# Microcontroller (1) Lab Manual

**Part (3)**

 Prepared By:

Eng: Mohsen Ali AL-awami

Supervise red By:

Dr: Fadel AL-aqawa

2010-2011

LAB Expeirment (3)

* Main Topics:
* Switch Debouncing (port as I/O PROGRAMMER)
* TIMER / Counter PROGRAMMING
* Learning Objectives/Tasks:

Upon Compliation this expeirment ,you will be able to :

* Using the port pins as input throught the switchs
* List the timers mode of the 8051 and their associated registers.
* Describe the various modes of the 8051 timers
* Programs the 8051 timers to generate time delays
* Programs the 8051 counters as event counters

**Switch Debouncing (port as I/O PROGRAMMER)**

There are times that we need to access only 1 or 2 bits of the port instead of 8 bites ,a powerful features of the 8051 I/O ports is their capability to access individual bits of the port without altering the rest of the bits in the port.

**Example 1 :**

The following code toggles the bit p1.2 continuously .

Back : CPL P1.2 ;complement p1.2 only

ACALL DELAY

SJMP BACK

;another variation of the above program follows

AGAIN : SETB P1.2 ;change only p1.2=high

ACALL DELAY

CLR P1.2 ;change only p1.2=low

ACALL DELAY

SJMP AGAIN

**Example 2:**

Write a program to perform the following :

1. Keep mentoring the p1.2 bit until it become high
2. When p1.2 becomes high ,write value 45h to port 0
3. Send a high –to- low pulse to p2.3

Solution :

SETB P1.2 ; make switch p1.2 an input switch

MOV A,# 45H ; A=45h

AGAIN : JNB P1.2 ,AGAIN ;get out when p1.2 =1

Mov P0,A ;mov A to P0

SETB P2.3 ;make P2.3 high

CLR P2.3 ;make p2.3 low for H-TO-L

* In this program, instruction (JNB P1.2 AGAIN ) stays in the loop as long as p1.2 is low ,When p1.2 becomes high ,it gets out of the loop and write the value of 45H in to port 0 and creat H--TO –L pluse by the sequence of the instructions SETB and CLR .

**PROGRAMMING TIMERS**

* The 8051 has two timers, they can be used either as:
* Timers to generate a time delay or as
* Event counters to count events happening

outside the microcontroller

* Both Timer 0 and Timer 1 are 16 bits

wide

* Since 8051 has an 8-bit architecture, each 16-bits timer is accessed as two separate registers of low byte and high byte.

**Timer 0 & 1 Registers**

* Accessed as low byte and high byte
* The low byte register is called TL0/TL1

And The high byte register is called TH0/TH1

* Accessed like any other register
* MOV TL0,#4FH
* MOV R5,TH0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

------------------------------------------- ------------------------------

TH0 TL0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

------------------------------------------- ------------------------------

TH1 TL1

**TMOD Register**

* Both timers 0 and 1 use the same register, called TMOD (timer mode),set the various timer operation modes

TMOD is a 8-bit register

* The lower 4 bits are for Timer 0
* The upper 4 bits are for Timer 1
* In each case,
* The lower 2 bits are used to set the timer mode
* The upper 2 bits to specify the operation

|  |  |  |  |
| --- | --- | --- | --- |
| GATE | C/T | M1 | M0 |
| TIMER1 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| GATE | C/T | M1 | M0 |
| TIMER0 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| M1 | M0 | MODE | OPERATING MODE |
| 0 | 0 | 0 | **13-bit timer mode** 8-bit timer/counter THx with TLx as 5-bitprescaler. |
| 0 | 1 | 1 | **16-bit timer mode** 16-bit timer/counter THx and TLx arecascaded; there is no prescaler. |
| 1 | 0 | 2 | **8-bit auto reload** 8-bit auto reload timer/counter; THx holds avalue which is to be reloaded TLx each time  it overflows. |
| 1 | 1 | 3 | **Split timer mode** |

**GATE**: Gating control when set. Timer/counter is enable only while the INTx pin is high and the TRx control pin is set, When cleared the timer is enabled whenever the TRx control bit is set.

**C/T:** **Timer or counter selected, Cleared for timer operation (input from internal system clock),Set for counter operation (input from Tx input pin).**

**Example 3:**

Indicate which mode and which timer are selected for each of the following.

(a) MOV TMOD, #01H (b) MOV TMOD, #20H (c) MOV TMOD, #12H

**Solution:**

We convert the value from hex to binary. From Figure 9-3 we have:

(a) TMOD = 00000001, mode 1 of timer 0 is selected.

(b) TMOD = 00100000, mode 2 of timer 1 is selected.

(c) TMOD = 00010010, mode 2 of timer 0, and mode 1 of timer 1 are

selected.

**Example 4:**

Find the timer’s clock frequency and its period for various 8051-based system,

with the crystal frequency 11.0592 MHz when C/T bit of TMOD is 0.

**Solution:**

1/12 × 11.0529 MHz = 921.6 MHz;

T = 1/921.6 kHz = 1.085 us

% 12

XTAL

OSCILLATOR

* Timers of 8051 do starting and stopping by either software or hardware control
* In using software to start and stop the timer where GATE=0
* 􀂃 The start and stop of the timer are controlled by way of software by the TR (timer start) bits TR0 and TR1

1. The SETB instruction starts it, and it is stopped by the CLR instruction
2. These instructions start and stop the timers as long as GATE=0 in the TMOD register
3. The hardware way of starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register.

**Example 5:**

Find the value for TMOD if we want to program timer 0 in mode 2,use 8051 XTAL for the clock source, and use instructions to start and stop the timer.

TMOD = 0000 0010

**Mode 1 Programming**

The following are the characteristics and operations of mode1:

1. It is a 16-bit timer; therefore, it allows value of 0000 to FFFFH to be loaded into the timer’s register TL and TH

2. After TH and TL are loaded with a 16-bit

initial value, the timer must be started This is done by SETB TR0 for timer 0 and SETB TR1 for timer 1

3. After the timer is started, it starts to count up It counts up until it reaches its limit of FFFFH.

XTAL

OSCILLATOR

% 12

TF

TL

TH

TF goes high if Overfolw

C/T =0 TR FFFF 0000 flag

3. When it rolls over from FFFFH to 0000, it sets high a flag bit called TF (timer flag)

– Each timer has its own timer flag: TF0 for

timer 0, and TF1 for timer 1

– This timer flag can be monitored

􀂃 When this timer flag is raised, one option would be to stop the timer with the instructions CLR TR0 or CLR TR1, for timer 0 and timer 1, respectively

4. After the timer reaches its limit and rolls over, in order to repeat the process

􀂃 TH and TL must be reloaded with the original value, and TF must be reloaded to 0.

To generate a time delay

1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected

2. Load registers TL and TH with initial count value

3. Start the timer

4. Keep monitoring the timer flag (TF) with the JNB TFx,target instruction to see if it is raised

􀂃 Get out of the loop when TF becomes high

5. Stop the timer

6. Clear the TF flag for the next round

7. Go back to Step 2 to load TH and TL

Again.

**Example 6:**

In the following program, we create a square wave of 50% duty cycle (with

equal portions high and low) on the P1.5 bit. Timer 0 is used to generate the

time delay. Analyze the program

MOV TMOD,#01 ;Timer 0, mode 1(16-bit mode)

HERE: MOV TL0,#0F2H ;TL0=F2H, the low byte

MOV TH0,#0FFH ;TH0=FFH, the high byte

CPL P1.5 ;toggle P1.5

ACALL DELAY

SJMP HERE

In the above program notice the following step.

1. TMOD is loaded.

2. FFF2H is loaded into TH0-TL0.

3. P1.5 is toggled for the high and low portions of the pulse.

…

**Example 6: (cont’)**

DELAY:

SETB TR0 ;start the timer 0

AGAIN: JNB TF0,AGAIN ;monitor timer flag 0

;until it rolls over

CLR TR0 ;stop timer 0

CLR TF0 ;clear timer 0 flag

RET

4. The DELAY subroutine using the timer is called.

5. In the DELAY subroutine, timer 0 is started by the SETB TR0 instruction.

6. Timer 0 counts up with the passing of each clock, which is provided by the

crystal oscillator. As the timer counts up, it goes through the states of FFF3,

FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, and so on until it

reaches FFFFH. One more clock rolls it to 0, raising the timer flag (TF0=1).

At that point, the JNB instruction falls through.

FFF2  FFF3 FFF4 ……..>> FFFF 0000

TF=0 TF=0 TF=0 TF=0 TF=1

7. Timer 0 is stopped by the instruction CLR TR0. The DELAY subroutine

ends, and the process is repeated.

Notice that to repeat the process, we must reload the TL and TH registers, and

start the process is repeated

**Example 7;**

In Example 7, calculate the frequency of the square wave generated

on pin P1.5.

**Solution:**

In the timer delay calculation of Example 7, we did not include the

overhead due to instruction in the loop. To get a more accurate timing,

we need to add clock cycles due to this instructions in the loop. To do

that, we use the machine cycle, as shown below.

**Cycles**

HERE: MOV TL0,#0F2H 2

MOV TH0,#0FFH 2

CPL P1.5 1

ACALL DELAY 2

SJMP HERE 2

DELAY:

SETB TR0 1

AGAIN: JNB TF0,AGAIN 14

CLR TR0 1

CLR TF0 1

RET 2

**Total 28**

T = 2 × 28 × 1.085 us = 60.76 us and F = 16458.2 Hz

**Example 8:**

Find the delay generated by timer 0 in the following code, using both

of the Methods of Figure 9-4. Do not include the overhead due to

instruction.

CLR P2.3 ;Clear P2.3

MOV TMOD,#01 ;Timer 0, 16-bitmode

HERE: MOV TL0,#3EH ;TL0=3Eh, the low byte

MOV TH0,#0B8H ;TH0=B8H, the high byte

SETB P2.3 ;SET high timer 0

SETB TR0 ;Start the timer 0

AGAIN: JNB TF0,AGAIN ;Monitor timer flag 0

CLR TR0 ;Stop the timer 0

CLR TF0 ;Clear TF0 for next round

CLR P2.3

**Solution:**

(a) (FFFFH – B83E + 1) = 47C2H = 18370 in decimal and 18370 ×

1.085 us = 19.93145 ms

(b) Since TH – TL = B83EH = 47166 (in decimal) we have 65536 –

47166 = 18370. This means that the timer counts from B38EH to

FFFF. This plus Rolling over to 0 goes through a total of 18370 clock

cycles, where each clock is 1.085 us in duration. Therefore, we have

18370 × 1.085 us = 19.93145 ms as the width of the pulse.

**Example 8:**

The following program generates a square wave on P1.5 continuously

using timer 1 for a time delay. Find the frequency of the square

wave if XTAL = 11.0592 MHz. In your calculation do not

include the overhead due to Instructions in the loop.

MOV TMOD,#10 ;Timer 1, mod 1 (16-bitmode)

AGAIN: MOV TL1,#34H ;TL1=34H, low byte of timer

MOV TH1,#76H ;TH1=76H, high byte timer

SETB TR1 ;start the timer 1

BACK: JNB TF1,BACK ;till timer rolls over

CLR TR1 ;stop the timer 1

CPL P1.5 ;comp. p1. to get hi, lo

CLR TF1 ;clear timer flag 1

SJMP AGAIN ;is not auto-reload

**Solution:**

Since FFFFH – 7634H = 89CBH + 1 = 89CCH and 89CCH = 35276

clock count and 35276 × 1.085 us = 38.274 ms for half of the

square wave. The frequency = 13.064Hz.

Also notice that the high portion and low portion of the square wave

pulse are equal. In the above calculation, the overhead due to all

the instruction in the loop is not included.

LAB Expeirment (4)

* Main Topics:
* LCD Pins Descriptions
* LCD Commands
* LCD Operations
* Learning Objectives/Tasks:

Upon Compliation this expeirment ,you will be able to :

* List reasons that LCD are gaging wide-spread use,replacing LED’S
* Describe the function of the pins of atypical LCD
* List instruction command codes for programming an LCD
* Interface an LCD to the 8051
* Program an LCD by sending data or commands to it from 8051
* **INTERFACING LCD TO 8051:**

**8051 LCD**

**5v**

P0.1

P0.7

P2.0

P2.1

P2.3

D0 Vcc

Vee

D7 Vss

RS R/W E

**10k pot**

* **LCD Operation**

LCD is finding widespread use replacing LEDs

􀂾 The declining prices of LCD

􀂾 The ability to display numbers, characters, and graphics

􀂾 Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD

􀂾 Ease of programming for characters and graphics.

LCD Pin

Descriptions

* **LCD Pin Descriptions**

Pin Descriptions for LCD:

|  |  |  |  |
| --- | --- | --- | --- |
| pin | symbol | I/O | **Descriptions** |
| 1 | vss | -- | Ground |
| 2 | vcc | -- | +5V power supply |
| 3 | Vee | -- | Power supply to control contrast |
| 4 | Rs | I | RS=0 to select command register,  RS=1 to select data register |
| 5 | r/w | I | R/W=0 for write,R/W=1 for read |
| 6 | E | I/O | Enable |
| 7 | DB0 | I/O | The 8-bit data bus |

|  |  |  |  |
| --- | --- | --- | --- |
| pin | symbol | I/O | **Descriptions** |
| 8 | DB1 | I/O | The 8-bit data bus |
| 9 | DB2 | I/O | The 8-bit data bus |
| 10 | DB3 | I/O | The 8-bit data bus |
| 11 | DB4 | I/O | The 8-bit data bus |
| 12 | DB5 | I/O | The 8-bit data bus |
| 13 | DB6 | I/O | The 8-bit data bus |
| 14 | DB7 | I/O | The 8-bit data bus |

* LCD Command Codes:

|  |  |
| --- | --- |
| CODE(HEX) | Command to LCD Instruction Register |
| 1 | Clear display screen |
| 2 | Return home |
| 4 | Decrement cursor (shift cursor to left) |
| 6 | Increment cursor (shift cursor to right) |
| 5 | Shift display right |
| 7 | Shift display left |
| 8 | Display off, cursor off |
| A | Display off, cursor on |
| C | Display on, cursor off |

|  |  |
| --- | --- |
| CODE(HEX) | Command to LCD Instruction Register |
| E | Display on, cursor blinking |
| F | Display on, cursor blinking |
| 10 | Shift cursor position to left |
| 14 | Shift cursor position to right |
| 18 | Shift the entire display to the left |
| 1C | Shift the entire display to the right |
| 80 | Force cursor to beginning to 1st line |
| C0 | Force cursor to beginning to 2nd line |
| 38 | 2 lines and 5x7 matrix |

* Sending Codes and Data to LCDs w/ Time Delay:

To send any of the commands to the LCD, make pin RS=0. For data,

make RS=1. Then send a high-to-low pulse to the E pin to enable the

internal latch of the LCD. This is shown in the code below.

;calls a time delay before sending next data/command

;P1.0-P1.7 are connected to LCD data pins D0-D7

;P2.0 is connected to RS pin of LCD

;P2.1 is connected to R/W pin of LCD

;P2.2 is connected to E pin of LCD

ORG

MOV A,#38H ;INIT. LCD 2 LINES, 5X7 MATRIX

ACALL COMNWRT ;call command subroutine

ACALL DELAY ;give LCD some time

MOV A,#0EH ;display on, cursor on

ACALL COMNWRT ;call command subroutine

ACALL DELAY ;give LCD some time

MOV A,#01 ;clear LCD

ACALL COMNWRT ;call command subroutine

ACALL DELAY ;give LCD some time

MOV A,#06H ;shift cursor right

ACALL COMNWRT ;call command subroutine

ACALL DELAY ;give LCD some time

MOV A,#84H ;cursor at line 1, pos. 4

ACALL COMNWRT ;call command subroutine

ACALL DELAY ;give LCD some time

MOV A,#’N’ ;display letter N

ACALL DATAWRT ;call display subroutine

ACALL DELAY ;give LCD some time

MOV A,#’O’ ;display letter O

ACALL DATAWRT ;call display subroutine

AGAIN: SJMP AGAIN ;stay here

COMNWRT: ;send command to LCD

MOV P1,A ;copy reg A to port 1

CLR P2.0 ;RS=0 for command

CLR P2.1 ;R/W=0 for write

SETB P2.2 ;E=1 for high pulse

CLR P2.2 ;E=0 for H-to-L pulse

RET

DATAWRT: ;write data to LCD

MOV P1,A ;copy reg A to port 1

CLR P2.0 ;RS=0 for command

CLR P2.1 ;R/W=0 for write

SETB P2.2 ;E=1 for high pulse

CLR P2.2 ;E=0 for H-to-L pulse

RET

DELAY: MOV R3,#50 ;50 or higher for fast CPUs

HERE2: MOV R4,#255 ;R4 = 255

HERE: DJNZ R4,HERE ;stay until R4 becomes 0

DJNZ R3,HERE2

RET

END

* Sending Codes and Data to LCDs w/ Busy Flag:

;Check busy flag before sending data, command to LCD

;p1=data pin

;P2.0 connected to RS pin

;P2.1 connected to R/W pin

;P2.2 connected to E pin

ORG

MOV A,#38H ;init. LCD 2 lines ,5x7 matrix

ACALL COMMAND ;issue command

MOV A,#0EH ;LCD on, cursor on

ACALL COMMAND ;issue command

MOV A,#01H ;clear LCD command

ACALL COMMAND ;issue command

MOV A,#06H ;shift cursor right

ACALL COMMAND ;issue command

MOV A,#86H ;cursor: line 1, pos. 6

ACALL COMMAND ;command subroutine

MOV A,#’N’ ;display letter N

ACALL DATA\_DISPLAY

MOV A,#’O’ ;display letter O

ACALL DATA\_DISPLAY

HERE:SJMP HERE ;STAY HERE

COMMAND:

ACALL READY ;is LCD ready?

MOV P1,A ;issue command code

CLR P2.0 ;RS=0 for command

CLR P2.1 ;R/W=0 to write to LCD

SETB P2.2 ;E=1 for H-to-L pulse

CLR P2.2 ;E=0,latch in

RET

DATA\_DISPLAY:

ACALL READY ;is LCD ready?

MOV P1,A ;issue data

SETB P2.0 ;RS=1 for data

CLR P2.1 ;R/W =0 to write to LCD

SETB P2.2 ;E=1 for H-to-L pulse

CLR P2.2 ;E=0,latch in

RET

READY:

SETB P1.7 ;make P1.7 input port

CLR P2.0 ;RS=0 access command reg

SETB P2.1 ;R/W=1 read command reg

;read command reg and check busy flag

BACK:SETB P2.2 ;E=1 for H-to-L pulse

CLR P2.2 ;E=0 H-to-L pulse

JB P1.7,BACK ;stay until busy flag=0

RET

END