
1958  1958   C88E 85 85               STA HL+1
1959  1959   C890
1960  1960   C890 A5 A4               LDA DESTINATION
1961  1961   C892 85 86               STA DE
1962  1962   C894 A5 A5               LDA DESTINATION+1
1963  1963   C896 85 87               STA DE+1
1964  1964   C898
1965  1965   C898              COPY_MEMORY1
1966  1966   C898 A0 00               LDY #0
1967  1967   C89A B1 84               LDA (HL),Y
1968  1968   C89C 91 86               STA (DE),Y
1969  1969   C89E 20 C6 C0            JSR INC_HL
1970  1970   C8A1 20 D4 C0            JSR INC_DE
1971  1971   C8A4
1972  1972   C8A4 A5 84               LDA HL
1973  1973   C8A6 C5 A2               CMP END_ADDRESS
1974  1974   C8A8 D0 EE               BNE COPY_MEMORY1
1975  1975   C8AA A5 85               LDA HL+1
1976  1976   C8AC C5 A3               CMP END_ADDRESS+1
1977  1977   C8AE D0 E8               BNE COPY_MEMORY1
1978  1978   C8B0
1979  1979   C8B0 A5 A4               LDA DESTINATION
1980  1980   C8B2 85 96               STA DISPLAY
1981  1981   C8B4 A5 A5               LDA DESTINATION+1
1982  1982   C8B6 85 97               STA DISPLAY+1
1983  1983   C8B8 20 B1 C5            JSR STILL_DATA
1984  1984   C8BB
```

```
1985  1985   C8BB 60                RTS
1986  1986   C8BC
1987  1987   C8BC
1988  1988   C8BC
1989  1989   C8BC
1990  1990   C8BC
1991  1991   C8BC             ; NMI SERVICE ROUTINE
1992  1992   C8BC             ; SAVE CPU REGISTERS TO USER REGISTERS FOR PROGRAM DEBUGGING
1993  1993   C8BC
1994  1994   C8BC             NMI_SERVICE
1995  1995   C8BC
1996  1996   C8BC 85 9A                STA USER_A
1997  1997   C8BE                     ; STA GPIO1     ; 8-BIT DISPLAY WILL SHOW CONTENT OF
1998  1998   C8BE
1999  1999   C8BE A9 BF          LDA #$BF
2000  2000   C8C0 8D 02 80      STA PORT1      ; TURN OFF BRK SIGNAL
2001  2001   C8C3
2002  2002   C8C3             ; STILL WITH USER STACK
2003  2003   C8C3
2004  2004   C8C3 68            PLA
2005  2005   C8C4 85 9E         STA USER_P
2006  2006   C8C6
2007  2007   C8C6 68            PLA
2008  2008   C8C7 85 96         STA DISPLAY
2009  2009   C8C9 85 98         STA PC_USER
2010  2010   C8CB 68            PLA
2011  2011   C8CC 85 97         STA DISPLAY+1
2012  2012   C8CE 85 99         STA PC_USER+1
2013  2013   C8D0 84 9C         STY USER_Y
2014  2014   C8D2 86 9B         STX USER_X
2015  2015   C8D4
2016  2016   C8D4 BA            TSX
2017  2017   C8D5 86 9D         STX USER_S
2018  2018   C8D7
2019  2019   C8D7 20 79 C5      JSR KEY_ADDR ; DISPLAY LOCATION THAT BREAKED
2020  2020   C8DA
2021  2021   C8DA             ; RESTORE SYSTEM STACK
2022  2022   C8DA
2023  2023   C8DA A6 9F         LDX SAVE_SP
2024  2024   C8DC 9A            TXS
2025  2025   C8DD
2026  2026   C8DD 60            RTS
2027  2027   C8DE
2028  2028   C8DE             ; DISPLAY USER REGSITERS
2029  2029   C8DE
2030  2030   C8DE A5 83       HEX_REG  LDA REG_C
2031  2031   C8E0 C9 00               CMP #0
2032  2032   C8E2 D0 14         BNE CHK_REG1
2033  2033   C8E4
2034  2034   C8E4 A5 9A         LDA USER_A
2035  2035   C8E6              ; STA GPIO1
2036  2036   C8E6 20 A1 C3      JSR DATA_DISPLAY
2037  2037   C8E9 A9 82         LDA #82H
2038  2038   C8EB 85 8E         STA BUFFER+2
2039  2039   C8ED A9 3F         LDA #3FH        ; REGISTER A
2040  2040   C8EF 85 8F         STA BUFFER+3
2041  2041   C8F1 A9 00         LDA #0
2042  2042   C8F3 85 90         STA BUFFER+4
2043  2043   C8F5 85 91         STA BUFFER+5
2044  2044   C8F7 60            RTS
2045  2045   C8F8
2046  2046   C8F8             CHK_REG1
2047  2047   C8F8 C9 01               CMP #1
2048  2048   C8FA D0 14         BNE CHK_REG2
```

```
2049   2049   C8FC
2050   2050   C8FC A5 9B          LDA USER_X
2051   2051   C8FE               ; STA GPIO1
2052   2052   C8FE
2053   2053   C8FE 20 A1 C3       JSR DATA_DISPLAY
2054   2054   C901 A9 82          LDA #82H
2055   2055   C903 85 8E          STA BUFFER+2
2056   2056   C905 A9 07          LDA #7          ; REGISTER X
2057   2057   C907 85 8F          STA BUFFER+3
2058   2058   C909 A9 00          LDA #0
2059   2059   C90B 85 90          STA BUFFER+4
2060   2060   C90D 85 91          STA BUFFER+5
2061   2061   C90F 60             RTS
2062   2062   C910
2063   2063   C910 C9 02       CHK_REG2  CMP #2
2064   2064   C912 D0 14          BNE CHK_REG3
2065   2065   C914
2066   2066   C914 A5 9C          LDA USER_Y
2067   2067   C916               ; STA GPIO1
2068   2068   C916
2069   2069   C916 20 A1 C3       JSR DATA_DISPLAY
2070   2070   C919 A9 82          LDA #82H
2071   2071   C91B 85 8E          STA BUFFER+2
2072   2072   C91D A9 B6          LDA #0B6H         ; REGISTER Y
2073   2073   C91F 85 8F          STA BUFFER+3
2074   2074   C921 A9 00          LDA #0
2075   2075   C923 85 90          STA BUFFER+4
2076   2076   C925 85 91          STA BUFFER+5
2077   2077   C927 60             RTS
2078   2078   C928
2079   2079   C928
2080   2080   C928 C9 03       CHK_REG3 CMP #3
2081   2081   C92A D0 14          BNE CHK_REG4
2082   2082   C92C
2083   2083   C92C A5 9D          LDA USER_S
2084   2084   C92E               ; STA GPIO1
2085   2085   C92E
2086   2086   C92E 20 A1 C3       JSR DATA_DISPLAY
2087   2087   C931 A9 82          LDA #82H
2088   2088   C933 85 8E          STA BUFFER+2
2089   2089   C935 A9 AE          LDA #0AEH         ; REGISTER S
2090   2090   C937 85 8F          STA BUFFER+3
2091   2091   C939 A9 00          LDA #0
2092   2092   C93B 85 90          STA BUFFER+4
2093   2093   C93D 85 91          STA BUFFER+5
2094   2094   C93F 60             RTS
2095   2095   C940
2096   2096   C940 C9 05       CHK_REG4 CMP #5
2097   2097   C942 D0 42                BNE CHK_REG5
2098   2098   C944
2099   2099   C944 A9 00           LDA #0            ; RESET HL TO 0000
2100   2100   C946 85 84           STA HL
2101   2101   C948 85 85           STA HL+1
2102   2102   C94A
2103   2103   C94A A5 9E           LDA USER_P
2104   2104   C94C                ; STA GPIO1
2105   2105   C94C 29 01           AND #1
2106   2106   C94E F0 06           BEQ NEXT_BIT1
2107   2107   C950 A5 84           LDA HL
2108   2108   C952 09 01           ORA #1
2109   2109   C954 85 84           STA HL
2110   2110   C956
2111   2111   C956 A5 9E       NEXT_BIT1 LDA USER_P
2112   2112   C958 29 02                 AND #2
```

```
2113   2113   C95A F0 06          BEQ NEXT_BIT2
2114   2114   C95C
2115   2115   C95C A5 84          LDA HL
2116   2116   C95E 09 10          ORA #10H
2117   2117   C960 85 84          STA HL
2118   2118   C962
2119   2119   C962 A5 9E       NEXT_BIT2 LDA USER_P
2120   2120   C964 29 04                 AND #4
2121   2121   C966 F0 06          BEQ NEXT_BIT3
2122   2122   C968
2123   2123   C968 A5 85          LDA HL+1
2124   2124   C96A 09 01          ORA #1
2125   2125   C96C 85 85          STA HL+1
2126   2126   C96E
2127   2127   C96E A5 9E       NEXT_BIT3 LDA USER_P
2128   2128   C970 29 08                 AND #8
2129   2129   C972 F0 06          BEQ OK1
2130   2130   C974
2131   2131   C974 A5 85          LDA HL+1
2132   2132   C976 09 10          ORA #10H
2133   2133   C978 85 85          STA HL+1
2134   2134   C97A 20 AF C3    OK1       JSR ADDRESS_DISPLAY
2135   2135   C97D
2136   2136   C97D A9 1F          LDA #1FH
2137   2137   C97F 85 8D          STA BUFFER+1
2138   2138   C981 A9 85          LDA #085H
2139   2139   C983 85 8C          STA BUFFER
2140   2140   C985 60                   RTS
2141   2141   C986
2142   2142   C986
2143   2143   C986 C9 04       CHK_REG5  CMP #4
2144   2144   C988 D0 42                 BNE CHK_REG6
2145   2145   C98A
2146   2146   C98A A9 00                 LDA #0            ; RESET HL TO 0000
2147   2147   C98C 85 84          STA HL
2148   2148   C98E 85 85          STA HL+1
2149   2149   C990
2150   2150   C990 A5 9E          LDA USER_P
2151   2151   C992              ;  STA GPIO1
2152   2152   C992
2153   2153   C992 29 10          AND #10H
2154   2154   C994 F0 06          BEQ NEXT_BIT4
2155   2155   C996 A5 84          LDA HL
2156   2156   C998 09 01          ORA #1
2157   2157   C99A 85 84          STA HL
2158   2158   C99C
2159   2159   C99C A5 9E       NEXT_BIT4 LDA USER_P
2160   2160   C99E 29 20                 AND #20H
2161   2161   C9A0 F0 06          BEQ NEXT_BIT5
2162   2162   C9A2 A5 84          LDA HL
2163   2163   C9A4 09 10          ORA #10H
2164   2164   C9A6 85 84          STA HL
2165   2165   C9A8
2166   2166   C9A8 A5 9E       NEXT_BIT5 LDA USER_P
2167   2167   C9AA 29 40                 AND #40H
2168   2168   C9AC F0 06          BEQ NEXT_BIT6
2169   2169   C9AE
2170   2170   C9AE A5 85          LDA HL+1
2171   2171   C9B0 09 01          ORA #1
2172   2172   C9B2 85 85          STA HL+1
2173   2173   C9B4
2174   2174   C9B4 A5 9E       NEXT_BIT6 LDA USER_P
2175   2175   C9B6 29 80                 AND #80H
2176   2176   C9B8 F0 06          BEQ OK2
```

```
2177  2177   C9BA
2178  2178   C9BA A5 85          LDA HL+1
2179  2179   C9BC 09 10          ORA #10H
2180  2180   C9BE 85 85          STA HL+1
2181  2181   C9C0
2182  2182   C9C0 20 AF C3    OK2       JSR ADDRESS_DISPLAY
2183  2183   C9C3
2184  2184   C9C3 A9 1F          LDA #1FH
2185  2185   C9C5 85 8D          STA BUFFER+1
2186  2186   C9C7 A9 37          LDA #37H
2187  2187   C9C9 85 8C          STA BUFFER
2188  2188   C9CB 60                   RTS
2189  2189   C9CC
2190  2190   C9CC C9 10       CHK_REG6  CMP #10H
2191  2191   C9CE B0 1C                 BCS NOT_HEX
2192  2192   C9D0
2193  2193   C9D0             ; NOW DISPLAY PAGE ZERO BYTE FROM 0 TO 9
2194  2194   C9D0
2195  2195   C9D0 38                    SEC
2196  2196   C9D1 E9 06          SBC #6
2197  2197   C9D3
2198  2198   C9D3             ; NOW A IS LOCATION IS PAGE ZERO 0-9
2199  2199   C9D3 AA                    TAX
2200  2200   C9D4 B5 00          LDA 0,X
2201  2201   C9D6 86 AA          STX SAVE_X
2202  2202   C9D8
2203  2203   C9D8 20 A1 C3       JSR DATA_DISPLAY
2204  2204   C9DB
2205  2205   C9DB A6 AA          LDX SAVE_X
2206  2206   C9DD
2207  2207   C9DD 8A             TXA
2208  2208   C9DE 85 85          STA HL+1
2209  2209   C9E0 20 AF C3       JSR ADDRESS_DISPLAY
2210  2210   C9E3
2211  2211   C9E3 A9 82          LDA #82H
2212  2212   C9E5 85 8F          STA BUFFER+3
2213  2213   C9E7 A9 00          LDA #0
2214  2214   C9E9 85 8E          STA BUFFER+2
2215  2215   C9EB 60             RTS
2216  2216   C9EC
2217  2217   C9EC             NOT_HEX
2218  2218   C9EC 60                    RTS
2219  2219   C9ED
2220  2220   C9ED
2221  2221   C9ED A5 AE       BEEP3    LDA MUTE
2222  2222   C9EF                     ; STA GPIO1
2223  2223   C9EF C9 00                CMP #0
2224  2224   C9F1 F0 01          BEQ BEEP_ON
2225  2225   C9F3 60             RTS
2226  2226   C9F4
2227  2227   C9F4 20 F8 C9    BEEP_ON  JSR BEEP
2228  2228   C9F7 60                   RTS
2229  2229   C9F8
2230  2230   C9F8
2231  2231   C9F8             ; PRODUCE BEEP WHEN KEY PRESSED
2232  2232   C9F8             ; CALIBRATED TO 523Hz
2233  2233   C9F8
2234  2234   C9F8 AD 01 80    BEEP     LDA PORT0
2235  2235   C9FB 29 40                AND #40H
2236  2236   C9FD F0 15          BEQ NO_BEEP    ; CHECK IF REPEAT KEY IS PRESSED, THEN NO
2237  2237   C9FF
2238  2238   C9FF A2 25                LDX #25H
2239  2239   CA01
2240  2240   CA01 A9 3F       BEEP2    LDA #3FH
```

```
2241  2241  CA03 8D 02 80       STA PORT1
2242  2242  CA06 20 15 CA       JSR BEEP_DELAY
2243  2243  CA09 A9 BF          LDA #0BFH
2244  2244  CA0B 8D 02 80       STA PORT1
2245  2245  CA0E 20 15 CA       JSR BEEP_DELAY
2246  2246  CA11
2247  2247  CA11 CA             DEX
2248  2248  CA12 D0 ED          BNE BEEP2
2249  2249  CA14
2250  2250  CA14 60          NO_BEEP   RTS
2251  2251  CA15
2252  2252  CA15             ;BEEP_DELAY LDY #$B0       ; 0BBH      ; adjust for the tone
2253  2253  CA15 A0 90       BEEP_DELAY LDY #$90       ; RAM testing
2254  2254  CA17
2255  2255  CA17 88          BEEP_LOOP DEY
2256  2256  CA18 D0 FD                 BNE BEEP_LOOP
2257  2257  CA1A 60             RTS
2258  2258  CA1B
2259  2259  CA1B             ; DISPLAY COLD BOOT MESSAGE
2260  2260  CA1B             ;
2261  2261  CA1B
2262  2262  CA1B             COLD_MESSAGE
2263  2263  CA1B
2264  2264  CA1B A9 02          LDA #2
2265  2265  CA1D 85 81                STA REG_D
2266  2266  CA1F
2267  2267  CA1F A9 08          LDA #8
2268  2268  CA21 85 82          STA REG_B
2269  2269  CA23
2270  2270  CA23 A2 07          LDX #7
2271  2271  CA25
2272  2272  CA25             DISPLAY2
2273  2273  CA25 20 B3 C2       JSR SCAN2
2274  2274  CA28
2275  2275  CA28 C6 81          DEC REG_D
2276  2276  CA2A D0 F9          BNE DISPLAY2
2277  2277  CA2C
2278  2278  CA2C CA             DEX
2279  2279  CA2D
2280  2280  CA2D C6 82          DEC REG_B
2281  2281  CA2F D0 F4          BNE DISPLAY2
2282  2282  CA31 60             RTS
2283  2283  CA32
2284  2284  CA32
2285  2285  CA32
2286  2286  CA32
2287  2287  CA32             ; NMI and IRQ are called via RAM-vector. This enables the pr
2288  2288  CA32             ; to insert his own routines.
2289  2289  CA32
2290  2290  CA32
2291  2291  CA32 6C FA 00    NMI    JMP     ($FA)
2292  2292  CA35 6C FE 00    IRQ    JMP     ($FE)
2293  2293  CA38
2294  2294  CA38             ;************************** 2021 modification **************
2295  2295  CA38
2296  2296  CA38 A2 00       COLD_TERMINAL LDX #0
2297  2297  CA3A 20 8F C0               JSR PSTRING
2298  2298  CA3D 20 9E C0       JSR NEW_LINE
2299  2299  CA40 60             RTS
2300  2300  CA41
2301  2301  CA41 A2 32       CLR_SCREEN LDX #50
2302  2302  CA43
2303  2303  CA43 A9 0A       CLR_LOOP   LDA #10
2304  2304  CA45 20 1F C0         JSR SEND_BYTE
```

```
2305  2305   CA48
2306  2306   CA48 CA              DEX
2307  2307   CA49 D0 F8           BNE CLR_LOOP
2308  2308   CA4B 60              RTS
2309  2309   CA4C
2310  2310   CA4C A9 01       KEY_DUMP   LDA #1
2311  2311   CA4E 85 AE                  STA MUTE   ; TURN OFF BEEP
2312  2312   CA50
2313  2313   CA50 A0 08                  LDY #8
2314  2314   CA52
2315  2315   CA52 84 AB       DUMP1      STY SAVE_Y
2316  2316   CA54
2317  2317   CA54 A5 96                  LDA DISPLAY
2318  2318   CA56 85 84           STA HL
2319  2319   CA58 A5 97           LDA DISPLAY+1
2320  2320   CA5A 85 85           STA HL+1
2321  2321   CA5C
2322  2322   CA5C 20 FE C0               JSR PRINT_LINE
2323  2323   CA5F
2324  2324   CA5F A5 96           LDA DISPLAY
2325  2325   CA61 85 84           STA HL
2326  2326   CA63 A5 97           LDA DISPLAY+1
2327  2327   CA65 85 85           STA HL+1
2328  2328   CA67
2329  2329   CA67 20 28 C1        JSR PRINT_ASCII
2330  2330   CA6A
2331  2331   CA6A A5 84           LDA HL
2332  2332   CA6C 85 96           STA DISPLAY
2333  2333   CA6E A5 85           LDA HL+1
2334  2334   CA70 85 97           STA DISPLAY+1
2335  2335   CA72
2336  2336   CA72 A4 AB           LDY SAVE_Y
2337  2337   CA74
2338  2338   CA74
2339  2339   CA74 88              DEY
2340  2340   CA75 D0 DB           BNE DUMP1
2341  2341   CA77
2342  2342   CA77 20 9E C0        JSR NEW_LINE
2343  2343   CA7A 20 B1 C5        JSR STILL_DATA
2344  2344   CA7D
2345  2345   CA7D 60              RTS
2346  2346   CA7E
2347  2347   CA7E            ;-------------------------------------------------------------
2348  2348   CA7E
2349  2349   CA7E            ; WAIT UNTIL LCD READY BIT SET
2350  2350   CA7E
2351  2351   CA7E 48         LCDREADY         PHA
2352  2352   CA7F AD 02 90   READY            LDA COMMAND_READ
2353  2353   CA82 29 80                       AND #BUSY
2354  2354   CA84 D0 F9                       BNE READY   ; LOOP IF BUSY FLAG = 1
2355  2355   CA86 68                          PLA
2356  2356   CA87 60              RTS
2357  2357   CA88
2358  2358   CA88
2359  2359   CA88            LCD_COMMAND_WRITE
2360  2360   CA88 20 7E CA                      JSR LCDREADY
2361  2361   CA8B 8D 00 90                      STA COMMAND_WRITE
2362  2362   CA8E 60                 RTS
2363  2363   CA8F
2364  2364   CA8F
2365  2365   CA8F 20 7E CA   LCD_DATA_WRITE  JSR LCDREADY
2366  2366   CA92 8D 01 90                   STA DATA_WRITE
2367  2367   CA95 60                         RTS
2368  2368   CA96
```

```
2369  2369   CA96
2370  2370   CA96
2371  2371   CA96 20 7E CA    CLR_SCREEN2      JSR LCDREADY
2372  2372   CA99 A9 01                        LDA #1
2373  2373   CA9B 20 88 CA         JSR LCD_COMMAND_WRITE
2374  2374   CA9E 60                           RTS
2375  2375   CA9F
2376  2376   CA9F A9 38       INITLCD          LDA #38H
2377  2377   CAA1 20 88 CA                     JSR LCD_COMMAND_WRITE
2378  2378   CAA4 A9 0C                        LDA #0CH
2379  2379   CAA6 20 88 CA              JSR LCD_COMMAND_WRITE
2380  2380   CAA9 20 96 CA                     JSR CLR_SCREEN2
2381  2381   CAAC A2 00                        LDX #0
2382  2382   CAAE A0 00            LDY #0
2383  2383   CAB0 20 B4 CA         JSR GOTO_XY
2384  2384   CAB3 60                           RTS
2385  2385   CAB4
2386  2386   CAB4
2387  2387   CAB4             ; GOTO_XY(X,Y)
2388  2388   CAB4             ; ENTRY: A = Y POSITION
2389  2389   CAB4             ;        B = X POSITION
2390  2390   CAB4
2391  2391   CAB4 8A          GOTO_XY          TXA
2392  2392   CAB5 C9 00                        CMP #0
2393  2393   CAB7 D0 08            BNE CASE1
2394  2394   CAB9 98                           TYA
2395  2395   CABA 18               CLC
2396  2396   CABB 69 80                        ADC #80H
2397  2397   CABD 20 88 CA                     JSR LCD_COMMAND_WRITE
2398  2398   CAC0 60               RTS
2399  2399   CAC1
2400  2400   CAC1 C9 01       CASE1            CMP #1
2401  2401   CAC3 D0 08                        BNE CASE2
2402  2402   CAC5 98                           TYA
2403  2403   CAC6 18               CLC
2404  2404   CAC7 69 C0            ADC #0C0H
2405  2405   CAC9 20 88 CA         JSR LCD_COMMAND_WRITE
2406  2406   CACC 60               RTS
2407  2407   CACD
2408  2408   CACD 60          CASE2           RTS
2409  2409   CACE
2410  2410   CACE
2411  2411   CACE
2412  2412   CACE             ; WRITE ASCII CODE TO LCD AT CURRENT POSITION
2413  2413   CACE             ; ENTRY: A
2414  2414   CACE
2415  2415   CACE 20 7E CA    PUTCH_LCD        JSR LCDREADY
2416  2416   CAD1 20 8F CA                     JSR LCD_DATA_WRITE
2417  2417   CAD4 60                           RTS
2418  2418   CAD5
2419  2419   CAD5             ; TEST LCD
2420  2420   CAD5
2421  2421   CAD5 AD 02 90    TEST_LCD         LDA COMMAND_READ
2422  2422   CAD8 29 80                        AND #BUSY
2423  2423   CADA F0 01                        BEQ FOUND_LCD
2424  2424   CADC
2425  2425   CADC 60              RTS
2426  2426   CADD
2427  2427   CADD 20 9F CA    FOUND_LCD        JSR INITLCD
2428  2428   CAE0
2429  2429   CAE0 A9 36           LDA #'6'
2430  2430   CAE2 20 CE CA        JSR PUTCH_LCD
2431  2431   CAE5 A9 35           LDA #'5'
2432  2432   CAE7 20 CE CA        JSR PUTCH_LCD
```

```
2433  2433   CAEA A9 30           LDA #'0'
2434  2434   CAEC 20 CE CA        JSR PUTCH_LCD
2435  2435   CAEF A9 32           LDA #'2'
2436  2436   CAF1 20 CE CA        JSR PUTCH_LCD
2437  2437   CAF4 A9 20           LDA #' '
2438  2438   CAF6 20 CE CA        JSR PUTCH_LCD
2439  2439   CAF9 A9 4B           LDA #'K'
2440  2440   CAFB 20 CE CA        JSR PUTCH_LCD
2441  2441   CAFE A9 49           LDA #'I'
2442  2442   CB00 20 CE CA        JSR PUTCH_LCD
2443  2443   CB03 A9 54           LDA #'T'
2444  2444   CB05 20 CE CA        JSR PUTCH_LCD
2445  2445   CB08 A9 20                     LDA #' '
2446  2446   CB0A 20 CE CA        JSR PUTCH_LCD
2447  2447   CB0D A9 33                     LDA #'3'
2448  2448   CB0F 20 CE CA        JSR PUTCH_LCD
2449  2449   CB12 A9 32                     LDA #'2'
2450  2450   CB14 20 CE CA        JSR PUTCH_LCD
2451  2451   CB17 A9 4B                     LDA #'K'
2452  2452   CB19 20 CE CA        JSR PUTCH_LCD
2453  2453   CB1C                          ; LDA #'B'
2454  2454   CB1C              ;  JSR PUTCH_LCD
2455  2455   CB1C
2456  2456   CB1C A9 20           LDA #' '
2457  2457   CB1E 20 CE CA        JSR PUTCH_LCD
2458  2458   CB21
2459  2459   CB21 A9 52           LDA #'R'
2460  2460   CB23 20 CE CA        JSR PUTCH_LCD
2461  2461   CB26 A9 41                     LDA #'A'
2462  2462   CB28 20 CE CA        JSR PUTCH_LCD
2463  2463   CB2B A9 4D                     LDA #'M'
2464  2464   CB2D 20 CE CA        JSR PUTCH_LCD
2465  2465   CB30
2466  2466   CB30 A0 00           LDY #0
2467  2467   CB32 A2 01                     LDX #1
2468  2468   CB34 20 B4 CA                  JSR GOTO_XY
2469  2469   CB37
2470  2470   CB37 A9 32                     LDA #'2'
2471  2471   CB39 20 CE CA        JSR PUTCH_LCD
2472  2472   CB3C A9 34                     LDA #'4'
2473  2473   CB3E 20 CE CA        JSR PUTCH_LCD
2474  2474   CB41 A9 30                     LDA #'0'
2475  2475   CB43 20 CE CA        JSR PUTCH_LCD
2476  2476   CB46 A9 30                     LDA #'0'
2477  2477   CB48 20 CE CA        JSR PUTCH_LCD
2478  2478   CB4B A9 20                     LDA #' '
2479  2479   CB4D 20 CE CA        JSR PUTCH_LCD
2480  2480   CB50 A9 55                     LDA #'U'
2481  2481   CB52 20 CE CA        JSR PUTCH_LCD
2482  2482   CB55 A9 41                     LDA #'A'
2483  2483   CB57 20 CE CA        JSR PUTCH_LCD
2484  2484   CB5A A9 52                     LDA #'R'
2485  2485   CB5C 20 CE CA        JSR PUTCH_LCD
2486  2486   CB5F A9 54                     LDA #'T'
2487  2487   CB61 20 CE CA                  JSR PUTCH_LCD
2488  2488   CB64
2489  2489   CB64 A9 20           LDA #' '
2490  2490   CB66 20 CE CA        JSR PUTCH_LCD
2491  2491   CB69 A9 4C                     LDA #'L'
2492  2492   CB6B 20 CE CA        JSR PUTCH_LCD
2493  2493   CB6E A9 43                     LDA #'C'
2494  2494   CB70 20 CE CA        JSR PUTCH_LCD
2495  2495   CB73 A9 44                     LDA #'D'
2496  2496   CB75 20 CE CA        JSR PUTCH_LCD
```

```
2497  2497  CB78
2498  2498  CB78
2499  2499  CB78 60                        RTS
2500  2500  CB79
2501  2501  CB79
2502  2502  CB79
2503  2503  CB79 A9 85       TEST_TICK  LDA #SERVICE_IRQ&$FF
2504  2504  CB7B 85 FE                 STA $FE
2505  2505  CB7D A9 CB          LDA #SERVICE_IRQ>>8&$FF
2506  2506  CB7F 85 FF          STA $FF
2507  2507  CB81
2508  2508  CB81 58             CLI       ; ENABLE IRQ
2509  2509  CB82 4C 82 CB             JMP $
2510  2510  CB85
2511  2511  CB85
2512  2512  CB85              SERVICE_IRQ
2513  2513  CB85
2514  2514  CB85                      ; SED       ; SET DECIMAL MODE
2515  2515  CB85 E6 B0                 INC SEC100
2516  2516  CB87 A5 B0          LDA SEC100
2517  2517  CB89 C9 0A          CMP #10
2518  2518  CB8B 30 10          BMI SKIP1
2519  2519  CB8D A9 00          LDA #0
2520  2520  CB8F 85 B0          STA SEC100
2521  2521  CB91
2522  2522  CB91 18             CLC
2523  2523  CB92 A5 AF          LDA SEC
2524  2524  CB94 69 01          ADC #1
2525  2525  CB96 85 AF                STA SEC
2526  2526  CB98
2527  2527  CB98 A5 AF          LDA SEC
2528  2528  CB9A 8D 00 80       STA GPIO1
2529  2529  CB9D
2530  2530  CB9D              SKIP1
2531  2531  CB9D 40                    RTI
2532  2532  CB9E
2533  2533  CB9E              ;-----------------------------------------------------------
2534  2534  CB9E
2535  2535  CB9E A9 00        MAIN    LDA #0
2536  2536  CBA0 85 8C                STA BUFFER
2537  2537  CBA2 85 8D          STA BUFFER+1
2538  2538  CBA4 85 92          STA INVALID   ; CLEAR INVALID FLAG
2539  2539  CBA6 85 AE          STA MUTE      ; BEEP ON
2540  2540  CBA8
2541  2541  CBA8              ; INSERT 6502 TEXT
2542  2542  CBA8
2543  2543  CBA8 A9 AF          LDA #0AFH
2544  2544  CBAA 85 91                STA BUFFER+5
2545  2545  CBAC A9 AE          LDA #0AEH
2546  2546  CBAE 85 90          STA BUFFER+4
2547  2547  CBB0 A9 BD          LDA #0BDH
2548  2548  CBB2 85 8F          STA BUFFER+3
2549  2549  CBB4 A9 9B          LDA #9BH
2550  2550  CBB6 85 8E          STA BUFFER+2
2551  2551  CBB8
2552  2552  CBB8
2553  2553  CBB8
2554  2554  CBB8              ; STORE VECTOR INTERRUPT
2555  2555  CBB8
2556  2556  CBB8 A9 BC          LDA #NMI_SERVICE&0FFH   ; NMI MUST BE SET BEFORE USING SIN
2557  2557  CBBA 85 FA          STA $FA
2558  2558  CBBC 85 FE          STA $FE
2559  2559  CBBE
2560  2560  CBBE A9 C8          LDA #(NMI_SERVICE>>8)
```

```
2561   2561   CBC0 85 FB         STA $FB
2562   2562   CBC2 85 FF         STA $FF
2563   2563   CBC4
2564   2564   CBC4 A2 FF         LDX #$FF
2565   2565   CBC6 9A            TXS         ; SET SYSTEM STACK TO 1FFH
2566   2566   CBC7 A9 7F               LDA #$7F    ; AND USER STACK TO 17FH
2567   2567   CBC9 85 9D         STA USER_S
2568   2568   CBCB
2569   2569   CBCB D8            CLD
2570   2570   CBCC 78            SEI         ; DISABLE IRQ
2571   2571   CBCD
2572   2572   CBCD A9 00         LDA #0
2573   2573   CBCF 85 94         STA STATE    ; INITIAL STATE
2574   2574   CBD1 85 95         STA ZERO_FLAG
2575   2575   CBD3
2576   2576   CBD3 A9 00               LDA #0
2577   2577   CBD5 85 96         STA DISPLAY
2578   2578   CBD7 85 98         STA PC_USER
2579   2579   CBD9 A9 02         LDA #02H
2580   2580   CBDB 85 97         STA DISPLAY+1
2581   2581   CBDD 85 99         STA PC_USER+1
2582   2582   CBDF
2583   2583   CBDF
2584   2584   CBDF A5 96               LDA DISPLAY
2585   2585   CBE1 85 84         STA HL
2586   2586   CBE3 A5 97         LDA DISPLAY+1
2587   2587   CBE5 85 85         STA HL+1
2588   2588   CBE7
2589   2589   CBE7
2590   2590   CBE7
2591   2591   CBE7               ;JSR ADDRESS_DISPLAY
2592   2592   CBE7 A0 00               LDY #0
2593   2593   CBE9 B1 84         LDA (HL),Y
2594   2594   CBEB               ;JSR DATA_DISPLAY
2595   2595   CBEB
2596   2596   CBEB A5 A8               LDA COLD
2597   2597   CBED C9 99         CMP #99H
2598   2598   CBEF F0 15         BEQ WARM_BOOT
2599   2599   CBF1
2600   2600   CBF1 A9 99         LDA #99H
2601   2601   CBF3 85 A8         STA COLD
2602   2602   CBF5
2603   2603   CBF5
2604   2604   CBF5 A9 FF               LDA #$FF
2605   2605   CBF7 8D 00 80      STA GPIO1     ; TEST GPIO1
2606   2606   CBFA
2607   2607   CBFA 20 1B CA            JSR COLD_MESSAGE
2608   2608   CBFD 20 F8 C9      JSR BEEP
2609   2609   CC00 20 41 CA      JSR CLR_SCREEN
2610   2610   CC03 20 38 CA      JSR COLD_TERMINAL
2611   2611   CC06
2612   2612   CC06
2613   2613   CC06
2614   2614   CC06
2615   2615   CC06
2616   2616   CC06
2617   2617   CC06             WARM_BOOT
2618   2618   CC06
2619   2619   CC06                     ;LDA #'*'
2620   2620   CC06               ;JSR SEND_BYTE
2621   2621   CC06
2622   2622   CC06               ;JSR CLR_SCREEN
2623   2623   CC06               ;JSR COLD_TERMINAL
2624   2624   CC06 20 D5 CA      JSR TEST_LCD
```

```
2625  2625   CC09
2626  2626   CC09              ;JSR PRINT_LINE
2627  2627   CC09
2628  2628   CC09 A9 00        LDA #0
2629  2629   CC0B 8D 00 80     STA GPIO1
2630  2630   CC0E
2631  2631   CC0E 20 51 C3    LOOP3  JSR SCANKEY
2632  2632   CC11 20 CA C3           JSR KEYEXE
2633  2633   CC14 20 ED C9     JSR BEEP3
2634  2634   CC17 4C 0E CC     JMP LOOP3
2635  2635   CC1A
2636  2636   CC1A             ;-------------------------------------------------------
2637  2637   CC1A 00          START_MSG .BYTE 0
2638  2638   CC1B 00                    .BYTE 0
2639  2639   CC1C 9B                    .BYTE 9BH
2640  2640   CC1D BD                    .BYTE 0BDH
2641  2641   CC1E AE                    .BYTE 0AEH
2642  2642   CC1F AF               .BYTE 0AFH
2643  2643   CC20 00                    .BYTE 0
2644  2644   CC21 00               .BYTE 0
2645  2645   CC22 00               .BYTE 0
2646  2646   CC23 00               .BYTE 0
2647  2647   CC24 00               .BYTE 0
2648  2648   CC25 00                    .BYTE 0
2649  2649   CC26
2650  2650   CC26
2651  2651   CC26 BD          SEGTAB .BYTE0BDH    ;'0'
2652  2652   CC27 30            .BYTE030H    ;'1'
2653  2653   CC28 9B            .BYTE09BH    ;'2'
2654  2654   CC29 BA            .BYTE0BAH    ;'3'
2655  2655   CC2A 36            .BYTE036H    ;'4'
2656  2656   CC2B AE            .BYTE0AEH    ;'5'
2657  2657   CC2C AF            .BYTE0AFH    ;'6'
2658  2658   CC2D 38            .BYTE038H    ;'7'
2659  2659   CC2E BF            .BYTE0BFH    ;'8'
2660  2660   CC2F BE            .BYTE0BEH    ;'9'
2661  2661   CC30 3F            .BYTE03FH    ;'A'
2662  2662   CC31 A7            .BYTE0A7H    ;'B'
2663  2663   CC32 8D            .BYTE08DH    ;'C'
2664  2664   CC33 B3            .BYTE0B3H    ;'D'
2665  2665   CC34 8F            .BYTE08FH    ;'E'
2666  2666   CC35 0F            .BYTE00FH    ;'F'
2667  2667   CC36
2668  2668   CC36
2669  2669   CC36
2670  2670   CC36             ; Key-posistion-code to key-internal-code conversion table.
2671  2671   CC36
2672  2672   CC36             KEYTAB:
2673  2673   CC36 03          K0  .BYTE03H  ;HEX_3
2674  2674   CC37 07          K1  .BYTE07H  ;HEX_7
2675  2675   CC38 0B          K2  .BYTE0BH  ;HEX_B
2676  2676   CC39 0F          K3  .BYTE0FH  ;HEX_F
2677  2677   CC3A 20          K4  .BYTE20H  ;NOT USED
2678  2678   CC3B 21          K5  .BYTE21H  ;NOT USED
2679  2679   CC3C 02          K6  .BYTE02H  ;HEX_2
2680  2680   CC3D 06          K7  .BYTE06H  ;HEX_6
2681  2681   CC3E 0A          K8  .BYTE0AH  ;HEX_A
2682  2682   CC3F 0E          K9  .BYTE0EH  ;HEX_E
2683  2683   CC40 22          K0A .BYTE22H  ;NOT USED
2684  2684   CC41 23          K0B .BYTE23H  ;NOT USED
2685  2685   CC42 01          K0C .BYTE01H  ;HEX_1
2686  2686   CC43 05          K0D .BYTE05H  ;HEX_5
2687  2687   CC44 09          K0E .BYTE09H  ;HEX_9
2688  2688   CC45 0D          K0F .BYTE0DH  ;HEX_D
```

```
2689  2689   CC46 13          K10  .BYTE13H  ;STEP
2690  2690   CC47 1F          K11  .BYTE1FH  ;TAPERD
2691  2691   CC48 00          K12  .BYTE00H  ;HEX_0
2692  2692   CC49 04          K13  .BYTE04H  ;HEX_4
2693  2693   CC4A 08          K14  .BYTE08H  ;HEX_8
2694  2694   CC4B 0C          K15  .BYTE0CH  ;HEX_C
2695  2695   CC4C 12          K16  .BYTE12H  ;GO
2696  2696   CC4D 1E          K17  .BYTE1EH  ;TAPEWR
2697  2697   CC4E 1A          K18  .BYTE1AH  ;CBR
2698  2698   CC4F 18          K19  .BYTE18H  ;PC
2699  2699   CC50 1B          K1A  .BYTE1BH  ;REG
2700  2700   CC51 19          K1B  .BYTE19H  ;ADDR
2701  2701   CC52 17          K1C  .BYTE17H  ;DEL
2702  2702   CC53 1D          K1D  .BYTE1DH  ;RELA
2703  2703   CC54 15          K1E  .BYTE15H  ;SBR
2704  2704   CC55 11          K1F  .BYTE11H  ;-
2705  2705   CC56 14          K20  .BYTE14H  ;DATA
2706  2706   CC57 10          K21  .BYTE10H  ;+
2707  2707   CC58 16          K22  .BYTE16H  ;INS
2708  2708   CC59 1C          K23  .BYTE1CH  ;MOVE
2709  2709   CC5A
2710  2710   CC5A             ; PAGE FOR CONSTANT STRINGS AREA
2711  2711   CC5A
2712  2712   EF00                  .ORG 0EF00H        ; ROM MONITOR
2713  2713   EF00                       ; .ORG 06F00H        ; RAM TEST
2714  2714   EF00
2715  2715   EF00
2716  2716   EF00 0A0D36353032TEXT1      .BYTE 10,13,"6502 MICROPROCESSOR KIT V1.1",0
2716  2716   EF06 204D4943524F50524F434553534F52204B49542056312E3100
2717  2717   EF1F 0A0D4C6F6164TEXT2      .BYTE 10,13,"Load HEX file...",0
2717  2717   EF25 204845582066696C652E2E2E00
2718  2718   EF32 436F6D706C65TEXT3      .BYTE "Completed...",0
2718  2718   EF38 7465642E2E2E00
2719  2719   EF3F
2720  2720   EF3F
2721  2721   EF3F 3E 3E 00    PROMPT     .BYTE ">>", 0
2722  2722   EF42
2723  2723   EF42
2724  2724   EF42
2725  2725   EF42
2726  2726   EF42                    ; VECTOR NMI,RESET AND IRQ
2727  2727   EF42
2728  2728   EF42
2729  2729   FFFA                .ORG 0FFFAH
2730  2730   FFFA
2731  2731   FFFA 32 CA                 .WORD  NMI
2732  2732   FFFC 00 C0          .WORD  0C000H  ; RESET VECTOR
2733  2733   FFFE 35 CA          .WORD  IRQ        ; IRQ VECTOR
2734  2734   0000
2735  2735   0000
2736  2736   0000
2737  2737   0000
2738  2738   0000                .END
2739  2739   0000
2740  2740   0000
2741  2741   0000
tasm: Number of errors = 0
2746
```

NOTE