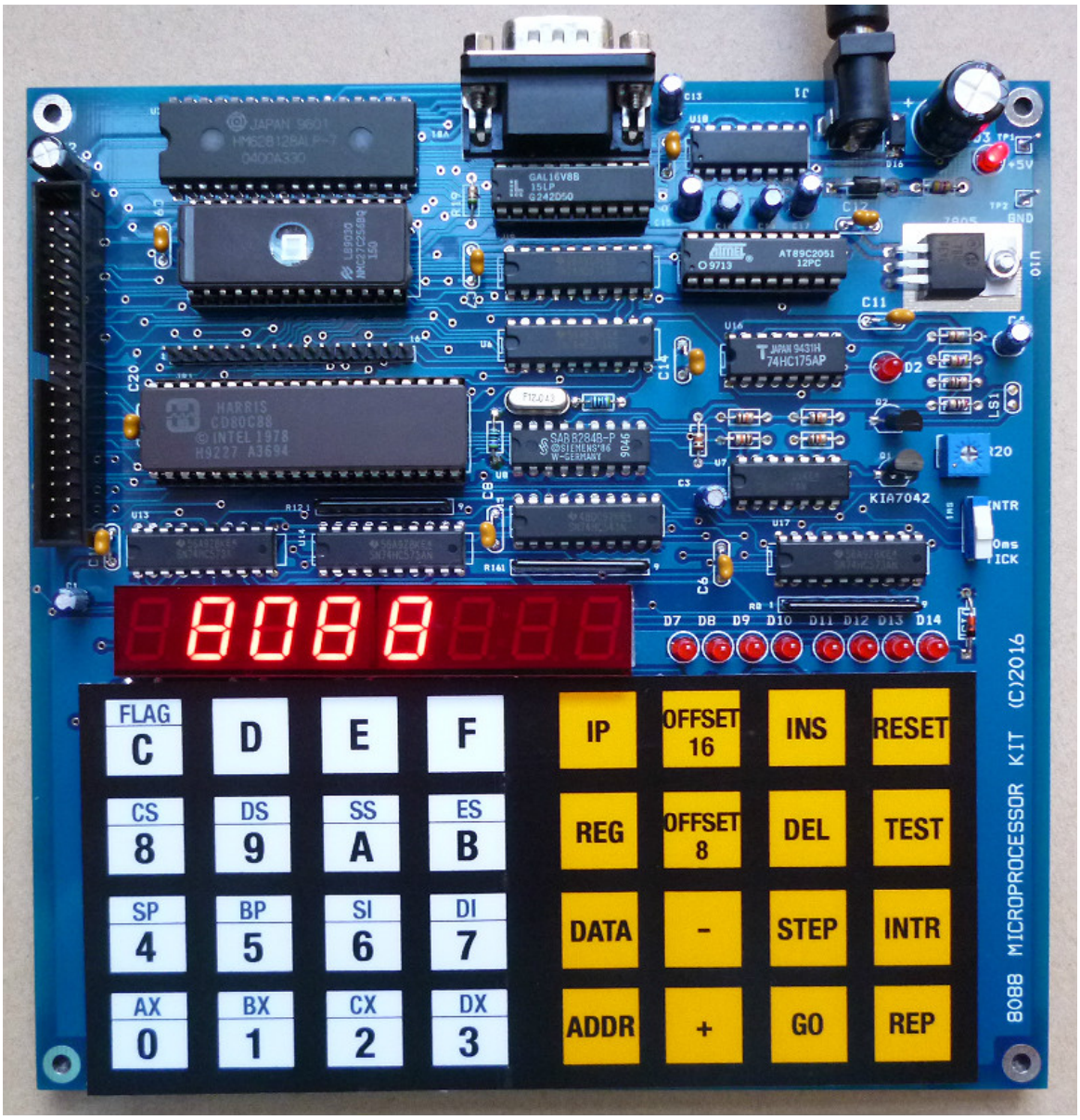
8088 Microprocessor Kit User's Manual



Rev 1.0, January 2017

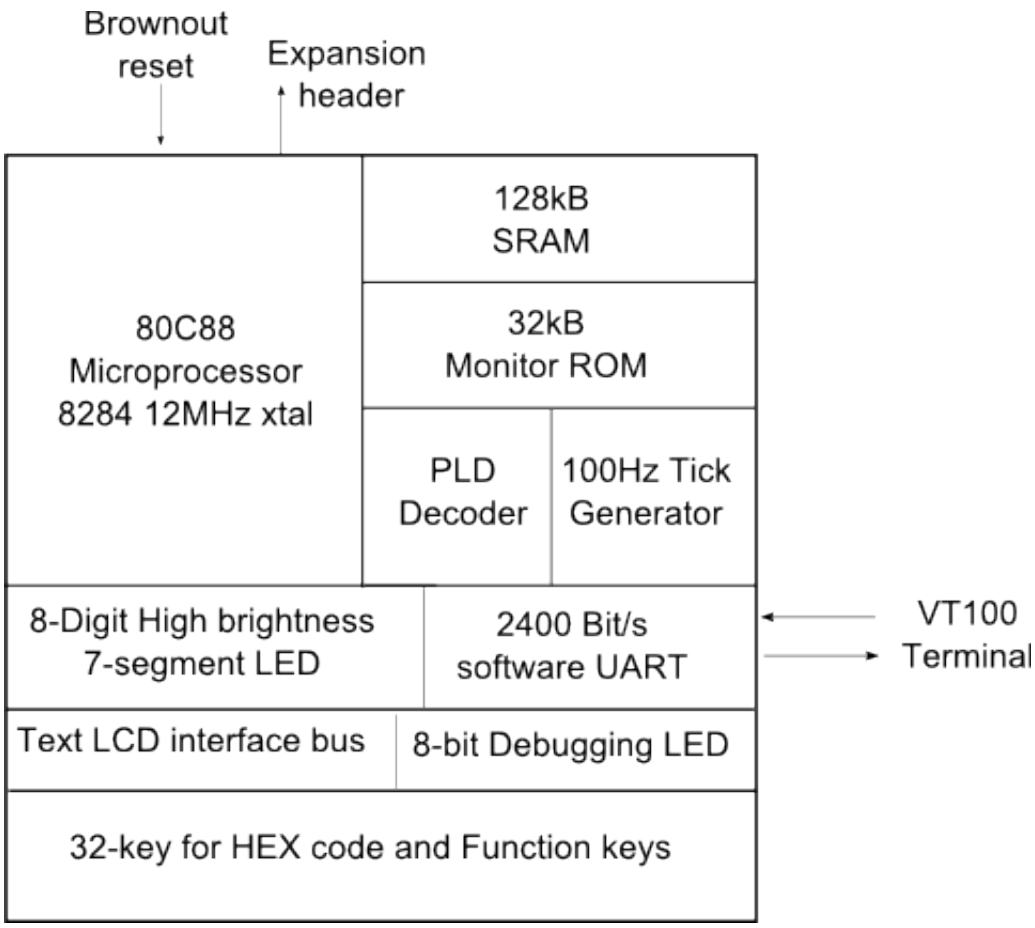
8088 MICROPROCESSOR KIT

CONTENTS

OVERVIEW

The 8088 Microprocessor kit is the educational kit designed for self learning the popular x86 coding. The CPU is CMOS chip, Harris 80C88 with 128kB user RAM and 32kB monitor ROM. The kit provides hex display and hex keys. It can display memory contents with 64kB logical locations. The hex key can use for entering the 8088 instructions. We can learn the operation of 8088 CPU by testing the code running with single step or break point easily. The kit also has serial pot for Intel hex file downloading.

FUNCTIONAL BLOCK DIAGRAM



Notes

1. UART is software control for low speed asynchronous communication.

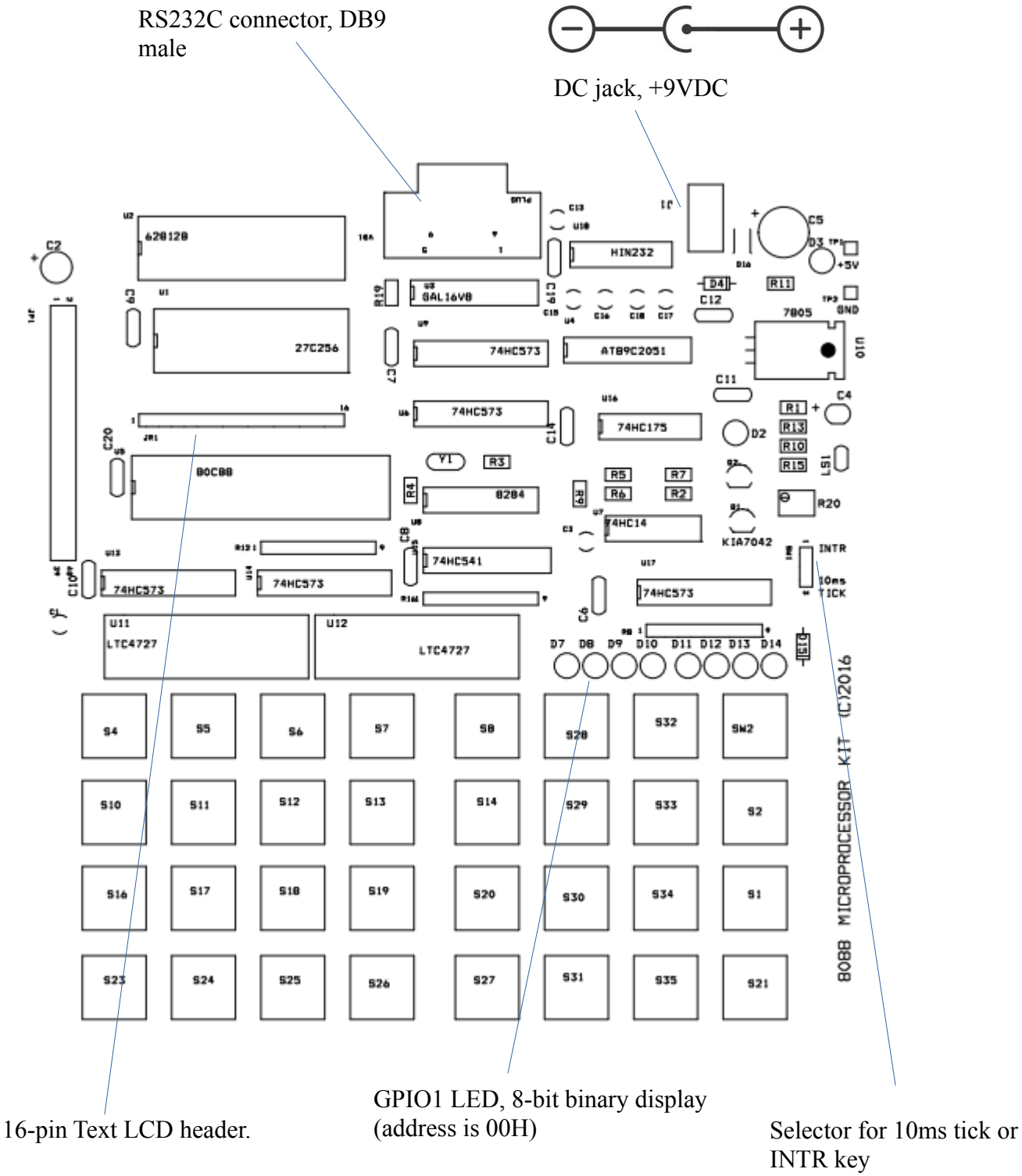
2. The kit has 8-bit LCD module interfacing bus.

3. 100Hz Tick generator is for interrupt experiment.

4. Ports for display and keypad interfacing were built with discrete logic IC chips.

5. Memory and Port decoders are made with Programmable Logic Device, PLD.

HARDWARE LAYOUT



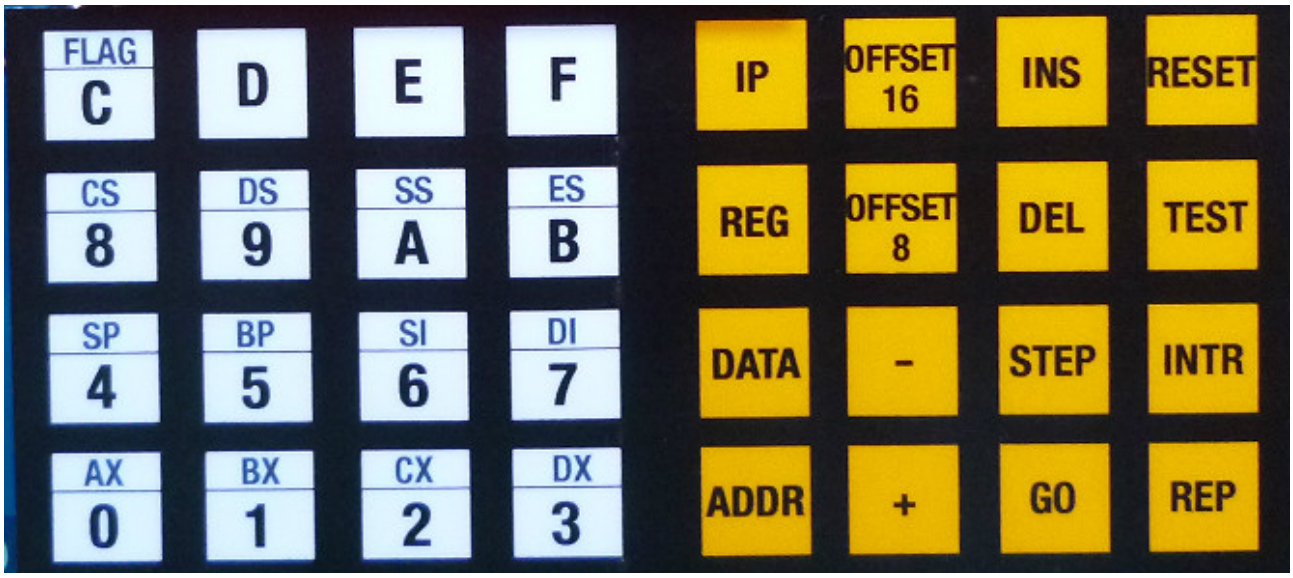
Important Notes

1. Plugging or removing the LCD module must be done when the kit is powered off!

2. AC adapter should provide approx. +9VDC, higher voltage will cause the voltage regulator chip becomes hot.

3. The kit has diode protection for wrong polarity of adapter jack. If the center pin is not the positive (+), the diode will be reverse bias, preventing wrong polarity feeding to voltage regulator.

KEYBOARD LAYOUT



HEX keys Hexadecimal number 0 to F with associated user registers, AX, BX, CX, DX,SP, BP, SI, DI, CS, DS, SS, ES and flag display.

CPU control keys

RESET Reset the CPU, the 8088 will JUMP to location FFFF0.

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

Monitor function keys

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

INS Insert one byte to the next location, the 1024 bytes will be shifted down.

DEL Delete one byte at current display, the next 1024 bytes will be moved up.

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

GO Jump from monitor program to user code

- Decrement current display address by one

+ Increment current display address by one

IP Set current display address with user Program Counter

REG Display user registers, used with HEX key.

0	AX register
1	BX register
2	CX register
3	DX register
4	SP register
5	BP register
6	SI register
7	DI register
8	CS register
9	DS register
A	SS register
B	ES register
C	FLAG register

DATA Set entry mode of hex keys to Data field

ADDR Set entry mode of hex keys to Address field

OFFSET16 Compute 16-bit offset, used with key + for Destination and key GO

OFFSET8 Compute 8-bit offset, used with key + for Destination and key GO

TEST Write the text to LCD if connected and test gpio1 LED

HARDWARE FEATURES

Hardware features:

-CPU: Harris 80C88 CMOS Microprocessor @4MHz
-Oscillator: 8284 with 12MHz Xtal
-Memory: 128kB RAM, 32kB EPROM
-Memory and I/O Decoder chip: Programmable Logic Device GAL16V8D
-Display: high brightness 6-digit 7-segment LED
-Keyboard: 32 keys
-RS232 port: software controlled UART 2400 bit/s 8n1
-Debugging LED: 8-bit GPIO1 LED at location 00H
-Tick: 10ms tick produced by 89C2051 for time trigger experiment
-Text LCD interface: direct CPU bus interface text LCD
-Brownout reset: KIA7042 reset chip for power brownout reset
-Power consumption: 180mA @7.5V AC adapter
-Expansion header: 40-pin header

MONITOR PROGRAM FEATURES

MONITOR program features:

-Enter 8088 instructions using hex code directly
-Test code running with single step or break point
-User registers for status capturing
-Insert/delete byte
-Intel hex file downloading using 2400 bit/s RS232 port.

MEMORY AND I/O MAPS

The kit provides two spaces of memory, i.e. 1) RAM, 2) monitor ROM and space for I/O ports.

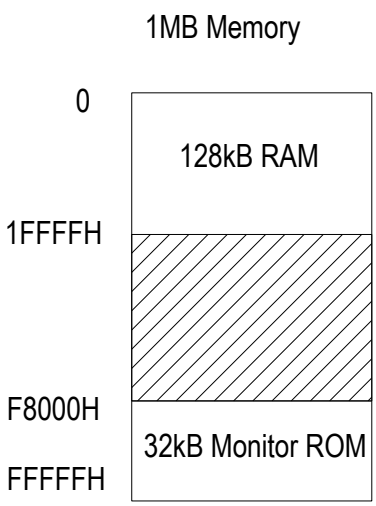
On power up, the 8088 will jump to reset vector at location FFFF0.

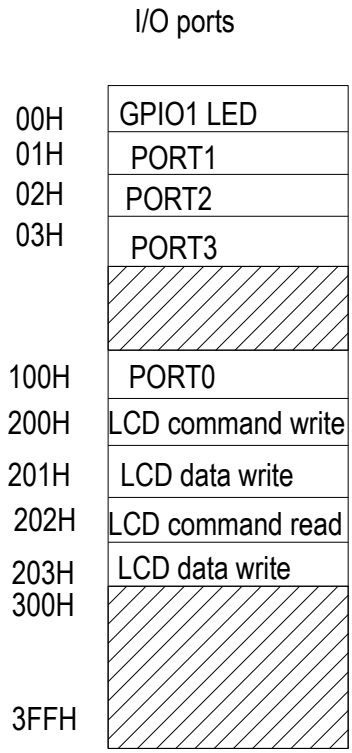
The 8088 can address up to 1MB with 20-bit address lines, A0-A19. The 1st 128kB is RAM. User can set the interrupt vectors in RAM easily. The ROM is located at F8000-FFFFF.

I/O ports are located from 00H to 2FFH. From 300H to 3FFH are for expansion.

GPIO1 LED is located at 0. User can use instruction that write 8-bit data with 8-bit address easily, e.g.,

```
MOV AL,#1
OUT  0,AL
```





GETTING STARTED

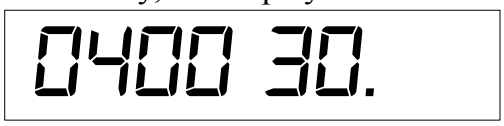
The kit accepts DC power supply with minimum voltage of +7.5V. It draws DC current approx. 180mA. However we can use +9VDC from any AC adapter. The example of AC adapter is shown below.

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.

When power up, we will see the boot message 8088 running.

Press IP key, the display address will be 400. The data field will show its content.



HOW TO ENTER PROGRAM USING HEX CODE

Let us try enter HEX CODE of the example program to the memory and test it. We write the program with x86 instructions.

Address	Hex code	Label	Instruction	comment
00400	B001	MAIN	MOV AL,1	Load AL with 1
00402	E600		OUT 0,AL	Write AL to GPIO1 @ 00

Our test program has only two instructions.

The first instruction is

`MOV AL,1`

Load AL register with the 8-bit constant, 01.

This instruction has two bytes hex code i.e., B0, and 01. B0 is instruction MOV AL,n and 01 is n.

The 2nd instruction is

`OUT 0,AL`. Copy AL register to output port, gpio1 LED at location 00.

The instruction's machine code is E6. The location of GPIO1 is 00.

The total of hex codes for this small program is 4 bytes that are, B0, 01, E6,00.

The first byte will be entered to location 0400. And the following bytes will be entered at 0401, 0402, 0403, 0404 and so on. The last byte is 00 at 403.

Let us see how to enter these codes into the memory.

Step 1 Press RESET then key IP, the display will show current memory address and its contents.

Shown the location 400 has data A4. There are small dots at the data field indicating the active field, ready for modifying the hex contents.

Step 2 Press key B and key 0. The new hex code B0 will be entered to the location 400.

0400 B0.

Step 3 Press key + to increment the location from 400 to 401. Then enter hex key 1.

0401 01

Repeat Step 3 until completed for the last location. We can verify the hex code with key + or key -.

To change the display location, press key ADDR. The dots will move to Address field. Any hex key pressed will change the display address.

USER REGISTERS DISPLAY

Before we test the code running, let us see how to examine user registers. User registers are the memory block in RAM that used to save the contents of CPU registers after completed a given program running. We can examine the user registers for checking our code running then.

Press key REG, then press key 0, it will show 16-bit content of Data register AX.

0C5E AX

Key 0 to Key 3 are for AX, BX, CX and DX. The content is 16-bit, for example AX = 0C5E or AH=0C and AL=5E. Similar for key REG 1, 2, 3 for BX, CX and DX registers.

Press key Reg, 4, 5, 6, 7 for SP, BP, SI and DI register.

FE00 SP

TEST CODE RUNNING WITH SINGLE STEP

Now get back to our program we have just entered. Let us take a look again.

Address	Hex code	Label	Instruction	comment
00400	B001	MAIN	MOV AL,1	Load AL with 1
00402	E600		OUT 0,AL	Write AL to GPIO1 @ 00

The code has only two instructions.

We will try test the program using single step running.

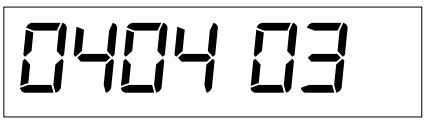
Step 1 Press key IP. We see that the location 400 has B0.

Step 2 Press REG key, then key 0, to check AX register,

Step 3 Press PC then STEP key. The instruction MOV AL,1 will be executed.

The display will show next instruction to be executed, that is OUT 0,AL.

Step 4 Press STEP, we will see the byte 01 will be sent to gpio1 LED.



What is the value on gpio1 LED?

To get key PC back to the location 400, press RESET key, then press IP.

We see that single step running is useful for learning the operation of each instruction. We can check the result with user registers easily.

Try another example,

```
0400                    ORG 400H

0400 FEC0      START    INC AL
0402 E600               OUT 0,AL
0404 EBFA               JMP START
```

Above program is for testing single step running again. Now we see that JMP START is branch instruction that makes CPU jump back to the repeat the body. The body code is simply increment AL the then write it to gpio1 LED.

Enter the hex code then use single step key to test it. See what is happening?

If we keep pressing key STEP and REP, what is happening?

TEST CODE RUNNING WITH BREAK POINT

Single step running enables us to examine the operation of x86 instruction one by one. Another example is to run at CPU speed then stop and return to monitor program to save CPU registers to user registers that we can examine the program operation.

```
0400                ORG 400H

0400 B8FF1E    START    MOV AX,1EFFH  ; LOAD AX WITH 1EFF
0403 BB6D25             MOV BX,256DH  ; LOAD BX WITH 256D
0406 03C3               ADD AX,BX     ; AX=AX+BX
0408 E600               OUT 0,AL      ; WRITE AL TO GPIO1
040A CC                 INT 3    ; BREAK, RETURN TO MONITOR
```

This program adds two 16-bit constants. The first number is loaded to AX register. The 2nd number is loaded to BX register. Then add instruction computes binary addition AX and BX. Result will save to AX register.

At the end, we put INT 3 instruction. INT 3 will generate software interrupt process, by returning to system monitor getting the vector at location 0000C. The service routine for INT 3 will save CPU registers to user registers.

Now let us enter code, when completed, press IP then GO.

Check result in AX register with key REG, 0.

What is the value in AX?

Can you compute such addition by hand? Better try with binary addition.

```
AX=     0001 1110 1111 1111
BX=     0010 0101 0110 1101
AX=     ____________________
```

Convert the result into HEX digit.

Compare your result with 8088 kit running. Check result in AX register.

Now we see that we can test the code execution by key STEP or key GO.

Key STEP will execute only single instruction then return to monitor program, saving the CPU registers to user registers.

Key GO will execute user code at CPU speed. If we put INT 3 at the end, the CPU will return to monitor program and save CPU registers to user registers.

Another example with key GO.

```
0400                ORG 400H

0400 B001      START   MOV AL,1

0402 E600      LOOP    OUT 0,AL
0404 D0C0               ROL  AL,1
0406 E80200             CALL DELAY
0409 EBF7               JMP LOOP

040B B90030    DELAY   MOV CX,3000H
040E E2FE               LOOP $
0410 C3                 RET
```

The program can test run with key GO. However, we see that at the end JMP LOOP will force CPU not to return to monitor program. Instead, to repeat running forever.

Let us enter the hex code and test run with key GO.

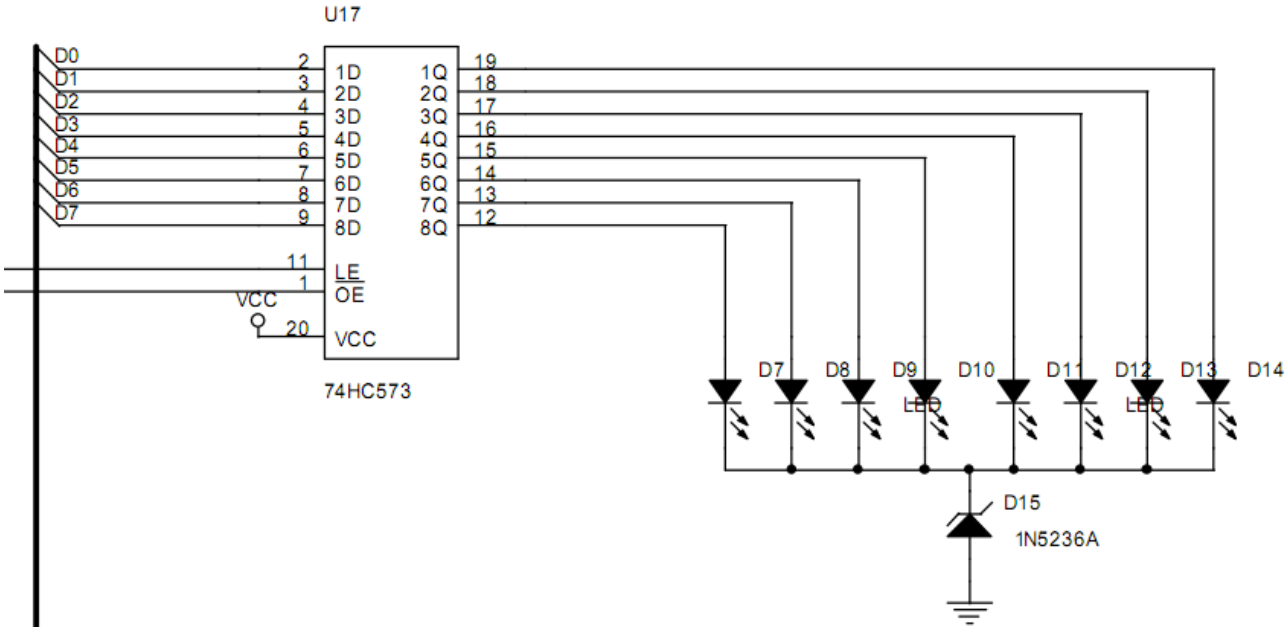
What is happening at gpio1 LED?

Can you change speed of LED running? How?

Can you change the pattern of LED? How?

GPIO1 LED

The 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 00. The output port is 8-bit data flip-flop. Logic 1 at the output will make LED lit.



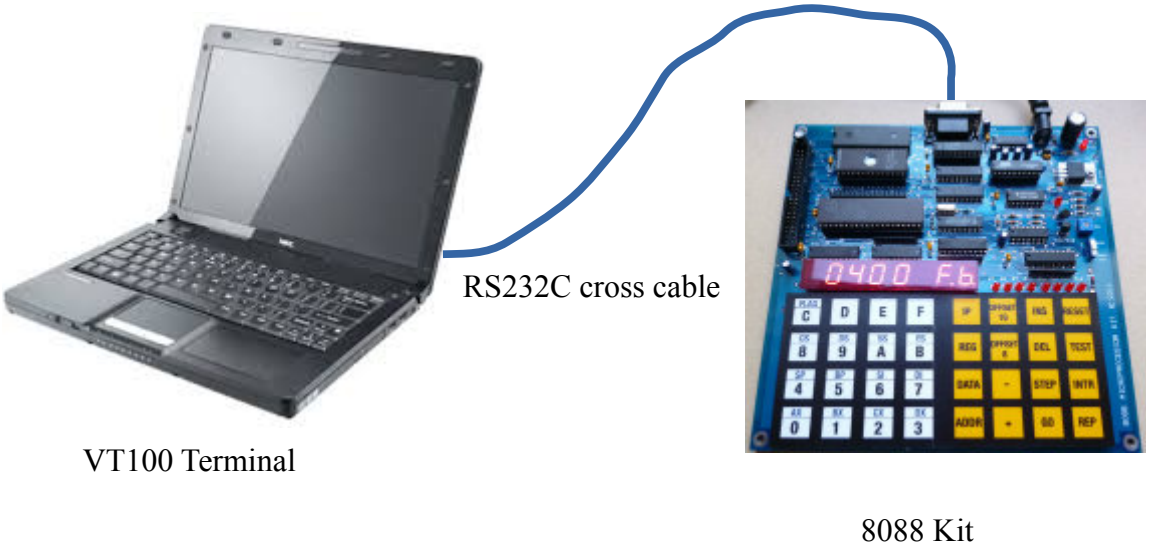
This debug LED is memory location at 00. We can use instruction OUT 0,AL that writes 8-bit data to it will make the LED turn on for bit '1' and turn off for bit '0'.

The hex code for OUT 0,AL is E6, 00. Only two bytes and easy to remember!

Anytime we need to check the 8-bit contents, we can copy it to AL then use this instruction to write it to the gpio1 LED easily.

CONNECTING 8088 KIT TO TERMINAL

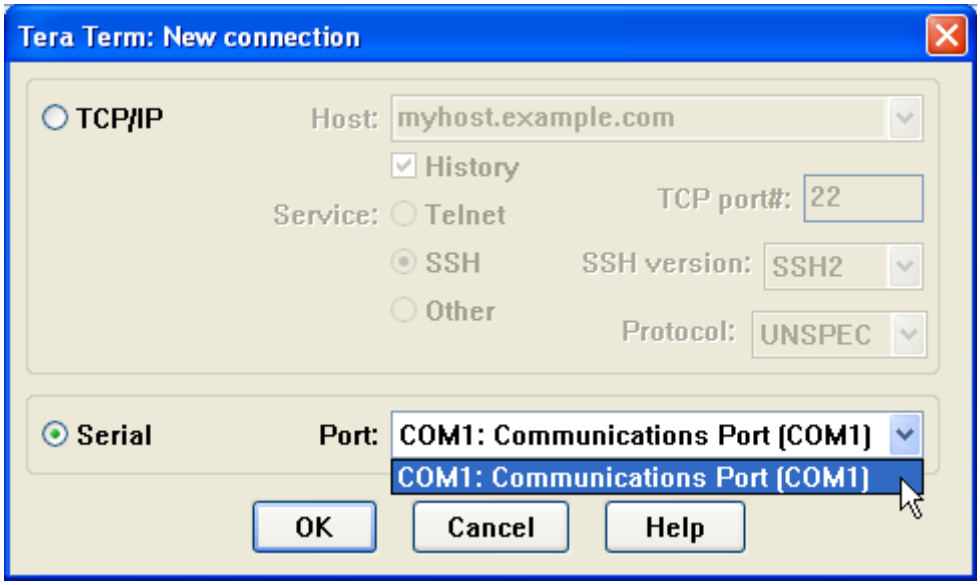
We can connect the 8088 kit to a terminal by RS232C cross cable. You may download free terminal program, teraterm from this URL, http://ttssh2.sourceforge.jp/index.html.en



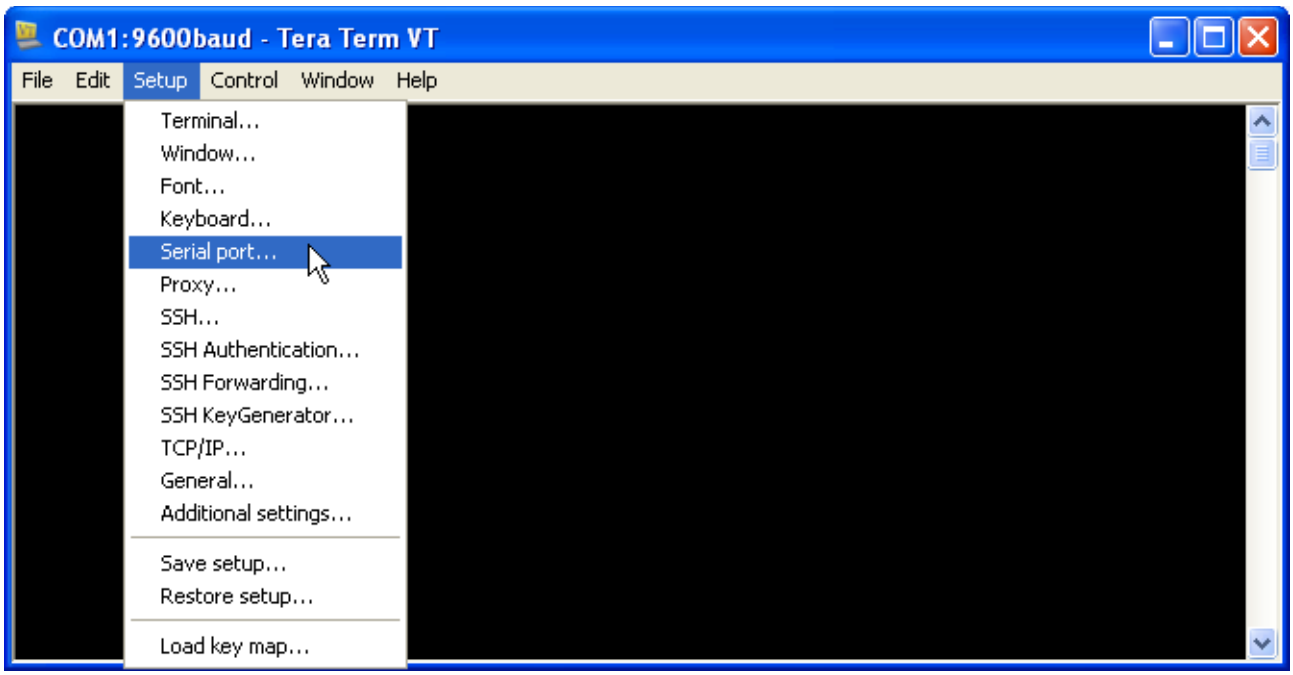
The example shows connecting laptop with COM1 port to the RS232C port of the 8088 kit. New laptop that has no COM port, we may use the USB-RS232 adapter for converting the USB port to RS232 port.

To download Intel 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.

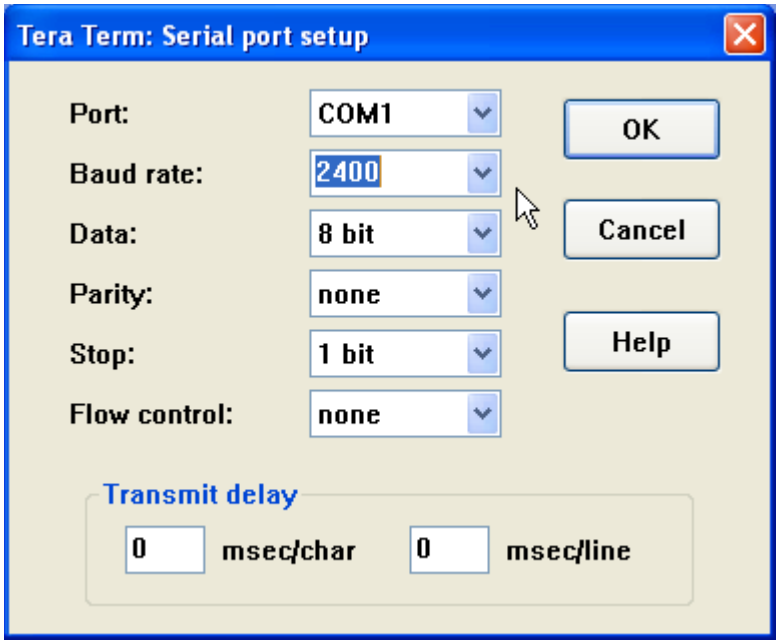
Step 1 Run teraterm, then click at Serial connection.



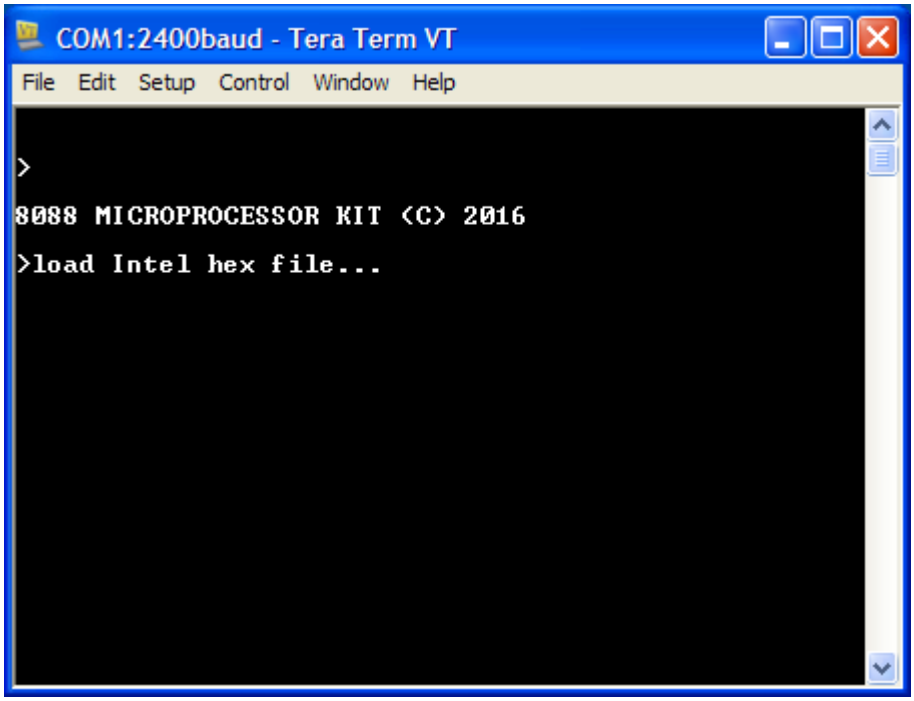
Step 2 Click setup>Serial port.



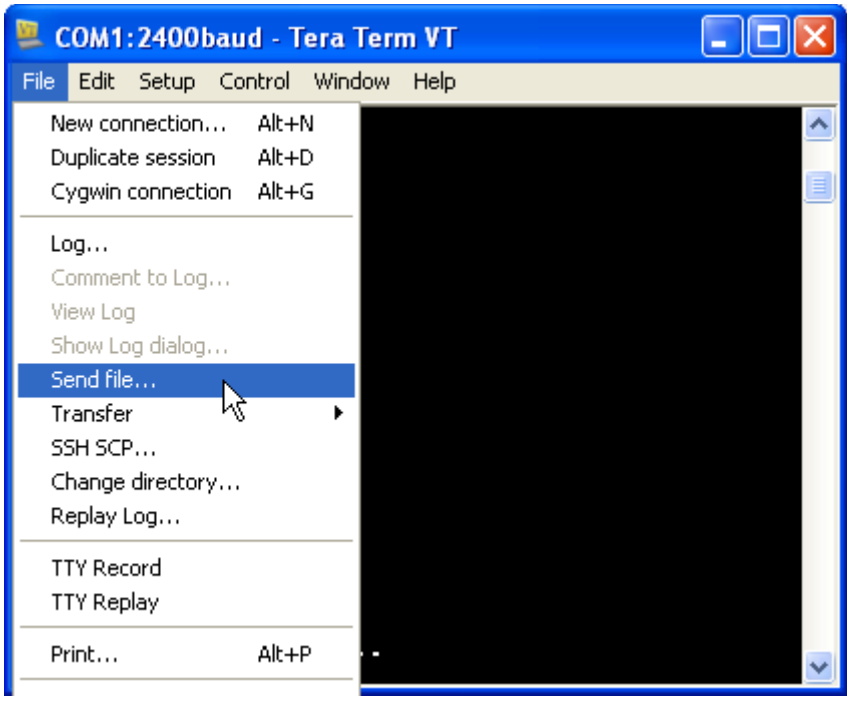
Step 3 Set serial port speed to 2400 and format as shown below.



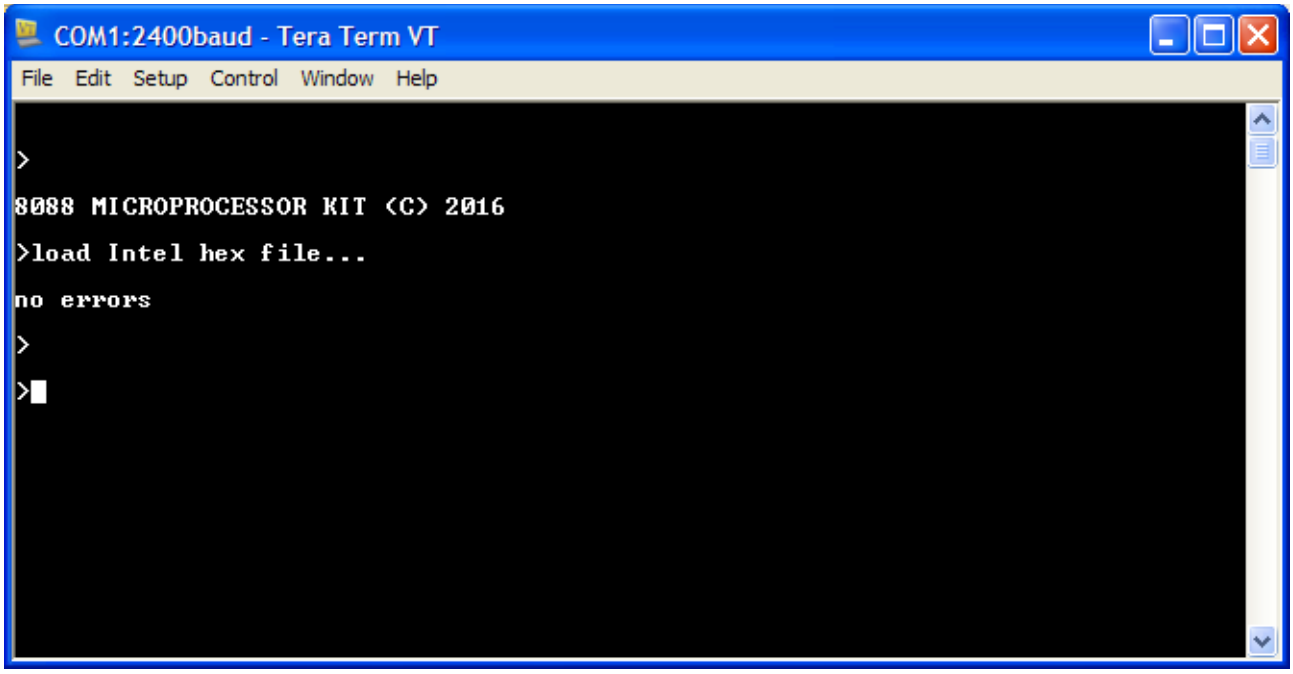
Step 4 Press ENTER key on terminal. The kit will connect terminal automatically. Press key ‘l’ to download Intel hex file.



Step 5 On PC, Click file>Send File>TEST1.HEX.



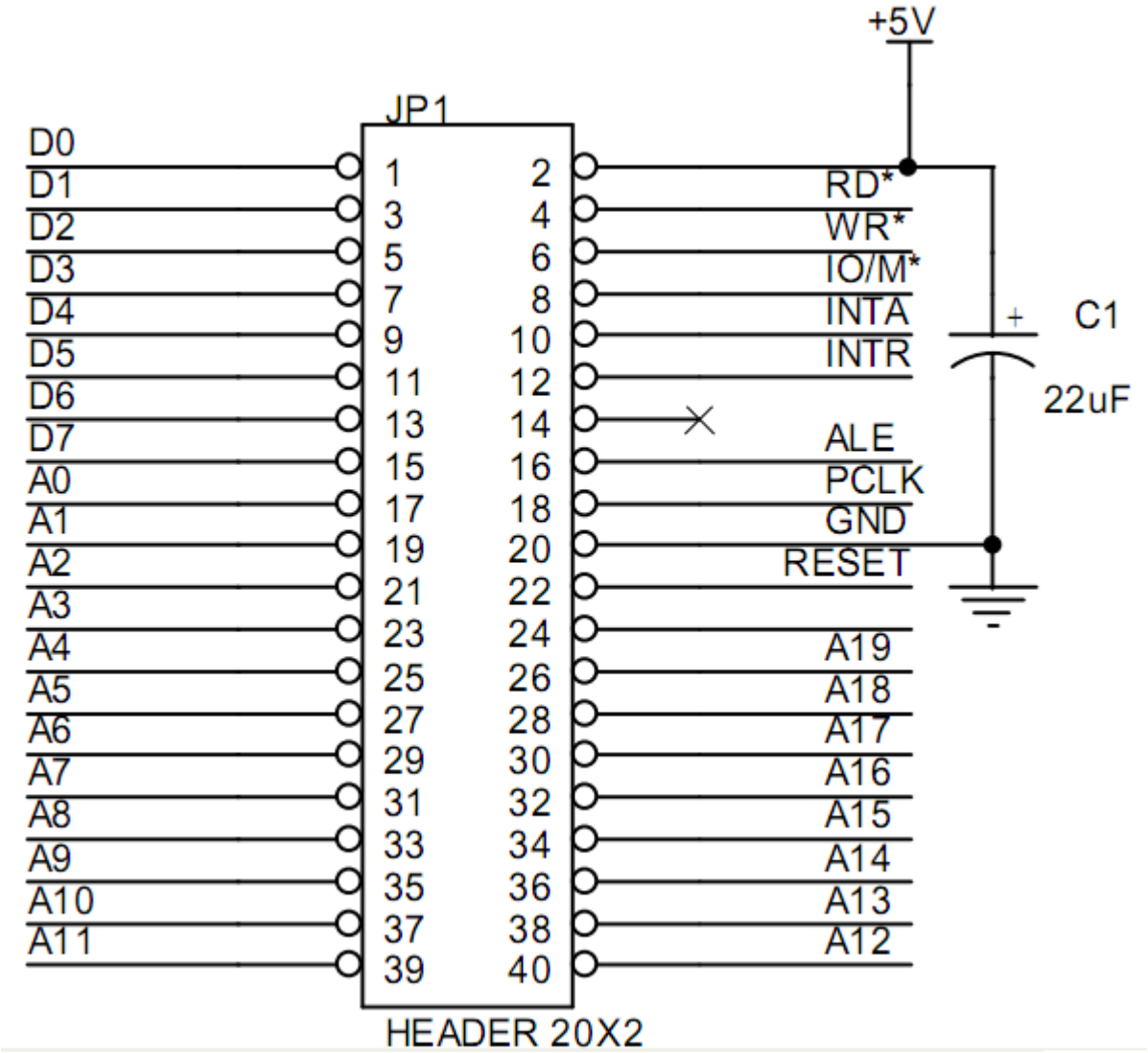
The kit will read the hex file, write to memory, when completed if no checksum error, the display will show no errors.



Press RESET then GO to run the program.

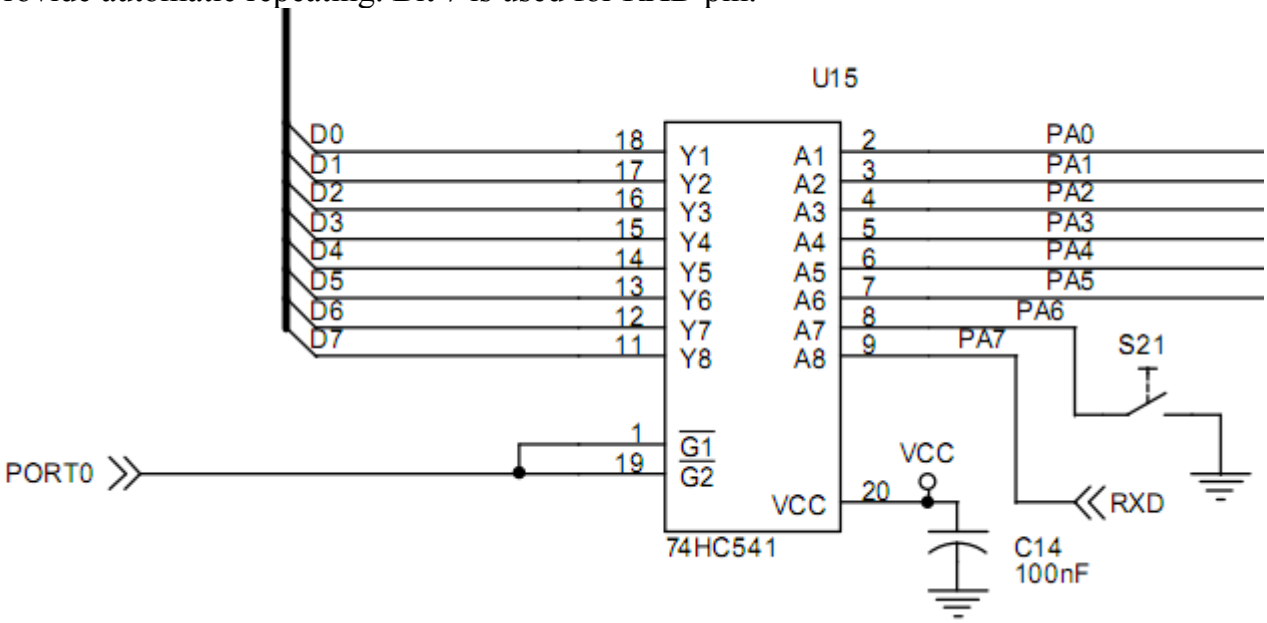
EXPANSION BUS HEADER

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.



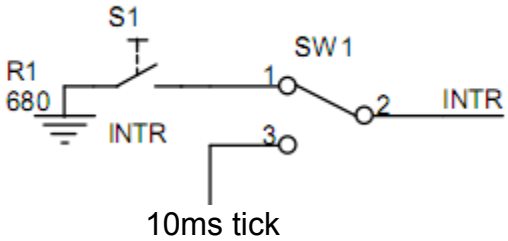
REP KEY

REP(repeat) key, S21 is one bit active low key switch connected to bit 6 of Port 0. To test the logic of S21, we can use instruction IN AL, DX and check bit 6 of the AL register with test bit instruction. REP key is used in monitor program together with key STEP, + or – to provide automatic repeating. Bit 7 is used for RXD pin.

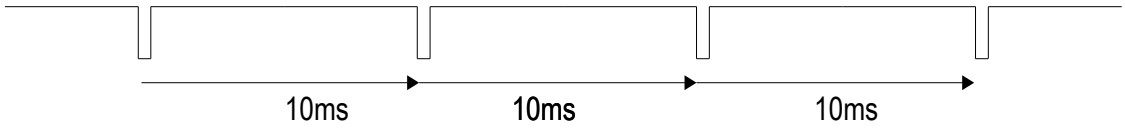


10ms TICK GENERATOR

SW1 is a selector for interrupt source between key INTR 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 D2 LED.



Tick is a 10ms periodic signal for triggering the 8088 INTR pin. When select SW1 to Tick, the 8088 CPU can be triggered by the external interrupt. The 100Hz tick or 10ms tick can be used to produce tasks that executed with multiple of tick.



Example program that uses 10ms tick is shown below.

We know that the x86 provides 256 interrupt vectors started at location 0000 to 03FF. The kit provides simple circuit that supplies the vector byte FF when the CPU request the vector byte on the data bus. Thus the vector location that stores IP and CS will be 3FD:3FC and 3FF:3FE.

The sample program will show the binary counting at one second rate or every 100 ticks. To test this code we must set SW1 to 10ms tick position.

```
03FC                org 3fch

03FC 00             dfb service_timer&0ffh
03FD 05             dfb  service_timer>>8

03FE 00             dfb 0
03FF 00             dfb 0

0400                org 400h

0400 FB             sti
0401 EBFE           jmp $

0500                org 500h

0500            service_timer
0500 FEC4           inc ah
0502 80FC64         cmp ah,100
0505 7506           jnz skip
0507 B400           mov ah,0
0509 FEC0           inc al
050B E600           out 0,al

050D CF         skip iret

0000                end
```

The service routine is located at 500H. It uses AH as the counter for checking if its contents is equal to 100 then clear it and increment AL register, write the AL to GPIO1 LED.

Main code is just enable the interrupt flag then JUMP here waiting for external interrupt from 10ms tick generator.

At location 3FC to 3FF we must insert the service routine's address with IP and CS.

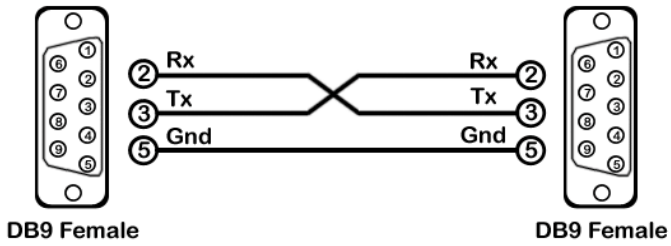
You can test this timer interrupt by entering the codes to the kit memory then start running with key GO at location 400.

What is happening at the gpio1 LED?

Can you change the counting rate? How?

RS232C PORT

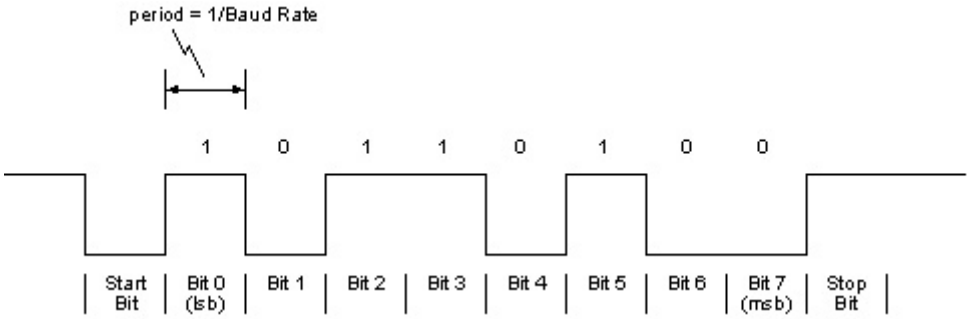
The RS232C port is for serial communication. We can use a cross cable or null MODEM cable to connect between the kit and terminal. The connector for both sides are DB9 female. We may build it or buying from computer stores.



For new PC or laptop computer without the RS232 port. It has only USB port, we may have the RS232C port by using the USB to RS232 converter.

DATA FRAME for UART COMMUNICATION

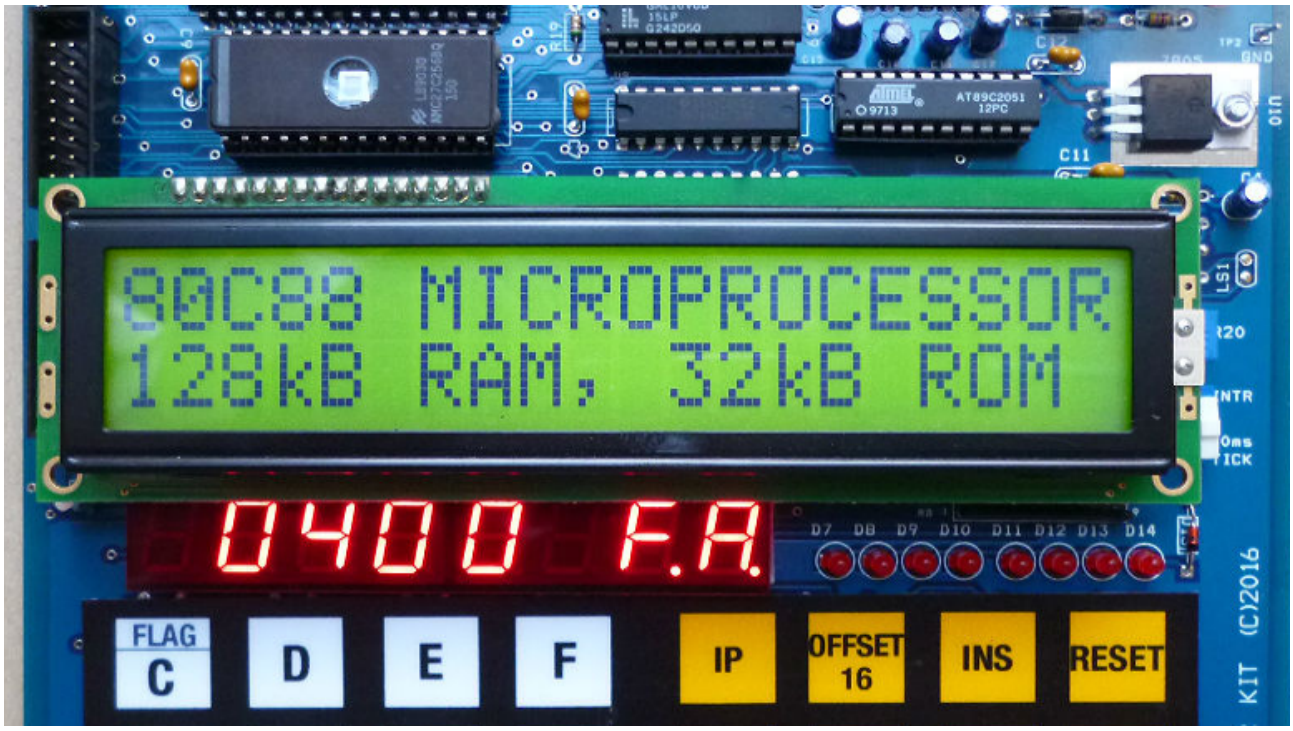
Serial data that communicated between kit and terminal is asynchronous format. The 68008 kit has no UART chip, instead it uses software controlled to produce bit rate of 2400 bit/s. The data frame is composed of start bit, 8-data bit and stop bit. For our kit, period = 1/2400 = 417 microseconds.



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

CONNECTING LCD MODULE

JR1 is 16-pin header for connecting the LCD module. The example shows connecting the 20x2 line text LCD module. R19 is a current limit resistor for back-light. R20 is trimmer POT for contrast adjustment. The LCD module is interfaced to the 8088 bus directly. The command and data registers are located in I/O space having address from 200H to 203H.



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

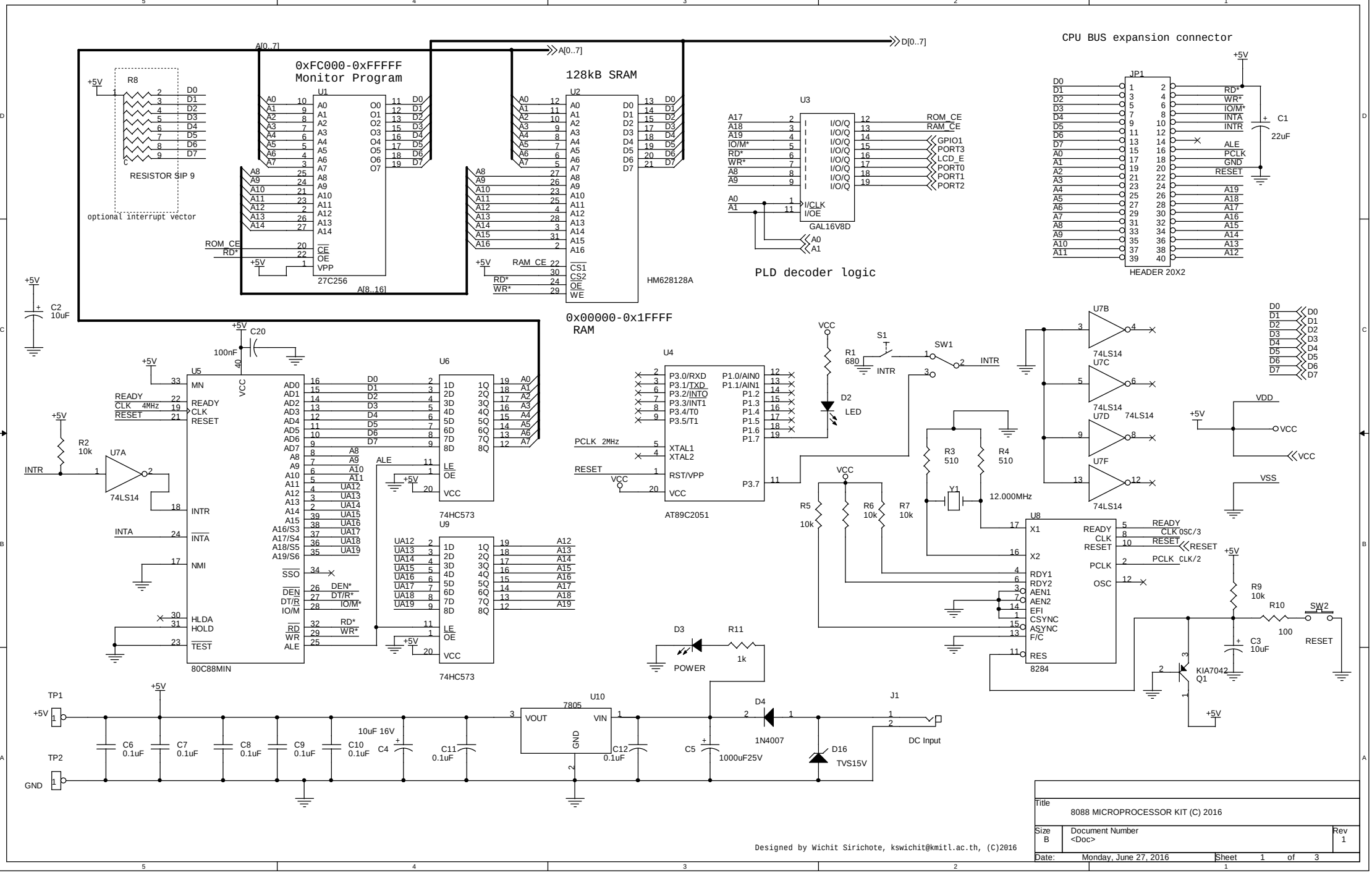
Text LCD module accepts ASCII codes for displaying the message on screen.
Without settings the LCD by software, no characters will be displayed. The first line will be black line by adjusting the R20 for contrast adjustment.

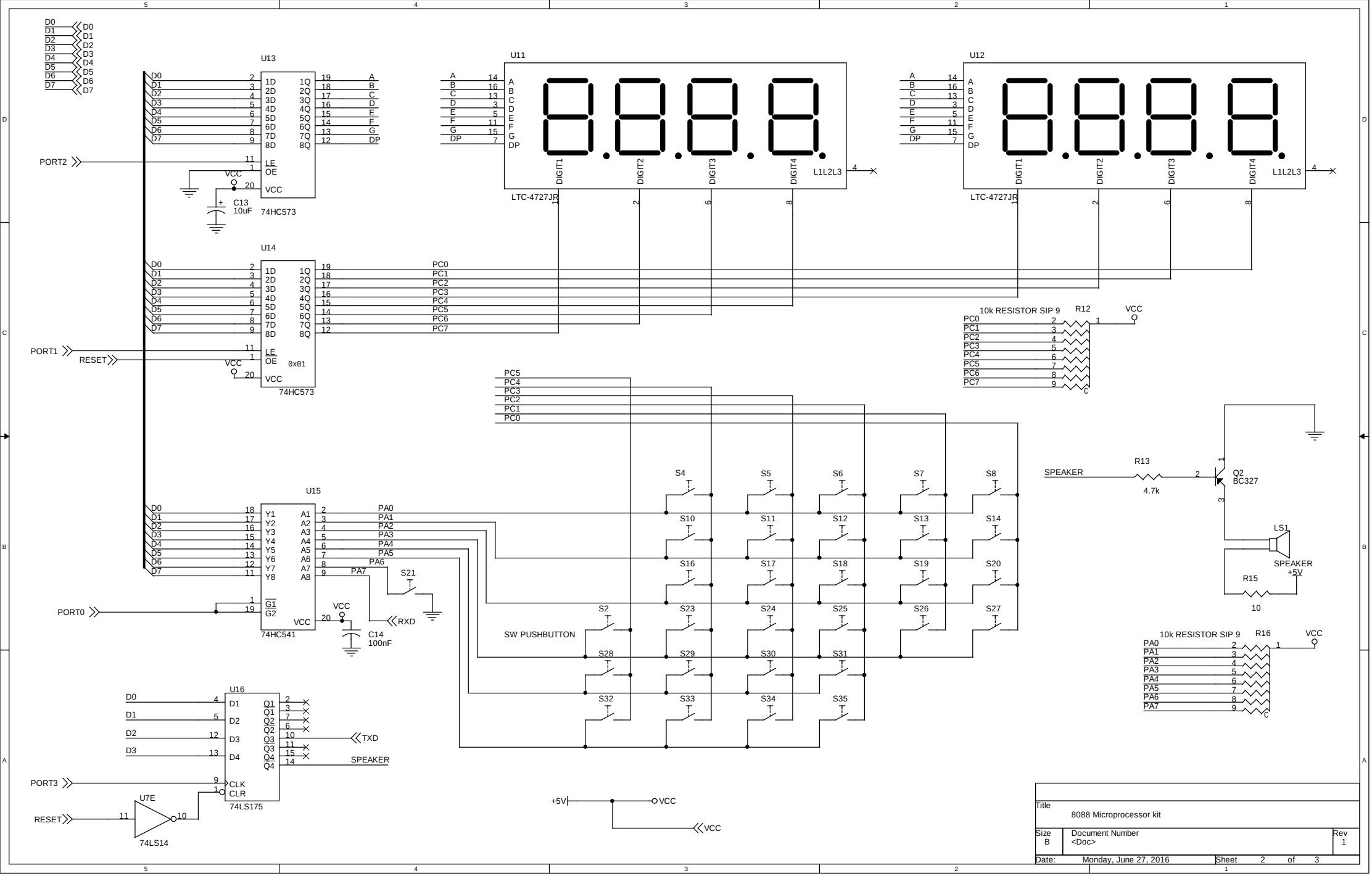
If the LCD module is connected, key TEST will write text to the LCD for testing.

LOGIC PROBE POWER SUPPLY

The kit provides test points TP1(+5V) and TP2(GND) for using the logic probe. Students may learn digital logic signals with logic probe easily. Tick signal is indicated by D2 LED blinking. Red clip is for +5V and Black clip for GND.

HARDWARE SCHEMATIC, PARTS LIST

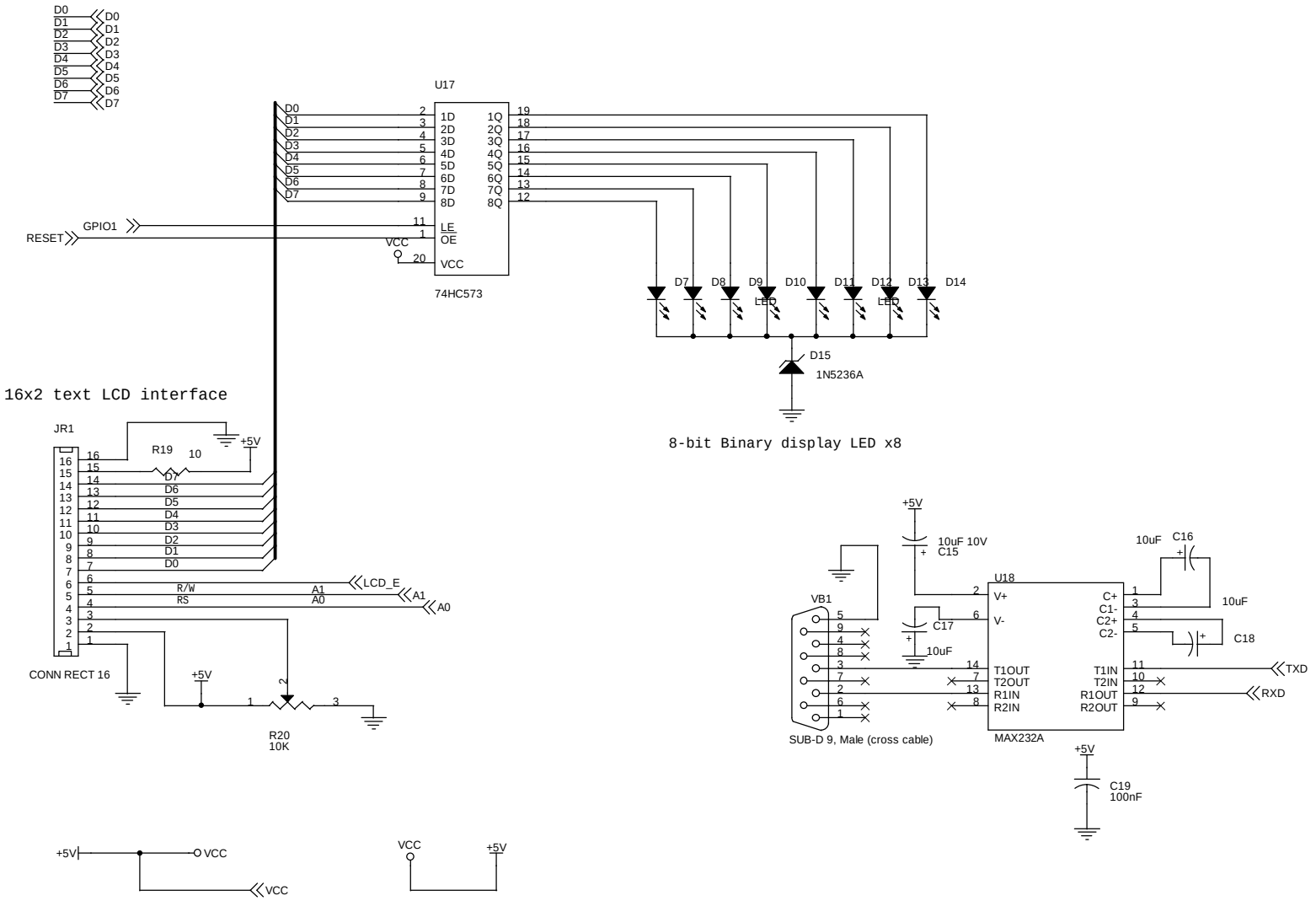

0xFC000-0xFFFFF
Monitor Program
U1
27C256
128kB SRAM
U2
HM628128A
0x00000-0x1FFFF
RAM
RESISTOR SIP 9
R8
optional interrupt vector
A[0..7]
A[8..16]
D[0..7]
ROM_CE
RAM_CE
RD*
WR*
CPU BUS expansion connector
JP1
HEADER 20X2
C1
22uF
U3
GAL16V8D
PLD decoder logic
ROM_CE
RAM_CE
GPIO1
PORT3
LCD_E
PORT0
PORT1
PORT2
C2
10uF
C20
100nF
U5
80C88MIN
U6
74HC573
U9
74HC573
U7A
74LS14
R2
10k
INTR
U4
AT89C2051
PCLK 2MHz
RESET
R1
680
S1
SW1
D2
LED
U7B
U7C
U7D
U7F
74LS14
VDD
VSS
R3
510
R4
510
Y1
12.000MHz
R5
10k
R6
10k
R7
10k
U8
8284
READY
CLK
RESET
PCLK
OSC
PCLK_CLK/2
R9
10k
R10
100
SW2
C3
10uF
KIA7042
Q1
D3
POWER
R11
1k
TP1
TP2
GND
C6
0.1uF
C7
0.1uF
C8
0.1uF
C9
0.1uF
C10
0.1uF
C4
10uF 16V
C11
0.1uF
U10
7805
VOUT
VIN
GND
C12
0.1uF
C5
1000uF25V
D4
1N4007
D16
TVS15V
J1
DC Input
Designed by Wichit Sirichote, kswichit@kmitl.ac.th, (C)2016
Title
8088 MICROPROCESSOR KIT (C) 2016
Size
B
Document Number
<Doc>
Rev
1
Date:
Monday, June 27, 2016
Sheet
1
of
3



Title
8088 Microprocessor kit

Size B | Document Number <Doc> | Rev 1

Date: Monday, June 27, 2016 | Sheet 2 of 3



Title	8088 Microprocessor kit	
Size B	Document Number <Doc>	Rev 1
Date: Monday, June 27, 2016	Sheet 3 of 3	

PARTS LIST

Semiconductors

U1 27C256 32kB EPROM
U2 HM628128A 128kB Static RAM
U3 GAL16V8D, programmable logic device
U4 AT89C2051, 8-bit microcontroller
U5 80C88, Harris 16-bit microprocessor
U6,U9,U13,U14,U17 74HC573
U7 74HC14
U8 8284, oscillator
U10 7805, voltage regulator
U12,U11 LTC-4727, 7-segment display
U15 74HC541
U16 74HC175
U18 MAX232A
D2,D7,D8,D9,D10,D11,D12, LED
D13,D14
D3 POWER
D4 1N4007
D15 1N5227A,
D16 TVS15V
Q1 KIA7042, voltage detector
Q2 BC557. PNP transistor

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

R1 680 Ohms
R2,R5,R6,R7,R9,R20 10K
R3,R4 510 Ohms
R8 RESISTOR SIP 9
R10 100 Ohms
R11 1k
R16,R12 10k RESISTOR SIP 9
R13 4.7k
R19,R15 10

Capacitors

C1 10uF electrolytic capacitor
C2,C3,C13,C16,C17,C18 10uF
C4 10uF 16V
C5 1000uF25V
C6,C7,C8,C9,C10 0.1uF
C11,C12 0.1uF
C14,C19,C20 100nF
C15 10uF

Additional parts

JP1 HEADER 20X2
JR1 CONN RECT 16
J1 DC Input
LS1 SPEAKER

SW1 SW MAG-SPDT
SW2 RESET
S1 INTR
S2 30 S2,S4,S5,S6,S7,S8,S10, SW PUSHBUTTON
S11,S12,S13,S14,S16,S17,
S18,S19,S20,S21,S23,S24,
S25,S26,S27,S28,S29,S30,
S31,S32,S33,S34,S35
TP1 +5V
TP2 GND

VB1 SUB-D 9, Male (cross cable)
Y1 12.000MHz

PCB double side plate through hole
LED color filter acrylic plastic
Keyboard sticker printable SVG file

MONITOR PROGRAM LISTINGS

```
                    ; MONITOR SOURCE CODE FOR 8088 MICROPROCESSOR KIT
                    ; COPYRIGHT (C) 2016 WICHIT SIRICHOTE
                    ;
                    ; MON88.ASM
                    ; ASSEMBLED WITH C32 CROSS ASSEMBLER


                    ;  9 SEPTEMBER 2015 CHANGE GPIO1'S LOCATION FROM 100H TO 00
                    ;                   USING GPIO1 WITH SIMPLE 8-BIT ADDRESS WILL BE EASIER
                    ;  30 MAY 2016      ADJUST BEEP FREQUENCY
                    ;  3 June 2016      change XTAL to 14.318MHz
                    ;  5 June 2016      test software UART at 4800 bit/s
                    ;  10 June 2016     change xtal to 12MHz
                    ;  15 June 2016     test software UART 2400 bit/s
                    ;  16 june 2016     add serial commands
                    ;  27 june 2016     add lcd drivers, test key (17H),offset16(18H),offset8
                    ;  28 june 2016     add offset calculation for 16-bit and 8-bit
                    ;  24 Aug 2016      add far call /ret for calling from lowest space


0000                      CPU     "8086.TBL"      ; CPU TABLE
0000                      HOF     "INT8"         ; HEX OUTPUT FORMAT


0100 =          PORT0   EQU 100H ; INPUT PORT, UART RXD BIT7, USER KEY S21
0001 =          PORT1   EQU 1  ; DIGIT CONTROL
0002 =          PORT2   EQU 2  ; SEGMENT
0003 =          PORT3   EQU 3  ; BI-COLOR LED, UART TXD BIT2, SPEAKER BIT3
0000 =          GPIO1   EQU 0   ; CHANGE LOCATION OF GPIO1 FROM 100 TO 0

0200 =          LCD             EQU 200H
0300 =          USER    EQU 300H
FE00 =          USER_STACK      EQU 0FE00H
FF00 =          SYSTEM_STACK    EQU 0FF00H

FF00 =          system_ram    equ 0ff00h       ;

001B =          Esc     equ 1bh
0000 =          eos equ 0
000D =          cr  equ 13
000A =          lf  equ 10

0000 =          command_write  equ 0
0002 =          command_read   equ 2
0001 =          data_write     equ 1
0003 =          data_read      equ 3
0080 =          busy           equ 80h



FF00                org system_ram

FF00            sram_pointer dfs 2      ; 16-bit pointer
FF02            bcs        dfs 1        ; byte check sum
FF03            bcs_error  dfs 1        ; byte check sum = 1 error

FF04            USER_FLAG   dfs 2
FF06            USER_IP     dfs 2
FF08            USER_CS     dfs 2
FF0A            user_ds     dfs 2
FF0C            user_es     dfs 2
FF0E            user_ss     dfs 2
FF10            user_sp     dfs 2


FF12            user_ax     dfs 2
FF14            user_bx     dfs 2
FF16            user_cx     dfs 2
FF18            user_dx     dfs 2
FF1A            user_bp     dfs 2
FF1C            user_si     dfs 2
FF1E            user_di     dfs 2

FF20            long_i     dfs 4        ; general 32-bit counter
FF24            long_j     dfs 4
FF28            long_k     dfs 4
```

```
FF2C            BUFFER     DFS 16       ; 16-BYTE BUFFER DISPLAY
FF3C            CURRENTAD  DFS 2        ; CURRENT DISPLAY ADDRESS

FF3E            STATE      DFS 1        ; DISPLAY STATE FOR ADDRESS AND DATA FILED

                ; STATE=3 OFFSET16 CALCULATION
                ; STATE=4 OFFSET8 CALCULATION


FF3F            COUNTER1   DFS 1        ; FOR ENTERING ADDRESS OR DATA MODE

FF40            SAVE_SYSTEM_STACK DFS 2
FF42            WARMCODE   DFS 2
FF44            command    dfs 1         ; serial command
FF45            flag1      dfs 1         ; user flag
                                       ; flag1.0 Space key was pressed
                                       ; flag1.1 Enter key was pressed
FF46            beep_flag dfs 1        ; beep/no beep

FF47            start_address dfs 2  ; for offset calculation
FF49            destination   dfs 2




C000                    org 0C000h

C000            start:
C000 FA          CLI                            ; DISABLE INTERRUPTS

C001 33C0                xor ax,ax
C003 8BF0                mov si,ax
C005 8BF8                mov di,ax

C007 B800FF              mov ax,SYSTEM_STACK       ;top_of_stack    ; set top of stack
C00A 8BE0                mov sp,ax


C00C BA0200      MOV DX,2
C00F B000        MOV AL,0
C011 EE          OUT DX,AL

C012 B0FF        MOV AL,0FFH
C014 BA0100      MOV DX,1
C017 EE          OUT DX,AL


                ; FILL INTERRUPT VECTORS TO POINT TO UNWANTED INTERRUPT SO THAT STRAY
                ; INTERRUPTS DO NOT CAUSE THE BOARD HANG


C018 BF0000      MOV     DI, 0                  ; START AT 0 (ASSUMES DS IS SET UP)
C01B B9FF00      MOV     CX, 255                ; DO 256 TIMES ; left the last vector fo
C01E            FILL_A:
C01E C7051BC9           MOV     word ptr [DI], UNWANTED_INT
C022 83C704      ADD     DI, 4                  ; FILL OFFSETS
C025 E2F7        LOOP    FILL_A

C027 BF0200      MOV     DI, 2                  ; START AT 2
C02A B9FF00      MOV     CX, 255                ; DO 256 TIMES
C02D            FILL_B:
C02D C7050000           MOV     word ptr [DI], 0000h
C031 83C704      ADD     DI, 4                  ; FILL SEGMENTS
C034 E2F7        LOOP    FILL_B


                ; INSERT VECTOR FOR INT 3 INSTRUCTION (WITH HEX CODE $CC)

C036 B81CC7      MOV AX,SERVICE_BREAK
C039 A30C00      MOV [000CH],AX ; SERVICE ADDRESS FOR INT 3
C03C B800F0      MOV AX,0F000H  ; CS SEGMENT FOR MONTIOR ROM
C03F A30E00      MOV [000EH],AX

                ; INSERT VECTOR FOR INT 1 TRAP

```

```
C042 B81CC7            MOV AX,SERVICE_BREAK
C045 A30400     MOV [0004H],AX ; SERVICE ADDRESS FOR INT 2
C048 B800F0     MOV AX,0F000H  ; CS SEGMENT FOR MONTIOR ROM
C04B A30600     MOV [0006H],AX


C04E B0FF       MOV AL,0FFH
C050 BA0000     MOV DX,GPIO1
C053 EE         OUT DX,AL

C054 E81006     CALL INIT
C057 E88E06     CALL CLR_DISPLAY

C05A B07F       MOV AL,7FH
C05C A232FF     MOV [BUFFER+6],AL
C05F B03F       MOV AL,3FH
C061 A231FF     MOV [BUFFER+5],AL
C064 B07F       MOV AL,7FH
C066 A230FF     MOV [BUFFER+4],AL
C069 A22FFF     MOV [BUFFER+3],AL


C06C A142FF     MOV AX,[WARMCODE]
C06F 3D99AA     CMP AX,0AA99H
C072 7410       JE WARM_BOOT
C074 B899AA     MOV AX,0AA99H
C077 A342FF     MOV [WARMCODE],AX

C07A B007       MOV AL,07 ; TEST BICOLOR LED
C07C E603       OUT PORT3,AL

C07E E8D106     CALL SHOW_MSG
C081 E87306     CALL BEEP

C084           WARM_BOOT

C084 BA0200     MOV  DX,2
C087 B04F       MOV     AL,4FH
                ;OUT DX,AL

C089 BA0100     MOV  DX,1
C08C B0FE       MOV  AL,0FEH
C08E EE         OUT     DX,AL

C08F B000       MOV     AL,0
C091 BA0000     MOV     DX,GPIO1
C094 EE         OUT     DX,AL   ; TURN OFF GPIO1 LED

C095 B004       MOV AL,04 ; TURN OFF BICOLOR LED
C097 E603       OUT PORT3,AL

                ;CALL READ_MEMORY


C099 BE2CFF    MAIN    MOV SI, BUFFER
C09C E87705            CALL SCAN1
C09F 80FCFF            CMP AH,-1
C0A2 7502       JNE CHK_REP
C0A4 EB17       JMP SKIP1

C0A6 BA0001    CHK_REP MOV DX,PORT0
C0A9 EC         IN AL,DX
C0AA 2440              AND AL,40H
C0AC 7402       JZ SKIP2
C0AE EBE9       JMP MAIN

                ; PUT SOME DELAY FOR REPEAT KEY THAT PRESSED
C0B0 B91E00    SKIP2   MOV CX,30

C0B3 51        SKIP3 PUSH CX
C0B4 BE2CFF     MOV SI,BUFFER
C0B7 E85C05     CALL SCAN1
C0BA 59         POP CX
C0BB E2F6       LOOP SKIP3
C0BD           SKIP1
C0BD E82B01     CALL DELAY    ; key released
```

```
C0C0 E82801     call delay

C0C3           UNTIL_PRESS

C0C3 BE2CFF     MOV SI,BUFFER
C0C6 E84D05     CALL SCAN1
C0C9 80FCFF     CMP AH,-1
C0CC 74F5       JE  UNTIL_PRESS

C0CE E81A01            CALL DELAY
C0D1 E82306     CALL BEEP

                ; CONVERT SCAN CODE TO INTERNAL CODE

C0D4 BE69CA     MOV SI,KEYTAB
C0D7 8AC4       MOV AL,AH
C0D9 B400       MOV AH,0
C0DB 03F0       ADD SI,AX
C0DD 2E         SEG CS
C0DE 8A04       MOV AL,[SI]

                ;out gpio1,al  ; check internal code

C0E0 3C10              CMP AL,10H
C0E2 7D05       JGE FUNCTIONS

                ; HEX KEY PRESSED

C0E4 E87302     CALL KEYHEX

C0E7 EBB0       jmp MAIN

                ; FUNCTIONS KEY PRESSED

C0E9           FUNCTIONS

C0E9 3C13       CMP AL,13H  ; KEY ADDRESS
C0EB 7505       JNE FUNC1
C0ED E85C02     CALL KEY_ADDRESS
C0F0 EBA7       JMP MAIN

C0F2 3C12      FUNC1   CMP AL,12H     ; KEY DATA
C0F4 7505       JNE FUNC2
C0F6 E84702     CALL KEY_DATA
C0F9 EB9E       JMP MAIN

C0FB 3C20      FUNC2   CMP AL,20H     ; KEY +
C0FD 7505              JNE FUNC3
C0FF E81802     CALL KEY_PLUS
C102 EB95       JMP MAIN

C104 3C19      FUNC3   CMP AL,19H    ; KEY -
C106 7505              JNE FUNC4
C108 E82202     CALL KEY_MINUS
C10B EB8C       JMP MAIN

C10D 3C10      FUNC4   CMP AL,10H; KEY IP
C10F 7505       JNE FUNC5
C111 E8F501     CALL KEY_IP
C114 EB83       JMP MAIN

C116 3C11      FUNC5 CMP AL,11H  ; KEY REG
C118 7506       JNE FUNC6
C11A E8D401     CALL KEY_REG
C11D E979FF     JMP MAIN

C120 3C21      FUNC6   CMP AL,21H; KEY GO
C122 7506       JNE FUNC7
C124 E84301     CALL KEY_GO
C127 E96FFF     JMP MAIN

C12A 3C22      FUNC7   CMP AL,22H; KEY STEP
C12C 7506              JNE FUNC8
C12E E88201     CALL KEY_STEP
C131 E965FF     JMP MAIN
```

```
C134 3C24      FUNC8   CMP AL,24H; KEY INSERT
C136 7506       JNE FUNC9
C138 E8CE00     CALL KEY_INS
C13B E95BFF     JMP MAIN

C13E 3C23      FUNC9   CMP AL,23H; KEY DELETE
C140 7506       JNE FUNC10
C142 E8AC00     CALL KEY_DEL
C145 E951FF     jmp main

C148 3C40      FUNC10  CMP AL,40H
C14A 7503              JNE FUNC11
C14C E87407     CALL TERMINAL

C14F 3C17      FUNC11  cmp al,17h
C151 7506              jne func12
C153 E88408     call test_key
C156 E940FF            jmp main

C159 3C18      func12  cmp al,18h             ; KEY OFFSET16
C15B 7506              jne func13
C15D E81100     call offset_16
C160 E936FF     jmp main

C163 3C1A      func13  CMP AL,1AH             ; KEY OFFSET8
C165 7506              JNE FUNC14
C167 E84400     CALL OFFSET_8
C16A E92CFF     JMP MAIN

C16D 90        FUNC14  NOP                   ; ADD MORE FUNCTIONS HERE

C16E E928FF     JMP MAIN




                ;*********************************************************************

C171 8B1E3CFF  offset_16 mov bx,[currentad]
C175 891E47FF            mov [start_address],bx

C179 B003         mov al,3
C17B A23EFF       mov [state],al    ; set state to 3 for offset 16 calculation

C17E B000         MOV AL,0
C180 A23FFF       MOV [COUNTER1],AL

C183 B040         mov al,40h
C185 A22DFF       mov [buffer+1],al
C188 B05E         mov al,5Eh
C18A A22CFF       mov [buffer],al

C18D A02FFF       MOV AL,[BUFFER+3]
C190 0C80         OR  AL,80H
C192 A22FFF       MOV [BUFFER+3],AL

C195 A030FF       MOV AL,[BUFFER+4]
C198 0C80         OR  AL,80H
C19A A230FF       MOV [BUFFER+4],AL

C19D A031FF       MOV AL,[BUFFER+5]
C1A0 0C80         OR  AL,80H
C1A2 A231FF       MOV [BUFFER+5],AL

C1A5 A032FF       MOV AL,[BUFFER+6]
C1A8 0C80         OR  AL,80H
C1AA A232FF       MOV [BUFFER+6],AL

C1AD C3           ret


C1AE 8B1E3CFF  OFFSET_8  mov bx,[currentad]
C1B2 891E47FF            mov [start_address],bx

C1B6 B004         mov al,4
```

```
C1B8 A23EFF       mov [state],al    ; set state to 4 for offset 8 calculation

C1BB B000         MOV AL,0
C1BD A23FFF       MOV [COUNTER1],AL

C1C0 B040         mov al,40h
C1C2 A22DFF       mov [buffer+1],al
C1C5 B05E         mov al,5Eh
C1C7 A22CFF       mov [buffer],al

C1CA A02FFF       MOV AL,[BUFFER+3]
C1CD 0C80         OR  AL,80H
C1CF A22FFF       MOV [BUFFER+3],AL

C1D2 A030FF       MOV AL,[BUFFER+4]
C1D5 0C80         OR  AL,80H
C1D7 A230FF       MOV [BUFFER+4],AL

C1DA A031FF       MOV AL,[BUFFER+5]
C1DD 0C80         OR  AL,80H
C1DF A231FF       MOV [BUFFER+5],AL

C1E2 A032FF       MOV AL,[BUFFER+6]
C1E5 0C80         OR  AL,80H
C1E7 A232FF       MOV [BUFFER+6],AL

C1EA C3           ret



C1EB B9F401    DELAY MOV CX,500
C1EE E2FE       LOOP $
C1F0 C3         RET

               ; DELETE BYTE AT THE CURRENT ADDRESS

C1F1 8B363CFF  KEY_DEL  MOV SI,[CURRENTAD]

C1F5 8BFE       MOV DI,SI
C1F7 81C70004   ADD DI,1024

C1FB 8A4401    DELETE   MOV AL,[SI+1]
C1FE 8804       MOV [SI],AL
C200 46         INC SI
C201 3BF7       CMP SI,DI
C203 75F6       JNE DELETE

C205 E89303     CALL READ_MEMORY
C208 C3         RET

               ; INSERT BYTE FROM THE CURRENT ADDRESS

C209 8B363CFF  KEY_INS  MOV SI,[CURRENTAD]
C20D 46         INC SI

C20E 8BFE       MOV DI,SI
C210 81C70004   ADD DI,1024

C214 8A45FF    INSERT1  MOV AL,[DI-1]
C217 8805       MOV [DI],AL
C219 4F         DEC DI
C21A 3BF7       CMP SI,DI
C21C 75F6       JNE INSERT1
C21E B000       MOV AL,0
```

```
C220 8805       MOV [DI],AL
C222 893E3CFF   MOV [CURRENTAD],DI
C226 E87203     CALL READ_MEMORY
C229 C3         RET


                ;*********************************************************************
C22A            COMPUTE_OFFSET16

C22A 8B163CFF   MOV DX,[CURRENTAD] ; GET DESTINATION ADDRESS
C22E 8B1E47FF   MOV BX,[START_ADDRESS] ;
C232 83C302     ADD BX,2
C235 2BD3       SUB DX,BX
C237 8B3E47FF   MOV DI,[START_ADDRESS]

C23B 8915       MOV [DI],DX

C23D 893E3CFF   MOV [CURRENTAD],DI
C241 B001       MOV AL,1
C243 A23EFF     MOV [STATE],AL
C246 E85203     CALL READ_MEMORY
C249 C3         RET


C24A            COMPUTE_OFFSET8

C24A 8B163CFF   MOV DX,[CURRENTAD] ; GET DESTINATION ADDRESS
C24E 8B1E47FF   MOV BX,[START_ADDRESS] ;
C252 83C301     ADD BX,1
C255 2BD3       SUB DX,BX
C257 8B3E47FF   MOV DI,[START_ADDRESS]
                                        ; HERE IS FOR OFFEST BYTE LOCATION
C25B 8815       MOV [DI],DL             ; WRITE ONLY LOW BYTE

C25D 893E3CFF   MOV [CURRENTAD],DI
C261 B001       MOV AL,1
C263 A23EFF     MOV [STATE],AL
C266 E83203     CALL READ_MEMORY
C269 C3         RET

                ;*********************************************************************


                ; JUMP FROM MONITOR PROGRAM TO USER PROGRAM WITH CS=0000
                ; STATE=3 COMPUTE OFFSET16
                ; STATE=4 COMPUTE OFFSET8

C26A A03EFF     KEY_GO  MOV AL,[STATE]
C26D 3C03               CMP AL,3
C26F 7504       JNE CHECK_GO4
C271 E8B6FF     CALL COMPUTE_OFFSET16
C274 C3         RET

C275            CHECK_GO4
C275 3C04               CMP AL,4
C277 7504       JNE CHECK_GO5
C279 E8CEFF     CALL COMPUTE_OFFSET8
C27C C3         RET

C27D            CHECK_GO5

C27D 8BC4               MOV AX,SP
C27F A340FF     MOV [SAVE_SYSTEM_STACK],AX

C282 A100FE     MOV AX,[USER_STACK]
C285 8BE0       MOV SP,AX           ; LOAD SP WITH USER STACK

C287 A108FF     MOV AX,[USER_CS]
C28A 50         PUSH AX
C28B A106FF     MOV AX,[USER_IP]
C28E 50         PUSH AX

C28F A11EFF     MOV AX,[USER_DI]
C292 50         PUSH AX
C293 A11CFF     MOV AX,[USER_SI]
C296 50         PUSH AX
```

```
C297 A11AFF     MOV AX,[USER_BP]
C29A 50         PUSH AX
C29B A118FF     MOV AX,[USER_DX]
C29E 50         PUSH AX
C29F A116FF     MOV AX,[USER_CX]
C2A2 50         PUSH AX
C2A3 A114FF     MOV AX,[USER_BX]
C2A6 50         PUSH AX
C2A7 A112FF     MOV AX,[USER_AX]
C2AA 50         PUSH AX



C2AB 58         POP AX
C2AC 5B         POP BX
C2AD 59         POP CX
C2AE 5A         POP DX
C2AF 5D         POP BP
C2B0 5E         POP SI
C2B1 5F         POP DI
C2B2 CB         RETF    ; JUMP TO USER PROGRAM WITH CS = 0000



                ; JUMP FROM MONITOR PROGRAM TO USER PROGRAM WITH CS=0000 and SET TRAP FLA

C2B3           KEY_STEP
C2B3 8BC4              MOV AX,SP
C2B5 A340FF     MOV [SAVE_SYSTEM_STACK],AX

C2B8 A100FE     MOV AX,[USER_STACK]
C2BB 8BE0       MOV SP,AX           ; LOAD SP WITH USER STACK

C2BD 9C         PUSHF

                ; SET TRAP FLAG

C2BE 8BEC              MOV BP,SP
C2C0 814E000001 OR WORD PTR[BP+0],100H

C2C5 A108FF     MOV AX,[USER_CS]
C2C8 50         PUSH AX
C2C9 A106FF     MOV AX,[USER_IP]
C2CC 50         PUSH AX

C2CD A11EFF     MOV AX,[USER_DI]
C2D0 50         PUSH AX
C2D1 A11CFF     MOV AX,[USER_SI]
C2D4 50         PUSH AX
C2D5 A11AFF     MOV AX,[USER_BP]
C2D8 50         PUSH AX
C2D9 A118FF     MOV AX,[USER_DX]
C2DC 50         PUSH AX
C2DD A116FF     MOV AX,[USER_CX]
C2E0 50         PUSH AX
C2E1 A114FF     MOV AX,[USER_BX]
C2E4 50         PUSH AX
C2E5 A112FF     MOV AX,[USER_AX]
C2E8 50         PUSH AX



C2E9 58         POP AX
C2EA 5B         POP BX
C2EB 59         POP CX
C2EC 5A         POP DX
C2ED 5D         POP BP
C2EE 5E         POP SI
C2EF 5F         POP DI
C2F0 CF         IRET    ; JUMP TO USER PROGRAM WITH CS = 0000 AND SET TRAP FLAG


                ; KEY REGISTERS

C2F1 B002      KEY_REG MOV AL,2
C2F3 A23EFF     MOV [STATE],AL
```

```
C2F6 E8EF03     CALL CLR_DISPLAY
C2F9 B050       MOV AL,50H
C2FB A231FF     MOV [BUFFER+5],AL
C2FE B079       MOV AL,79H
C300 A230FF     MOV [BUFFER+4],AL
C303 B06F       MOV AL,6FH
C305 A22FFF     MOV [BUFFER+3],AL
C308 C3         RET




C309 B000      KEY_IP  MOV AL,0
C30B A23EFF     MOV [STATE],AL
C30E 8B1E06FF   MOV BX,[USER_IP]
C312 891E3CFF   MOV [CURRENTAD],BX
C316 E88202     CALL READ_MEMORY
C319 C3         RET


C31A           KEY_PLUS
C31A B000              MOV AL,0
C31C A23EFF            MOV [STATE],AL
C31F A23FFF            MOV [COUNTER1],AL
C322 A13CFF            MOV AX,[CURRENTAD]
C325 40                INC AX
C326 A33CFF            MOV [CURRENTAD],AX
C329 E86F02            CALL READ_MEMORY
C32C C3                RET

C32D           KEY_MINUS
C32D B000              MOV AL,0
C32F A23EFF            MOV [STATE],AL
C332 A23FFF            MOV [COUNTER1],AL
C335 A13CFF            MOV AX,[CURRENTAD]
C338 48                DEC AX
C339 A33CFF            MOV [CURRENTAD],AX
C33C E85C02            CALL READ_MEMORY
C33F C3                RET


C340           KEY_DATA
C340 B000       MOV AL,0
C342 A23EFF     MOV [STATE],AL
C345 A23FFF     MOV [COUNTER1],AL
C348 E85002     CALL READ_MEMORY
C34B C3         RET



C34C           KEY_ADDRESS

C34C B001       MOV AL,1
C34E A23EFF     MOV [STATE],AL
C351 B000       MOV AL,0
C353 A23FFF     MOV [COUNTER1],AL

C356 E84202            CALL READ_MEMORY

C359 C3         RET


                ;----------------------------------KEYHEX--------------------------------

                ; ENTRY: AL = HEX KEY PRESSED

C35A 8A263EFF  KEYHEX   MOV AH,[STATE]
C35E 80FC01      CMP AH,1
C361 7504        JNE KEYHEX1
C363 E8BE01      CALL ENTER_ADDRESS
C366 C3          RET

C367 80FC00    KEYHEX1  CMP AH,0
C36A 7504        JNE KEYHEX2
C36C E88C01      CALL ENTER_DATA
C36F C3          RET
```

```

C370 80FC02    KEYHEX2  CMP AH,2
C373 7504        JNE KEYHEX3
C375 E81500      CALL REG_DISPLAY
C378 C3          RET

C379 80FC03    KEYHEX3  CMP AH,3
C37C 7504               JNE KEYHEX4
C37E E8CF01      CALL ENTER_DESTINATION
C381 C3          RET

C382 80FC04    KEYHEX4  CMP AH,4
C385 7504               JNE KEYHEX5
C387 E8C601      CALL ENTER_DESTINATION
C38A C3          RET

C38B 90        KEYHEX5  NOP


C38C C3                 RET

               ; REG KEY DISPLAY USER REGISTERS
               ; USE WITH HEX KEY

C38D 3C00      REG_DISPLAY CMP AL,0
C38F 7504           JNE REG1
C391 E86300         CALL DISPLAY_AX
C394 C3             RET

C395 3C01      REG1      CMP AL,1
C397 7504           JNE REG2
C399 E86F00         CALL DISPLAY_BX
C39C C3             RET

C39D 3C02      REG2      CMP AL,2
C39F 7504           JNE REG3
C3A1 E87B00         CALL DISPLAY_CX
C3A4 C3             RET

C3A5 3C03      REG3        CMP AL,3
C3A7 7504           JNE REG4
C3A9 E88700         CALL DISPLAY_DX
C3AC C3             RET

C3AD 3C04      REG4      CMP AL,4
C3AF 7504           JNE REG5
C3B1 E89300         CALL DISPLAY_SP
C3B4 C3             RET

C3B5 3C05      REG5      CMP AL,5
C3B7 7504           JNE REG6
C3B9 E89F00         CALL DISPLAY_BP
C3BC C3             RET

C3BD 3C06      REG6      CMP AL,6
C3BF 7504           JNE REG7
C3C1 E8AB00         CALL DISPLAY_SI
C3C4 C3             RET
C3C5 3C07      REG7      CMP AL,7
C3C7 7504           JNE REG8
C3C9 E8B700         CALL DISPLAY_DI
C3CC C3             RET

C3CD 3C08      REG8      CMP AL,8
C3CF 7504           JNE REG9
C3D1 E8C300         CALL DISPLAY_CS
C3D4 C3             RET

C3D5 3C09      REG9      CMP AL,9
C3D7 7504           JNE REG10
C3D9 E8CF00         CALL DISPLAY_DS
C3DC C3             RET

C3DD 3C0A      REG10     CMP AL,0AH
C3DF 7504                  JNE REG11
C3E1 E8DB00         CALL DISPLAY_SS
```

```
C3E4 C3             RET

C3E5 3C0B      REG11     CMP AL,0BH
C3E7 7504                  JNE REG12
C3E9 E8E700         CALL DISPLAY_ES
C3EC C3             RET

C3ED 3C0C      REG12       CMP AL,0CH
C3EF 7504                  JNE REG13
C3F1 E8F300         CALL DISPLAY_FLAG
C3F4 C3             RET

C3F5 90        REG13       NOP

C3F6 C3             RET



                ;-------------------------DISPLAY USER REGISTERS------------------------
C3F7 E8EE02    DISPLAY_AX  CALL CLR_DISPLAY
C3FA A112FF                MOV AX,[USER_AX]
C3FD E8AF02         CALL WORD2LED

C400 B077           MOV AL,77H
C402 A22DFF         MOV [BUFFER+1],AL
C405 B064           MOV AL,64H
C407 A22CFF         MOV [BUFFER],AL
C40A C3             RET

C40B E8DA02    DISPLAY_BX  CALL CLR_DISPLAY
C40E A114FF                MOV AX,[USER_BX]
C411 E89B02         CALL WORD2LED

C414 B07C           MOV AL,7CH
C416 A22DFF         MOV [BUFFER+1],AL
C419 B064           MOV AL,64H
C41B A22CFF         MOV [BUFFER],AL
C41E C3             RET

C41F E8C602    DISPLAY_CX  CALL CLR_DISPLAY
C422 A116FF                MOV AX,[USER_CX]
C425 E88702         CALL WORD2LED

C428 B039           MOV AL,39H
C42A A22DFF         MOV [BUFFER+1],AL
C42D B064           MOV AL,64H
C42F A22CFF         MOV [BUFFER],AL
C432 C3             RET

C433 E8B202    DISPLAY_DX  CALL CLR_DISPLAY
C436 A118FF                MOV AX,[USER_DX]
C439 E87302         CALL WORD2LED

C43C B05E           MOV AL,5EH
C43E A22DFF         MOV [BUFFER+1],AL
C441 B064           MOV AL,64H
C443 A22CFF         MOV [BUFFER],AL
C446 C3             RET

C447 E89E02    DISPLAY_SP  CALL CLR_DISPLAY
C44A A110FF                MOV AX,[USER_SP]
C44D E85F02         CALL WORD2LED

C450 B06D           MOV AL,6DH
C452 A22DFF         MOV [BUFFER+1],AL
C455 B073           MOV AL,73H
C457 A22CFF         MOV [BUFFER],AL
C45A C3             RET

C45B E88A02    DISPLAY_BP  CALL CLR_DISPLAY
C45E A11AFF                MOV AX,[USER_BP]
C461 E84B02         CALL WORD2LED

C464 B07C           MOV AL,7CH
C466 A22DFF         MOV [BUFFER+1],AL
C469 B073           MOV AL,73H
```

```
C46B A22CFF          MOV [BUFFER],AL
C46E C3              RET

C46F E87602    DISPLAY_SI  CALL CLR_DISPLAY
C472 A11CFF                MOV AX,[USER_SI]
C475 E83702          CALL WORD2LED

C478 B06D            MOV AL,6DH
C47A A22DFF          MOV [BUFFER+1],AL
C47D B030            MOV AL,30H
C47F A22CFF          MOV [BUFFER],AL
C482 C3              RET

C483 E86202    DISPLAY_DI  CALL CLR_DISPLAY
C486 A11EFF                MOV AX,[USER_DI]
C489 E82302          CALL WORD2LED

C48C B05E            MOV AL,5EH
C48E A22DFF          MOV [BUFFER+1],AL
C491 B030            MOV AL,30H
C493 A22CFF          MOV [BUFFER],AL
C496 C3              RET

C497 E84E02    DISPLAY_CS  CALL CLR_DISPLAY
C49A A108FF                MOV AX,[USER_CS]
C49D E80F02          CALL WORD2LED

C4A0 B039            MOV AL,39H
C4A2 A22DFF          MOV [BUFFER+1],AL
C4A5 B06D            MOV AL,6DH
C4A7 A22CFF          MOV [BUFFER],AL
C4AA C3              RET

C4AB E83A02    DISPLAY_DS  CALL CLR_DISPLAY
C4AE A10AFF                MOV AX,[USER_DS]
C4B1 E8FB01          CALL WORD2LED

C4B4 B05E            MOV AL,5EH
C4B6 A22DFF          MOV [BUFFER+1],AL
C4B9 B06D            MOV AL,6DH
C4BB A22CFF          MOV [BUFFER],AL
C4BE C3              RET

C4BF E82602    DISPLAY_SS  CALL CLR_DISPLAY
C4C2 A10EFF                MOV AX,[USER_SS]
C4C5 E8E701          CALL WORD2LED

C4C8 B06D            MOV AL,6DH
C4CA A22DFF          MOV [BUFFER+1],AL
C4CD B06D            MOV AL,6DH
C4CF A22CFF          MOV [BUFFER],AL
C4D2 C3              RET

C4D3 E81202    DISPLAY_ES  CALL CLR_DISPLAY
C4D6 A10CFF                MOV AX,[USER_ES]
C4D9 E8D301          CALL WORD2LED

C4DC B079            MOV AL,79H
C4DE A22DFF          MOV [BUFFER+1],AL
C4E1 B06D            MOV AL,6DH
C4E3 A22CFF          MOV [BUFFER],AL
C4E6 C3              RET

C4E7 E8FE01    DISPLAY_FLAG CALL CLR_DISPLAY
C4EA A104FF                MOV AX,[USER_FLAG]

C4ED E8BF01          CALL WORD2LED

C4F0 B071            MOV AL,71H
C4F2 A22DFF          MOV [BUFFER+1],AL
C4F5 B038            MOV AL,38H
C4F7 A22CFF          MOV [BUFFER],AL
C4FA C3              RET
```

```
                 ; ENTER DATA FIELD
                 ; SELECT FIRST 64KB RAM WITH DATA SEGMENT REGISTER
                 ;

C4FB 8A263FFF  ENTER_DATA  MOV AH,[COUNTER1]
C4FF 80FC00         CMP AH,0
C502 750E           JNZ SHIFT_DATA
C504 B401           MOV AH,1
C506 88263FFF       MOV [COUNTER1],AH

C50A 8B363CFF       MOV SI,[CURRENTAD]
C50E B400           MOV AH,0
C510 8824           MOV [SI],AH

C512 B104      SHIFT_DATA  MOV CL,4
C514 8B363CFF       MOV SI,[CURRENTAD]
C518 8A1C           MOV BL,[SI]
C51A D2E3           SAL BL,CL
C51C 0AD8           OR BL,AL
C51E 881C           MOV [SI],BL

C520 E87800         CALL READ_MEMORY
C523 C3             RET





                 ; ENTER ADDRESS FIELD

C524 8A263FFF  ENTER_ADDRESS   MOV AH,[COUNTER1]
C528 80FC00        CMP AH,0
C52B 750D          JNZ SHIFT_ADDRESS
C52D B401          MOV AH,1
C52F 88263FFF      MOV [COUNTER1],AH

C533 BB0000        MOV BX,0
C536 891E3CFF      MOV [CURRENTAD],BX

C53A B104      SHIFT_ADDRESS   MOV CL,4
C53C 8B1E3CFF      MOV BX,[CURRENTAD]
C540 D3E3          SAL BX,CL
C542 0AD8          OR  BL,AL
C544 891E3CFF      MOV [CURRENTAD],BX

C548 891E06FF      MOV [USER_IP],BX

C54C E84C00        CALL READ_MEMORY

C54F C3            RET

                 ; ENTER ADDRESS FIELD

C550 8A263FFF  ENTER_DESTINATION  MOV AH,[COUNTER1]
C554 80FC00        CMP AH,0
C557 750D          JNZ SHIFT_ADDRESS1
C559 B401          MOV AH,1
C55B 88263FFF      MOV [COUNTER1],AH

C55F BB0000        MOV BX,0
C562 891E3CFF      MOV [CURRENTAD],BX

C566 B104      SHIFT_ADDRESS1   MOV CL,4
```

```
C568 8B1E3CFF      MOV BX,[CURRENTAD]
C56C D3E3          SAL BX,CL
C56E 0AD8          OR  BL,AL
C570 891E3CFF      MOV [CURRENTAD],BX

                   ;MOV [USER_IP],BX

C574 A13CFF        MOV AX,[CURRENTAD]
C577 E83501             CALL WORD2LED

C57A A02FFF        MOV AL,[BUFFER+3]
C57D 0C80          OR  AL,80H
C57F A22FFF        MOV [BUFFER+3],AL

C582 A030FF        MOV AL,[BUFFER+4]
C585 0C80          OR  AL,80H
C587 A230FF        MOV [BUFFER+4],AL

C58A A031FF        MOV AL,[BUFFER+5]
C58D 0C80          OR  AL,80H
C58F A231FF        MOV [BUFFER+5],AL

C592 A032FF        MOV AL,[BUFFER+6]
C595 0C80          OR  AL,80H
C597 A232FF        MOV [BUFFER+6],AL

C59A C3            RET









                 ; GET CURRENT IP

C59B            READ_MEMORY
C59B A13CFF             MOV AX,[CURRENTAD]
C59E 8BF0        MOV SI,AX
C5A0 8A04        MOV AL,[SI]
C5A2 50          PUSH AX

C5A3 A13CFF      MOV AX,[CURRENTAD]
C5A6 E80601      CALL WORD2LED

C5A9 58          POP AX
C5AA E8E700      CALL BYTE2LED

C5AD A03EFF      MOV AL,[STATE]
C5B0 3C00        CMP AL,0
C5B2 7531        JNE MODE1

C5B4            MODE0         ; DATA DISPLAY
C5B4 A02FFF      MOV AL,[BUFFER+3]
C5B7 247F        AND  AL,7FH
C5B9 A22FFF      MOV [BUFFER+3],AL

C5BC A030FF      MOV AL,[BUFFER+4]
C5BF 247F        AND  AL,7FH
C5C1 A230FF      MOV [BUFFER+4],AL

C5C4 A031FF      MOV AL,[BUFFER+5]
C5C7 247F        AND  AL,7FH
C5C9 A231FF      MOV [BUFFER+5],AL

C5CC A032FF      MOV AL,[BUFFER+6]
C5CF 247F        AND  AL,7FH
C5D1 A232FF      MOV [BUFFER+6],AL

C5D4 A02CFF      MOV AL,[BUFFER]
C5D7 0C80        OR  AL,80H
```

```
C5D9 A22CFF     MOV [BUFFER],AL

C5DC A02DFF     MOV AL,[BUFFER+1]
C5DF 0C80       OR  AL,80H
C5E1 A22DFF     MOV [BUFFER+1],AL
C5E4 C3                RET

C5E5           MODE1
C5E5 A02FFF     MOV AL,[BUFFER+3]
C5E8 0C80       OR  AL,80H
C5EA A22FFF     MOV [BUFFER+3],AL

C5ED A030FF     MOV AL,[BUFFER+4]
C5F0 0C80       OR  AL,80H
C5F2 A230FF     MOV [BUFFER+4],AL

C5F5 A031FF     MOV AL,[BUFFER+5]
C5F8 0C80       OR  AL,80H
C5FA A231FF     MOV [BUFFER+5],AL

C5FD A032FF     MOV AL,[BUFFER+6]
C600 0C80       OR  AL,80H
C602 A232FF     MOV [BUFFER+6],AL

C605 A02CFF     MOV AL,[BUFFER]
C608 247F       AND  AL,7FH
C60A A22CFF     MOV [BUFFER],AL

C60D A02DFF     MOV AL,[BUFFER+1]
C610 247F       AND  AL,7FH
C612 A22DFF     MOV [BUFFER+1],AL
C615 C3         RET




                ; SCAN DISPLAY AND KEYPAD ONE CYCLE
                ; ENTRY: SI POINTED TO DISPLAY BUFFER
                ; EXIT: AH=KEY 0-35
                ;       AH=-1 NO KEY PRESSED

C616 B300      SCAN1    MOV BL,0
C618 B701       MOV BH,1
C61A B4FF       MOV AH,-1

C61C B108       MOV CL,8

C61E           KCOL
C61E 8AC7       MOV AL,BH
C620 F6D0       NOT AL


C622 E601       OUT PORT1,AL ; WRITE DIGIT

                      ; SEG CS      ; TEST READ FROM CODE SEGMENT
C624 8A04       MOV AL,[SI]  ; GET BUFFER

C626 E602       OUT PORT2,AL ; WRTE SEGEMENT

C628 247F       AND AL,7FH   ; MASK OFF DOT IF SET
C62A 3C06       CMP AL,6
C62C 7504       JNZ SKIP10
C62E B503       MOV CH,3
C630 EB02       JMP DELAY1

C632 B50A      SKIP10  MOV CH,10

C634 FECD      DELAY1  DEC CH
C636 75FC       JNZ DELAY1


C638 B000       MOV AL,0      ; TURN OFF LED
C63A E602       OUT PORT2,AL
```

```
C63C B51E        MOV CH,30
C63E FECD       DELAY3  DEC CH
C640 75FC        JNZ DELAY3


                ; NOW CHECK KEY PRESSED

C642 B506        MOV CH,6

C644 BA0001      MOV DX,PORT0
C647 EC          IN AL,DX

C648 D0D8       KROW    RCR  AL,1
C64A 7202        JC NOKEY

C64C 8AE3        MOV AH,BL

C64E FEC3       NOKEY   INC BL
C650 FECD        DEC CH
C652 75F4        JNZ KROW

C654 D0C7        ROL BH,1
C656 46          INC SI

C657 FEC9        DEC CL
C659 75C3        JNZ KCOL

                ; check serial port connection

C65B BA0001             mov dx,port0

C65E EC                 IN al,dx  ; CHECK received bit
C65F A880        TEST al,80H   ; TEST RXD BIT
C661 7502        JNZ no_serial      ; skip if no serial data

C663 B424        mov ah,36     ; scan code for serial data

C665 90         no_serial nop

C666 C3          RET

                ; INITIALIZE USER REGISTER

C667 B80004     INIT  MOV AX,0400H
C66A A33CFF      MOV [CURRENTAD],AX
C66D A306FF      MOV [USER_IP],AX

C670 B80000      MOV AX,0
C673 A308FF      MOV [USER_CS],AX
C676 A30AFF      MOV [USER_DS],AX
C679 A30EFF      MOV [USER_SS],AX
C67C A30CFF      MOV [USER_ES],AX

C67F A23FFF      MOV [COUNTER1],AL  ; CLEAR COUNTER1
C682 B800FE      MOV AX,USER_STACK
C685 A310FF      MOV [USER_SP],AX

C688 C3          RET

                ; CONVERT AL TO 7-SEGMENT PATTERN
                ; ENTRY: AL
                ; EXIT: AL

C689 BE49CA     NIBBLE2LED  MOV SI,CONVERT
C68C B400                   MOV AH,0
C68E 03F0            ADD SI,AX
C690 2E                     SEG CS
C691 8A04            MOV AL,[SI]
C693 C3              RET

                ; CONVERT BYTE TO 7-SEGMENT AND PUT TO DATA FIELD
                ; ENTRY: AL

C694 50         BYTE2LED    PUSH AX
C695 240F                   AND AL,0FH
C697 E8EFFF          CALL NIBBLE2LED
```

```
C69A A22CFF         MOV [BUFFER],AL

C69D 58             POP AX
C69E D0C8           ROR AL,1
C6A0 D0C8           ROR AL,1
C6A2 D0C8           ROR AL,1
C6A4 D0C8           ROR AL,1

C6A6 240F           AND AL,0FH

C6A8 E8DEFF         CALL NIBBLE2LED
C6AB A22DFF         MOV [BUFFER+1],AL
C6AE C3             RET

                ; CONVERT WORD TO 7-SEGMENT AND PUT TO ADDRESS FIELD
                ; ENTRY: AX

C6AF 50        WORD2LED   PUSH AX
C6B0 50            PUSH AX
C6B1 240F                  AND AL,0FH
C6B3 E8D3FF         CALL NIBBLE2LED
C6B6 A22FFF         MOV [BUFFER+3],AL

C6B9 58             POP AX
C6BA D0C8           ROR AL,1
C6BC D0C8           ROR AL,1
C6BE D0C8           ROR AL,1
C6C0 D0C8           ROR AL,1

C6C2 240F           AND AL,0FH

C6C4 E8C2FF         CALL NIBBLE2LED
C6C7 A230FF         MOV [BUFFER+4],AL

C6CA 58                    POP AX
C6CB 8AC4           MOV AL,AH

C6CD 50             PUSH AX

C6CE 240F           AND AL,0FH
C6D0 E8B6FF         CALL NIBBLE2LED
C6D3 A231FF         MOV [BUFFER+5],AL

C6D6 58             POP AX
C6D7 D0C8           ROR AL,1
C6D9 D0C8           ROR AL,1
C6DB D0C8           ROR AL,1
C6DD D0C8           ROR AL,1

C6DF 240F           AND AL,0FH

C6E1 E8A5FF         CALL NIBBLE2LED
C6E4 A232FF         MOV [BUFFER+6],AL

C6E7 C3             RET

                ; CLEAR BUFFER TO BLANK DISPLAY

C6E8 BE2CFF    CLR_DISPLAY MOV SI,BUFFER
C6EB B108           MOV CL,8
C6ED B000           MOV AL,0
C6EF 8804      CLEAR     MOV [SI],AL
C6F1 46             INC SI
C6F2 FEC9           DEC CL
C6F4 75F9           JNZ CLEAR
C6F6 C3             RET


                ; BEEP WHEN KEY PRESSED

C6F7 50        BEEP        PUSH AX
C6F8 BA0001         MOV DX,PORT0
C6FB EC             IN AL,DX
C6FC 2440           AND AL,40H
C6FE 7414           JZ NO_BEEP

```

```
C700 BB1E00         MOV BX,30

C703 B00C      BEEP1     MOV AL,0CH
C705 E603           OUT PORT3,AL
C707 E80C00         CALL BUZZ_DELAY
C70A B004           MOV AL,4
C70C E603           OUT PORT3,AL
C70E E80500         CALL BUZZ_DELAY

C711 4B             DEC BX
C712 75EF           JNZ BEEP1

C714           NO_BEEP
C714 58             POP AX
C715 C3             RET

C716 B99500    BUZZ_DELAY  MOV CX,95h     ; FOR 523Hz rom beep

                          ; mov cx,80h     ; test in ram beep
C719 E2FE           LOOP $
C71B C3             RET

               ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C71C           SERVICE_BREAK

C71C A312FF         MOV [USER_AX],AX
C71F 891E14FF       MOV [USER_BX],BX
C723 890E16FF       MOV [USER_CX],CX
C727 891618FF       MOV [USER_DX],DX
C72B 892E1AFF       MOV [USER_BP],BP
C72F 89361CFF       MOV [USER_SI],SI
C733 893E1EFF       MOV [USER_DI],DI

C737 58             POP AX
C738 A306FF         MOV [USER_IP],AX
C73B A33CFF         MOV [CURRENTAD],AX

C73E 58             POP AX
C73F A308FF         MOV [USER_CS],AX
C742 58             POP AX
C743 A304FF         MOV [USER_FLAG],AX
C746 892610FF       MOV [USER_SP],SP

C74A E84EFE         CALL READ_MEMORY

C74D 8B2640FF       MOV SP,[SAVE_SYSTEM_STACK]

C751 C3             RET

               ; DISPLAY COLD BOOT MESSAGE

C752 B91000    SHOW_MSG    MOV CX,16
C755 BE59CA         MOV SI,COLDMSG
C758 BF2CFF         MOV DI,BUFFER
C75B           SHOW1
C75B 2E             SEG CS
C75C 8A04           MOV AL,[SI]
C75E 8805           MOV [DI],AL
C760 46             INC SI
C761 47             INC DI

C762 E2F7           LOOP SHOW1


C764 B90800         MOV CX,8
C767 BE34FF         MOV SI,BUFFER+8

C76A 51        SHOW3     PUSH CX

C76B B93200         MOV CX,50

C76E 56        SHOW2     PUSH SI
C76F 51             PUSH CX
C770 E8A3FE         CALL SCAN1
C773 59             POP CX
```

```
C774 5E             POP SI
C775 E2F7           LOOP SHOW2

C777 4E             DEC SI

C778 59             POP CX
C779 E2EF           LOOP SHOW3

C77B C3             RET


                ;********************** software UART 2400 bit/s ************************

                ; SEND ASCII LETTER TO TERMINAL
                ; ENTRY: AL

C77C 50         COUT       push ax
C77D 51                    push cx
C77E 52            push dx

C77F 8AD0       SEND_BYTE: mov dl,al

C781 B000                 mov al,0
C783 E603                 OUT PORT3,al
C785 E82800               CALL BIT_DELAY   ; SEND START BIT

C788 B608                 mov dh,8

C78A 8AC2       CHK_BIT:  mov al,dl
C78C 2401          and al,1
C78E 7406                 jz SEND_ZERO

C790 B004                  mov al,4
C792 E603                  OUT port3,al
C794 EB06                 JMP NEXT_BIT

C796 B000       SEND_ZERO: mov al,0
C798 E603                  OUT port3,al
C79A EB00                  JMP NEXT_BIT

C79C E81100     NEXT_BIT:  CALL BIT_DELAY


C79F D0CA           ror dl,1

C7A1 FECE                  dec dh
C7A3 75E5                  JNZ CHK_BIT

C7A5 B004                  mov al,4
C7A7 E603                 OUT PORT3,al
C7A9 E80400               CALL BIT_DELAY  ; SEND STOP BIT

C7AC 5A            pop dx
C7AD 59            pop cx
C7AE 58            pop ax

C7AF C3                   RET

                ; BIT PERIOD FOR 2400 BIT/S

C7B0 B95400     BIT_DELAY   MOV CX,54h     ; found 1205Hz!  (1201x2=2402)
C7B3 E2FE           LOOP $
C7B5 C3             RET

                ; 1.5 bit delay

C7B6 B97E00     delay_1_5   MOV CX,7Eh     ; for 1.5 bit from start bit
C7B9 E2FE           LOOP $
C7BB C3             RET

                ; RECEIVE BYTE FROM 2400 BIT/S TERMINAL
                ; EXIT: AL

C7BC 90         cin        nop
C7BD BA0001     GET_BYTE   mov dx,port0
```

```
C7C0 EC        CIN1    IN al,dx  ; CHECK START BIT
C7C1 A880         TEST al,80H   ; TEST RXD BIT
C7C3 75FB         JNZ CIN1      ; REPEAT IF HIGH

C7C5 E8EEFF        CALL delay_1_5


C7C8 B707          Mov bh,7

C7CA B300          Mov bl,0


C7CC EC        CHK_BIT_RX IN al,dx
C7CD 2480                 AND AL,80H
                  ;  OUT 0,AL

C7CF 02C3          ADD AL,BL
C7D1 8AD8          MOV BL,AL

C7D3 D0CB          ROR BL,1

C7D5 E8D8FF        CALL BIT_DELAY

C7D8 FECF                 Dec bh
C7DA 75F0                 JNZ CHK_BIT_RX

C7DC E8D1FF        CALL BIT_DELAY    ; CENTER OF STOP BIT

C7DF 8AC3          MOV Al,BL           ; RETURN BYTE TO A
                  ; out 0,al

C7E1 C3            RET



                ; print hex
                ; entry: al

C7E2 50        out1x:       push ax
C7E3 240F            and al,0fh
C7E5 0430            add al,"0"
C7E7 3C39            cmp al,"9"
C7E9 7E02                    jle out1x1     ; if al less than or equal 39h then print it
C7EB 0407                    add al,7       ; else add with 7

C7ED E88CFF    out1x1:      call cout
C7F0 58              pop ax
C7F1 C3              ret

C7F2 51        out2x:       push cx
C7F3 B104                    mov cl,4
C7F5 D2C8                    ror al,cl     ; rotate right four bits
C7F7 E8E8FF          call out1x
C7FA D2C0                    rol al,cl     ; rotate left four bits
C7FC E8E3FF          call out1x
C7FF 59                      pop cx
C800 C3              ret

C801 50        out4x:       push ax
C802 86E0                    xchg ah,al
C804 E8EBFF                  call out2x
C807 58                      pop ax
C808 E8E7FF                  call out2x
C80B C3                      ret



C80C B020      space:       mov al," "
C80E E86BFF          call cout
C811 C3              ret

C812 B00D      newline:    mov al,cr
C814 E865FF         call cout
C817 B00A           mov al,lf
C819 E860FF         call cout
C81C C3             ret
```

```
                  ; convert ASCII letter to one nibble 0-F
                  ; 0-9 -> al-30
                  ; A-F -> al-7
                  ; entry: al
                  ; exit: al

C81D 2C30         to_hex:  sub al,"0"
C81F 3C0A                  cmp al,10
C821 7C04           jl zero_nine
C823 24DF                  and al,11011111b
C825 2C07           sub al,7
C827 C3           zero_nine: ret

                  ; read two ASCII bytes and convert them to one bye 8-bit data
                  ; exit: al

C828 51           get_hex: push cx
C829 53                    push bx

C82A E890FF                call get_byte
C82D E8EDFF         call to_hex
C830 D0C0           rol al,1
C832 D0C0           rol al,1
C834 D0C0           rol al,1
C836 D0C0           rol al,1
C838 8AE0           mov ah,al
C83A E880FF         call get_byte
C83D E8DDFF         call to_hex
C840 02C4           add al,ah

C842 5B             pop bx
C843 59             pop cx

C844 C3             ret



                  ; get Intel hex record and write to SRAM
                  ;


C845 E875FF       get_record: call get_byte
C848 3C1B                     cmp al,esc
C84A 7501                     jne is_colon?
C84C C3                       ret

C84D 3C3A         is_colon?:  cmp al,":"
C84F 75F4              jne get_record  ; wait until found begin of record

C851 32C0                     xor al,al
C853 A202FF                   mov [bcs],al         ; clear byte check sum

C856 B90000            mov cx,0   ; clear counter
C859 E8CCFF            call get_hex ; get number of byte
C85C 8AC8              mov cl,al    ; put to cl

C85E 000602FF                 add [bcs],al

C862 E8C3FF            call get_hex ; get destination address, put to bx register
C865 8AF8                     mov bh,al           ; save high byte

C867 000602FF                 add [bcs],al


C86B E8BAFF                   call get_hex
C86E 8AD8                     mov bl,al           ; and low byte

C870 000602FF                 add [bcs],al

C874 E8B1FF            call get_hex
C877 000602FF                 add [bcs],al

C87B 3C01                     cmp  al,1           ; end of record type is 01 ?
C87D 751E                     jne  data_record    ; jump if not 01

```

```
C87F E83BFF    wait_cr:    call get_byte
C882 3C0D                  cmp al,cr
C884 75F9                  jne wait_cr     ; until end of record sending! with lf detect


                          ; mov  al,0ffh        ; finish loading turn debug led off
                          ; mov  dx,p1ltch
C886 E600                  out  gpio1,al

C888 A003FF                mov al,[bcs_error]
C88B 3C01                  cmp al,1
C88D 7507                  jne no_error

C88F BED0CA                mov si,check_sum_error
C892 E87100                call pstr
C895 C3                    ret

C896 BEE3CA    no_error:   mov si,check_sum_ok
C899 E86A00                call pstr
C89C C3                    ret

C89D E888FF    data_record: call get_hex       ; get data byte
C8A0 8807                  mov [bx],al         ; save to SRAM at ds:[bx]

C8A2 000602FF              add [bcs],al

C8A6 43                    inc bx              ; next location

                           ;mov dx,port0       ; light debug led indicates loading is ru
C8A7 E600           out gpio1,al

C8A9 E2F2           loop data_record    ; until cx = 0

C8AB A002FF                mov al,[bcs]
C8AE F6D8                  neg al
C8B0 A202FF                mov [bcs],al
C8B3 E872FF                call get_hex        ; get check sum

C8B6 3A0602FF              cmp al,[bcs]

C8BA 7405                  je  record_correct

C8BC B001                  mov al,1
C8BE A203FF                mov [bcs_error],al    ; set byte check sum error flag

C8C1           record_correct:

C8C1 EB82                  jmp get_record  ; back to next record



                ;***************************** TERMINAL SERVICE ROUTINES ****************
C8C3 58         TERMINAL    pop ax    ; dummy pop


C8C4            serial_command

C8C4 E81900             call get_command
C8C7 E82300             call download
C8CA E80200       call key_enter

                ; add the serial commands here, see example of Intel hex download key

                ;        call display_memory





C8CD EBF5                jmp serial_command


                ; key enter
C8CF A044FF     key_enter mov al,[command]
C8D2 3C0D                 cmp al,13
```

```
C8D4 7506         jne exit_enter_key
C8D6 BE8ECA       mov si,title1
C8D9 E82A00       call pstr

C8DC           exit_enter_key
C8DC E83500       call send_prompt
C8DF C3           ret



C8E0 E8D9FE    get_command: call cin
C8E3 3C20                   cmp al,20h
C8E5 7202            jb control_key
C8E7 0C20            or al,20h    ; change to lower case letter

C8E9           control_key
C8E9 A244FF          mov [command],al
C8EC C3                     ret


                ; command execute
                ; get command from serial port

C8ED A044FF    download: mov al,[command]
C8F0 3C6C                cmp al,"l"
C8F2 750E                jnz  exit_download

                ; load intel hex file

C8F4 BEB7CA    load:     mov si,load_hex
C8F7 E80C00              call pstr

C8FA B000                mov al,0
C8FC A203FF              mov [bcs_error],al

C8FF E843FF              call get_record

C902           exit_download

C902 E80F00       call send_prompt
C905 C3                  ret

C906 2E        pstr:    seg cs          ; override data segment for SI
C907 8A04               mov al,[si]
C909 3C00        cmp al,eos
C90B 7501               jnz pstr1
C90D C3          ret

C90E           pstr1:
C90E E86BFE      call cout
C911 46          inc si
C912 EBF2               jmp pstr

C914 BEB2CA    send_prompt: mov si,prompt
C917 E8ECFF             call pstr
C91A C3                 ret



C91B           UNWANTED_INT:

C91B CF                 iret


                ; LCD DRIVER HD44780

C91C 50        LCD_READY       push ax
C91D BA0202                    MOV DX,LCD+COMMAND_READ
C920 EC        WAIT          IN AL,DX
C921 2480          AND AL,80H
C923 75FB          JNE WAIT  ; UNTIL READY
C925 58            pop ax
C926 C3            RET

C927 E8F2FF    clear_lcd:  call lcd_ready
```

```
C92A B001                  MOV AL,1
C92C BA0002         MOV DX,LCD+COMMAND_WRITE
C92F EE                    OUT DX,AL
C930 C3             RET

C931 E8E8FF    init_lcd: call lcd_ready
C934 BA0002              MOV DX,LCD+COMMAND_WRITE
C937 B038         MOV AL,38H
C939 EE                  OUT DX,AL

C93A E8DFFF              call lcd_ready
C93D BA0002       MOV DX,LCD+COMMAND_WRITE
C940 B00C                MOV AL,0ch
C942 EE                  OUT DX,AL
C943 E8E1FF              call clear_lcd

C946 C3                  ret

                ; print ASCII text on LCD
                ; entry: SI

C947           put_str_lcd:
C947 2E                   seg cs        ; need for rom placement
C948 8A04                 MOV AL,[SI]
C94A 3C00                 CMP AL,0
C94C 7501          JNE put_str_lcd1
C94E C3                   ret

C94F           put_str_lcd1:

C94F E8CAFF              call lcd_ref
C952 BA0102              MOV DX,LCD+DATA_WRITE
C955 EE           OUT DX,AL
C956 46           INC SI
C957 EBEE         JMP put_str_lcd

                ; goto_xy set cursor location on lcd
                ; entry: BX: BH = x, BL = y

C959 E8C0FF    goto_xy:  call lcd_ref
C95C 8AC3                mov al,bl
C95E 3C00                cmp al,0
C960 7509                jne goto_xy1
C962 8AC7                mov al,bh
C964 0480                add al,80h
C966 BA0002       mov dx,lcd+command_write
C969 EE                  out dx,al
C96A C3                  ret

C96B 3C01      goto_xy1: cmp al,1
C96D 7509                jne goto_xy2
C96F 8AC7                mov al,bh
C971 04C0                add al,0c0h
C973 BA0002       mov dx,lcd+command_write
C976 EE                  out dx,al
C977 C3                  ret

C978 3C02      goto_xy2: cmp al,2
C97A 7509                jne goto_xy3
C97C 8AC7                mov al,bh
C97E 0494                add al,094h
C980 BA0002       mov dx,lcd+command_write
C983 EE                  out dx,al
C984 C3                  ret

C985 3C03      goto_xy3: cmp al,3
C987 7509                jne goto_xy4
C989 8AC7                mov al,bh
C98B 04D4                add al,0d4h
C98D BA0002       mov dx,lcd+command_write
C990 EE                  out dx,al
C991 C3                  ret

C992 C3        goto_xy4: ret

                ; put_ch_lcd put character to lcd
```

```
                ; entry: AL

C993 E886FF    put_ch_lcd: call lcd_ready
C996 BA0102                mov dx,lcd+data_write
C999 EE             out dx,al
C99A C3                    ret


C99B E893FF    TEST_LCD  CALL INIT_LCD
C99E BEB1C9       MOV SI,TEXT
C9A1 E8A3FF       CALL PUT_STR_LCD
C9A4 BB0100       mov bx,0001
C9A7 E8AFFF       call goto_xy
C9AA BEC6C9       mov si,text2
C9AD E897FF       call put_str_lcd
C9B0 C3           ret

C9B1 3830433838TEXT      DFB "80C88 MICROPROCESSOR",0
C9C6 3132386B42text2     dfb "128kB RAM, 32kB ROM",0

C9DA E81AFD    test_key  call beep
C9DD B0FF                mov al,0ffh
C9DF E600         out gpio1,al
C9E1 E8B7FF       call test_lcd
C9E4 B000         mov al,0
C9E6 E600         out gpio1,al
C9E8 C3           ret


                ;********************** TEST CODE BEFORE ROM PROGRAMMING****************

C9E9 B055      testuart   mov al,55h
C9EB E88EFD               call cout
C9EE EBF9          jmp testuart


C9F0 E8CAFD    testecho   call get_byte
C9F3 E886FD               call cout
C9F6 EBF8          jmp testecho


C9F8 B000      UART      MOV AL,0
C9FA E603           OUT PORT3,AL
C9FC E80900         CALL UART_DELAY
C9FF B004           MOV AL,4
CA01 E603           OUT PORT3,AL
CA03 E80200         CALL UART_DELAY
CA06 EBF0           JMP UART

CA08 B93300    UART_DELAY  MOV CX,33h     ; FOR 523Hz 33h for 2
CA0B E2FE           LOOP $
CA0D C3             RET


CA0E B001      TESTCODE    MOV AL,1
CA10 D0C8      LOOP      ROR AL,1
CA12 E600           OUT GPIO1,AL
CA14 EBFA           JMP LOOP

CA16 E8A4FD    testcin     call get_byte
CA19 EBFB                  jmp testcin


CA1B B91000    dump        mov cx,16

CA1E 51        dump2     push cx

CA1F B91000                 mov cx,16


CA22 26        dump1:       seg es
CA23 8A07                   mov al,[bx]
CA25 E8CAFD          call out2x

CA28 8AC1                   mov al,cl
CA2A 3C09                   cmp al,16-7    ; backward couting!
```

```
CA2C 7507                   jne dump6
CA2E B02D                   mov al,"-"
CA30 E849FD                 call cout
CA33 EB03                   jmp dump7

CA35 E8D4FD    dump6:       call space

CA38 43        dump7:       inc bx
CA39 E2E7            loop dump1

CA3B E8D4FD          call newline

CA3E 59              pop cx

CA3F E2DD            loop dump2

CA41 C3              ret


CA42 E8E3FD    test_get_hex  call get_hex
CA45 E600             out gpio1,al
CA47 EBF9             jmp test_get_hex



                 ;************************************** CONSTANTS *********************

                 ; CONVERT HEX TO 7-SEGMENT PATTERN FOR 0-F

                 ;convert  dfb 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh,77h,7ch,39h,5eh,79h

CA49 3F          CONVERT   DFB 03FH   ; '0'
CA4A 06             DFB 006H   ; '1'
CA4B 5B             DFB 05BH   ; '2'
CA4C 4F             DFB 04FH
CA4D 66             DFB 066H
CA4E 6D             DFB 06DH
CA4F 7D             DFB 07DH
CA50 07             DFB 007H
CA51 7F             DFB 07FH
CA52 6F             DFB 06FH
CA53 77             DFB 077H
CA54 7C             DFB 07CH
CA55 39             DFB 039H
CA56 5E             DFB 05EH
CA57 79             DFB 079H
CA58 71             DFB 071H   ; 'F'

                 ; COLD BOOT MESSAGE

CA59 00          COLDMSG   DFB 0
CA5A 00             DFB 0
CA5B 00             DFB 0
CA5C 7F             DFB 07FH
CA5D 7F             DFB 07FH
CA5E 3F             DFB 03FH
CA5F 7F             DFB 07FH
CA60 00             DFB 0
CA61 00             DFB 0
CA62 00             DFB 0
CA63 00             DFB 0
CA64 00             DFB 0
```

```
CA65 00           DFB 0
CA66 00           DFB 0
CA67 00           DFB 0
CA68 00           DFB 0


                 ; Key-posistion-code to key-internal-code conversion table.

CA69             KEYTAB:
CA69 10          K0  DFB 10H ; IP
CA6A 11          K1  DFB 11H  ; REG
CA6B 12          K2  DFB 12H ; DATA
CA6C 13          K3  DFB 13H  ; ADDR
CA6D FF          K4  DFB 0FFH ; N/A
CA6E FF          K5  DFB 0FFH ; N/A
CA6F 0F          K6  DFB 0FH  ; F
CA70 0B          K7      DFB 0BH  ; B
CA71 07          K8      DFB 7    ; 7
CA72 03          K9      DFB 3    ; 3
CA73 FF          K10     DFB 0FFH ; N/A
CA74 FF          K11     DFB 0FFH ; N/A
CA75 0E          K12     DFB 0EH  ; KEY E
CA76 0A          K13     DFB 0AH  ; KEY A
CA77 06          K14     DFB 6    ; KEY 6
CA78 02          K15     DFB 2    ; KEY 2
CA79 20          K16     DFB 20H  ; KEY +
CA7A 21          K17     DFB 21H  ; KEY GO
CA7B 0D          K18     DFB 0DH  ; KEY D
CA7C 09          K19     DFB 9    ; KEY 9
CA7D 05          K20     DFB 5    ; KEY 5
CA7E 01          K21     DFB 1    ; KEY 1
CA7F 19          K22     DFB 19H  ; KEY -
CA80 22          K23     DFB 22H  ; KEY STEP
CA81 0C          K24     DFB 0CH  ; KEY C
CA82 08          K25     DFB 8    ; KEY 8
CA83 04          K26     DFB 4    ; KEY 4
CA84 00          K27     DFB 0    ; KEY 0
CA85 1A          K28     DFB 1AH  ; CBR
CA86 23          K29     DFB 23H  ; KEY DEL
CA87 14          K30     DFB  14H ; KEY COPY
CA88 15          K31     DFB  15H ; KEY REL
CA89 16          K32     DFB  16H  ; KEY SEND
CA8A 17          K33     DFB  17H  ; KEY LOAD
CA8B 18          K34     DFB  18H  ; KEY SBR
CA8C 24          K35     DFB  24H  ; KEY INS
CA8D 40          K36     DFB  40h  ; RXD code




                 ; string constants

CA8E 0D0A0A3830title1:         dfb cr,lf,lf,"8088 MICROPROCESSOR KIT (C) 2016",eos
CAB2 0D0A0A3E00prompt:    dfb cr,lf,lf,">",eos
CAB7 6C6F616420load_hex:  dfb "load Intel hex file...",cr,lf,eos
CAD0 0D0A636865check_sum_error: dfb cr,lf,"checksum errors!",eos
CAE3 0D0A6E6F20check_sum_ok:   dfb cr,lf,"no errors",eos



                 ;****** utilities subroutines that may called from very far location, mus
D000              org 0D000H

D000 E813F6      scan_display call scan1
D003 CB                 retf

D004 E8F0F6      beep_far     call beep
D007 CB                       retf

D008 E871F7      cout_far     call cout  ; send AL to UART 2400 bit/s
D00B CB                       retf

D00C E8ADF7      cin_far      call cin
D00F CB                       retf

D010 E881F6      byte2led_far call byte2led
```

```
D013 CB                     retf

D014 E898F6    word2led_far call word2led
D017 CB                     retf

D018 E816F9    init_lcd_far call init_lcd
D01B CB                     retf

D01C E874F9    put_ch_lcd_far call put_ch_lcd
D01F CB                       retf


               ;***************************reset vector FFFF0 **************************
FFF0                      org 0FFF0h

FFF0 EA00C000F0  JMP  FAR PTR start,0F000H ; jump to start




0000               end
```

```
FF02  BCS                FF03  BCS_ERROR          C6F7  BEEP
C703  BEEP1              D004  BEEP_FAR           FF46  BEEP_FLAG
C7B0  BIT_DELAY          FF2C  BUFFER             0080  BUSY
C716  BUZZ_DELAY         C694  BYTE2LED           D010  BYTE2LED_FAR
C275  CHECK_GO4          C27D  CHECK_GO5          CAD0  CHECK_SUM_ERROR
CAE3  CHECK_SUM_OK       C78A  CHK_BIT            C7CC  CHK_BIT_RX
C0A6  CHK_REP            C7BC  CIN                C7C0  CIN1
D00C  CIN_FAR            C6EF  CLEAR              C927  CLEAR_LCD
C6E8  CLR_DISPLAY        CA59  COLDMSG            FF44  COMMAND
0002  COMMAND_READ       0000  COMMAND_WRITE      C22A  COMPUTE_OFFSET16
C24A  COMPUTE_OFFSET8    C8E9  CONTROL_KEY        CA49  CONVERT
FF3F  COUNTER1           C77C  COUT               D008  COUT_FAR
000D  CR                 FF3C  CURRENTAD          0003  DATA_READ
C89D  DATA_RECORD        0001  DATA_WRITE         C1EB  DELAY
C634  DELAY1             C63E  DELAY3             C7B6  DELAY_1_5
C1FB  DELETE             FF49  DESTINATION        C3F7  DISPLAY_AX
C45B  DISPLAY_BP         C40B  DISPLAY_BX         C497  DISPLAY_CS
C41F  DISPLAY_CX         C483  DISPLAY_DI         C4AB  DISPLAY_DS
C433  DISPLAY_DX         C4D3  DISPLAY_ES         C4E7  DISPLAY_FLAG
C46F  DISPLAY_SI         C447  DISPLAY_SP         C4BF  DISPLAY_SS
C8ED  DOWNLOAD           CA1B  DUMP               CA22  DUMP1
CA1E  DUMP2              CA35  DUMP6              CA38  DUMP7
C524  ENTER_ADDRESS      C4FB  ENTER_DATA         C550  ENTER_DESTINATION
0000  EOS                001B  ESC                C902  EXIT_DOWNLOAD
C8DC  EXIT_ENTER_KEY     C01E  FILL_A             C02D  FILL_B
FF45  FLAG1              C0F2  FUNC1              C148  FUNC10
C14F  FUNC11             C159  FUNC12             C163  FUNC13
C16D  FUNC14             C0FB  FUNC2              C104  FUNC3
C10D  FUNC4              C116  FUNC5              C120  FUNC6
C12A  FUNC7              C134  FUNC8              C13E  FUNC9
C0E9  FUNCTIONS          C7BD  GET_BYTE           C8E0  GET_COMMAND
C828  GET_HEX            C845  GET_RECORD         C959  GOTO_XY
C96B  GOTO_XY1           C978  GOTO_XY2           C985  GOTO_XY3
C992  GOTO_XY4           0000  GPIO1              C667  INIT
C931  INIT_LCD           D018  INIT_LCD_FAR       C214  INSERT1
C84D  IS_COLON?          CA69  K0                 CA6A  K1
CA73  K10                CA74  K11                CA75  K12
CA76  K13                CA77  K14                CA78  K15
CA79  K16                CA7A  K17                CA7B  K18
CA7C  K19                CA6B  K2                 CA7D  K20
CA7E  K21                CA7F  K22                CA80  K23
CA81  K24                CA82  K25                CA83  K26
CA84  K27                CA85  K28                CA86  K29
CA6C  K3                 CA87  K30                CA88  K31
CA89  K32                CA8A  K33                CA8B  K34
CA8C  K35                CA8D  K36                CA6D  K4
CA6E  K5                 CA6F  K6                 CA70  K7
CA71  K8                 CA72  K9                 C61E  KCOL
C35A  KEYHEX             C367  KEYHEX1            C370  KEYHEX2
C379  KEYHEX3            C382  KEYHEX4            C38B  KEYHEX5
CA69  KEYTAB             C34C  KEY_ADDRESS        C340  KEY_DATA
C1F1  KEY_DEL            C8CF  KEY_ENTER          C26A  KEY_GO
C209  KEY_INS            C309  KEY_IP             C32D  KEY_MINUS
C31A  KEY_PLUS           C2F1  KEY_REG            C2B3  KEY_STEP
C648  KROW               0200  LCD                C91C  LCD_READY
000A  LF                 C8F4  LOAD               CAB7  LOAD_HEX
FF20  LONG_I             FF24  LONG_J             FF28  LONG_K
CA10  LOOP               C099  MAIN               C5B4  MODE0
C5E5  MODE1              C812  NEWLINE            C79C  NEXT_BIT
C689  NIBBLE2LED         C64E  NOKEY              C714  NO_BEEP
C896  NO_ERROR           C665  NO_SERIAL          C171  OFFSET_16
C1AE  OFFSET_8           C7E2  OUT1X              C7ED  OUT1X1
C7F2  OUT2X              C801  OUT4X              0100  PORT0
0001  PORT1              0002  PORT2              0003  PORT3
CAB2  PROMPT             C906  PSTR               C90E  PSTR1
C993  PUT_CH_LCD         D01C  PUT_CH_LCD_FAR     C947  PUT_STR_LCD
C94F  PUT_STR_LCD1       C59B  READ_MEMORY        C8C1  RECORD_CORRECT
C395  REG1               C3DD  REG10              C3E5  REG11
C3ED  REG12              C3F5  REG13              C39D  REG2
C3A5  REG3               C3AD  REG4               C3B5  REG5
C3BD  REG6               C3C5  REG7               C3CD  REG8
C3D5  REG9               C38D  REG_DISPLAY        FF40  SAVE_SYSTEM_STACK
C616  SCAN1              D000  SCAN_DISPLAY       C77F  SEND_BYTE
C914  SEND_PROMPT        C796  SEND_ZERO          C8C4  SERIAL_COMMAND
C71C  SERVICE_BREAK      C53A  SHIFT_ADDRESS      C566  SHIFT_ADDRESS1
```

```
C512  SHIFT_DATA         C75B  SHOW1              C76E  SHOW2
C76A  SHOW3              C752  SHOW_MSG           C0BD  SKIP1
C632  SKIP10             C0B0  SKIP2              C0B3  SKIP3
C80C  SPACE              FF00  SRAM_POINTER       C000  START
FF47  START_ADDRESS      FF3E  STATE              FF00  SYSTEM_RAM
FF00  SYSTEM_STACK       C8C3  TERMINAL           CA16  TESTCIN
CA0E  TESTCODE           C9F0  TESTECHO           C9E9  TESTUART
CA42  TEST_GET_HEX       C9DA  TEST_KEY           C99B  TEST_LCD
C9B1  TEXT               C9C6  TEXT2              CA8E  TITLE1
C81D  TO_HEX             C9F8  UART               CA08  UART_DELAY
C0C3  UNTIL_PRESS        C91B  UNWANTED_INT       0300  USER
FF12  USER_AX            FF1A  USER_BP            FF14  USER_BX
FF08  USER_CS            FF16  USER_CX            FF1E  USER_DI
FF0A  USER_DS            FF18  USER_DX            FF0C  USER_ES
FF04  USER_FLAG          FF06  USER_IP            FF1C  USER_SI
FF10  USER_SP            FF0E  USER_SS            FE00  USER_STACK
C920  WAIT               C87F  WAIT_CR            FF42  WARMCODE
C084  WARM_BOOT          C6AF  WORD2LED           D014  WORD2LED_FAR
C827  ZERO_NINE
```

NOTE