การทดลองที่ 5 การใช้งาน LCD Module และ Keyboard

วัตถุประสงค์

1. เพื่อให้นิสิตสามารถเขียนโปรแกรมภาษา C อย่างง่ายในการควบคุมไมโครคอนโทรลเลอร์ได้

2. เพื่อให้นิสิตเข้าใจการการใช้งาน LCD Module และ Keyboard

อุปกรณ์ในการทดลอง

1. ชุดทดลองไมโครคอนโทรลเลอร์ SILA-START-C51

2. เครื่องคอมพิวเตอร์ PC พร้อมโปรแกรมสำหรับการเขียนและคอมไพล์ภาษา C

- โปรแกรม Keil51 v.xx

3. สายต่อพอร์ตอนุกรม

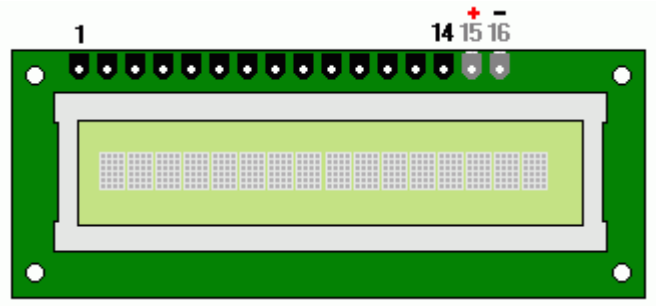
4. LCD Module

5. Keyboard 4x3

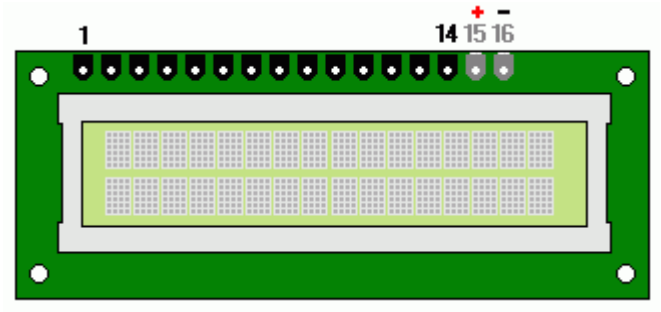
ทฤษฎี

การใช้งาน LCD Module

1). **ลักษณะและตำแหน่งของขา LCD โมดูลแต่ละแบบ**

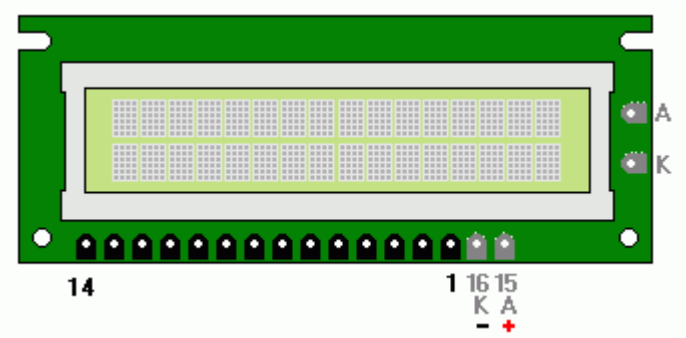


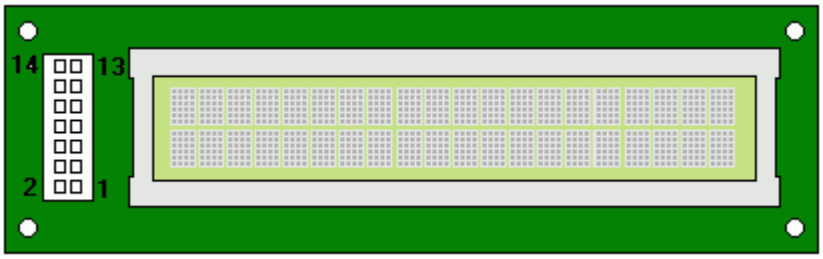
LCD 16x1 Line



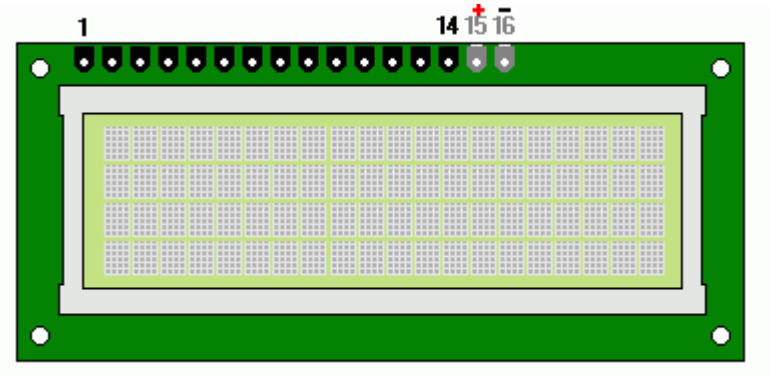
LCD 16x2 Line

LCD 16x2 Line





LCD 20x2 Line



LCD 20x4 Line

2). ตำแหน่งของขาและหน้าที่การใช้งานของ LCD โมดูล

Pin No.	Symbol	Description	Level	Function	
1	VSS	Ground	-	0V	Ground
2	VDD	Power Supply	-	+5V	ต่อกับแรงดันไฟเลี้ยง +5V
3	VO	LCD Control	-	-	ต่อกับแรงดันเพื่อปรับความเข้มของการแสดงผล
4	RS	Register Select	H/L	RS = 0 หมายถึงต้องการติดต่อกับรีจิสเตอร์คำสั่ง (Instruction Register) RS = 1 หมายถึงต้องการติดต่อกับรีจิสเตอร์ข้อมูล (Data Register)	
5	R/W	Read/Write	H/L	R/W = 0 หมายถึงต้องการเขียนข้อมูลไปยัง LCD โมดูล R/W = 1 หมายถึงต้องการอ่านข้อมูลจาก LCD โมดูล	
6	E	Enable	H, H->L	Enable Signal	
7 - 14	DB0-DB7	Data Bus	H/L	Data Bus Line	
15	A	Back Light A	-	Back Light +5V (สำหรับรุ่นที่มี Back Light)	
16	K	Back Light K	-	Back Light 0V (สำหรับรุ่นที่มี Back Light)	

3.) คำสั่งควบคุมการแสดงผลของ LCD โมดูล

ตารางที่1 คำสั่งควบคุมการแสดงผล LCD

Instruction		RS	R/W	Command Code (binary)								Description
				7	6	5	4	3	2	1	0	
1	Clear Display	0	0	0	0	0	0	0	0	0	1	Clear entire display and move cursor home (address 0)
2	Home Display	0	0	0	0	0	0	0	0	1	0	Move cursor home and return display to home position.
3	Entry Mode Set	0	0	0	0	0	0	0	1	M	S	Sets cursor direction (M: 0=left, 1=right) and display scrolling (S: 0=no scroll, 1=scroll)
4	Display/Cursor	0	0	0	0	0	0	1	D	C	B	Sets display on/off (D), cursor on/off (C) and blinking cursor (B). (0=off, 1=on)
5	Cursor or Display Shift	0	0	0	0	0	1	C	M	0	0	Cursor or Display Shift (C: 0=cursor, 1=display) left or right (M: 0=left, 1=right).
6	Function Set	0	0	0	0	1	DL	N	F	0	0	Data bus size (DL: 0=4-bits, 1=8-bits), lines No.(N: 0=1-line, 1=2-lines) and font size (F: 0=5x7, 1=5x10)
7	Set CG-RAM Address	0	0	0	1	CGRAM ADDRESS						Move pointer to Character Generator RAM location specified by address (ADDRESS)
8	Set DD-RAM Address	0	0	1	DDRAM ADDRESS							Move cursor to Display Data RAM location specified by address (ADDRESS)

9	Busy, ADD.Read	0	1	BF	ADDRESS	Read Busy flag, And Address Read
10	CGRAM,DDRAM WR	1	0	WRITE DATA		Write Data to DDRAM or CGRAM
11	CGRAM,DDRAM RD	1	1	READ DATA		Read Data to DDRAM or CGRAM

รายละเอียดคำสั่ง

1). **เคลียร์การแสดงผล** (Clear Display)

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

bit0=1 เคลียร์การแสดงผล

เคอร์เซอร์กลับไปอยู่ที่มุมซ้ายมือสุด

RS=0, R/W=0

2).Home Display

7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	-

bit1=1 เคอร์เซอร์กลับไปอยู่ที่มุมซ้ายมือสุด

ข้อมูลไม่มีการเปลี่ยนแปลง

RS=0, R/W=0

3).**โหมดการป้อนข้อมูล** (Entry Mode Set)

7	6	5	4	3	2	1	0
0	0	0	0	0	1	M	S

bit2=1

M (Address Increase/Decrease) , M=0 ลดตำแหน่งแอดเดรส, M=1 เพิ่มตำแหน่งแอดเดรส

RS=0, R/W=0

S (Shift bit) การเลื่อนข้อมูล,S=0 เคอร์เซอร์จะเลื่อนไปทางขวา ,S=1เคอร์เซอร์จะ

อยู่กับที่

4).การควบคุมการแสดงผล (Display/Cursor)

7	6	5	4	3	2	1	0
0	0	0	0	1	D	C	B

RS=0, R/W=0

bit3=1

D (Display ON/OFF) D=0 OFF, D=1 ON,

C (Cursor ON/OFF) C=0 OFF, C=1 ON,

B (Blinking Cursor ON/OFF) B=0 OFF, B=1 ON,

5).การควบคุมการเลื่อนเคอร์เซอร์ (Cursor or Display Shift)

7	6	5	4	3	2	1	0
0	0	0	1	C	M	-	-

RS=0, R/W=0

bit4=1

C (Cursor or Display Shift) C=0 shift cursor ,C=1 shift display

M (Move left/right) M=0 left, M=1 right,

6).ฟังก์ชั่นเซ็ท (Function Set)

7	6	5	4	3	2	1	0
0	0	1	D	N	F	-	-

RS=0, R/W=0

bit5=1

D (Data bus size) D= 0 is 4-bits,D= 1 is 8-bits,

N (lines No.)N= 0 is 1-line,N= 1 is 2-lines

F (font size) F= 0 is 5x7,F= 1 is 5x10

7).เซ็ทตำแหน่งใน CG-RAM (Set CG-RAM Address)

7	6	5	4	3	2	1	0
0	1	A5	A4	A3	A2	A1	A0

RS=0, R/W=0

bit6=1

หน่วยความจำชั่วคราว เก็บข้อมูลตัวอักษร CG-RAM (Character Generator RAM)

A0-A5 เป็นตำแหน่งแอดเดรสใน CG-RAM

8).เซ็ทตำแหน่งใน DD-RAM (Set DD-RAM Address)

7	6	5	4	3	□	1	0
1	A6	A5	A4	A3	A2	A1	A0

bit7=1

หน่วยความจำชั่วคราว เก็บข้อมูลการแสดง DD-RAM (Display Data RAM)

A0-A6 เป็นตำแหน่งแอดเดรสใน DD-RAM ซึ่งจะถูกคัดลอกไปยัง Address Counter (AC)

RS=0, R/W=0

DD-RAM คือหน่วยความจำที่เก็บข้อมูลการแสดงผล หากเขียนรหัส ASCII ลงในหน่วยความจำนี้ก็จะปรากฏที่จอ LCD ทันที ตำแหน่ง Address ของ LCD แต่ละแบบ

1 x 16 Display

Line 1	0	1	2	3	4	5	6	7	40	41	42	43	44	45	46	47

2 x 16 Display

Line 1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

4 x 20 Display

Line 1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
Line 3	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
Line 4	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D

เช่นกรณีใช้งาน LCD โมดูลแบบ 2x16 บรรทัดที่ 2 ใน columnแรกจะมีค่า address = 64(dec) หรือ 40H คำสั่งที่ใช้ในการเขียนข้อมูลออกไปยังโมดูล LCD บรรทัดต่างๆคือ การนำ 10000000 หรือ 80H มา OR กับ address ของ DDRAM เช่น 80H OR 40H =0C0H จะเป็นชุดคำสั่งที่ใช้เขียนไปยังโมดูล LCD บรรทัดที่ 2

9). การอ่าน BUSY Flag and Address Counter (BF and AC)

7	6	5	4	3	2	1	0
BF	A6	A5	A4	A3	A2	A1	A0

BF=bit7เป็นตัวบอกสถานะของ LCD

R/W=1กำหนดให้เป็น Read mode

RS=0, R/W=1

BF=0 ว่าง, BF=1 ไม่ว่าง

A0-A6 = Address Counter (AC)

10). การเขียนข้อมูลใน CG or DD-RAM

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

RS=1 กำหนดให้เป็นข้อมูล

R/W=0 กำหนดให้เป็น Write mode

RS=1, R/W=0

D0-D7 = ข้อมูลที่ต้องการเขียน

หากต้องการเขียนข้อมูลใน CG-RAM ให้เซ็ทตำแหน่ง CG-RAM ในข้อที่ 7 ก่อน

หากต้องการเขียนข้อมูลใน DD-RAM ให้เซ็ทตำแหน่ง DD-RAM ในข้อที่ 8 ก่อน

11). การอ่านข้อมูลจาก CG or DD-RAM

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

RS=1 กำหนดให้เป็นข้อมูล

R/W=1 กำหนดให้เป็น Read mode

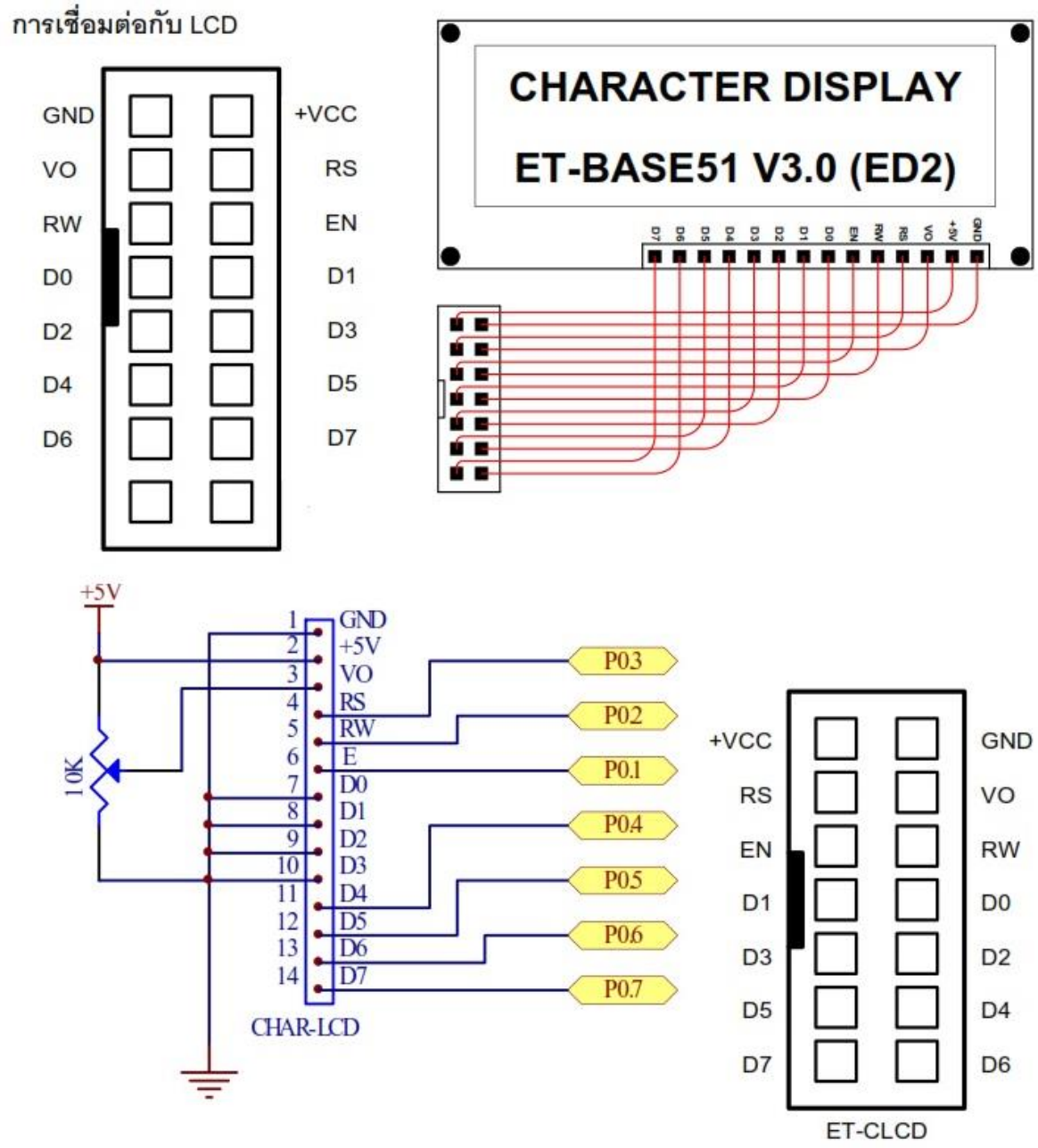
RS=1, R/W=0

D0-D7 =ข้อมูลที่อ่านได้

หากต้องการอ่านข้อมูลใน CG-RAM ให้เซ็ทตำแหน่ง CG-RAM ในข้อ 7 ก่อน

หากต้องการอ่านข้อมูลใน DD-RAM ให้เซ็ทตำแหน่ง DD-RAM ในข้อ 8 ก่อน

การเขียนโปรแกรมเพื่อควบคุม LCD



ขั้นตอนการเขียนโปรแกรมเพื่อใช้งาน LCD สามารถอธิบายได้ดังนี้

1) หากเป็นการเริ่มจ่ายไฟที่ระดับแรงดันถึง 4.5 V ให้ LCD ให้รออย่างน้อย 15ms เพื่อให้ LCD Reset ตัวเอง (Internal Reset)

2) Set ค่าเริ่มต้นต่างๆเพื่อให้ LCD เริ่มทำงานตามที่เราต้องการ โดย

กำหนดขาควบคุม

ให้ขา E = 1

ให้ขา RS= 0 กำหนดเป็นคำสั่ง

ให้ขา R/W = 0 เขียนคำสั่ง

ส่งข้อมูลคำสั่ง 4 ครั้ง

2.1) Function Set

Instruction = 0x28(00101000B) DL=0 4bit, N=1 2บรรทัด, F=0 5x7 dot

2.2) Display ON/OFF Control

Instruction = 0x0C (00001100B) D=1 Display ON, C=1 Cursor OFF, B=0 Blink OFF

2.3) Entry Mode Set

Instruction = 0x06 (00000110B) M=1 เพิ่มค่า DDRAM address ขึ้น, S=0 No scroll

2.4) Display clear

Instruction = 0x01 (00000001B)

3) Set DDRAM Address เพื่อเลือกตำแหน่งในการ Display โดยใช้ function gotolcd(unsigned char i)

กำหนดขาควบคุม

ให้ขา E = 1

ให้ขา RS= 0 กำหนดเป็นคำสั่ง

ให้ขา R/W = 0 เขียนคำสั่ง

เช่น

ถ้า i=0 คำสั่งที่ส่งไปคือ (00000000 OR 10000000)= 0x80 ตำแหน่งแรก แถวที่1

ถ้า i=0x40 คำสั่งที่ส่งไปคือ (01000000 OR 10000000)= 0xD0 ตำแหน่งแรก แถวที่ 2

4) เขียนตัวอักษรที่ต้องการแสดงไปยัง DDRAM

กำหนดขาควบคุม

ให้ขา E = 1

ให้ขา RS= 1 กำหนดเป็นข้อมูล

ให้ขา R/W = 0 เขียนข้อมูล

หมายเหตุ หลังจากการปฏิบัติการในแต่ละขั้นตอนต้องมีการตรวจสอบว่า LCD Module พร้อมที่จะรับคำสั่งต่อไปหรือไม่ โดยการตรวจสอบ Busy Flag (Bit 7) ใน Instruction Register โดย

กำหนดขาควบคุม

ให้ขา E = 1

ให้ขา RS= 0 กำหนดเป็นคำสั่ง

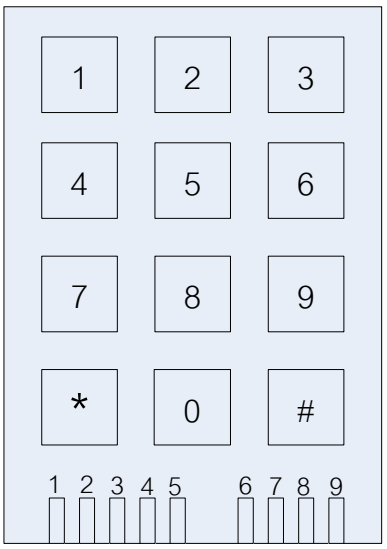
ให้ขา R/W = 1 อ่านคำสั่ง

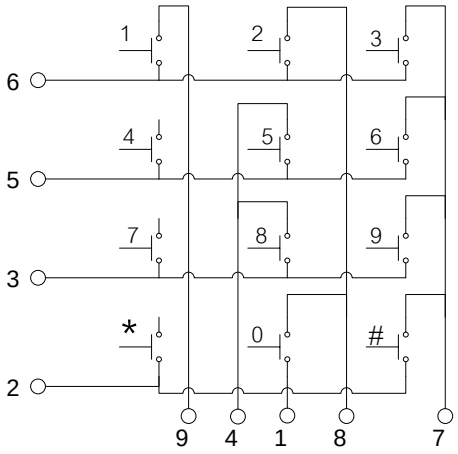
การใช้งาน Keyboard

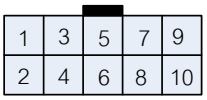
ในกรณีที่ต้องการที่รับค่าสถานะจาก Switch จำนวนมากนั้น สามารถทำได้โดยการเพิ่มจำนวนของ Input Port ให้มากขึ้นตาม อย่างไรก็ดี ในการเพิ่มจำนวนของ Input port จะทำให้ระบบมีราคาแพง การเพิ่มจำนวนของ Switch สามารถทำได้โดยใช้การทำงานของ Software ช่วยอาศัยหลักการพื้นฐานที่ว่า Microcomputer นั้น สามารถทำงานได้เร็วมากเมื่อเทียบกับการทำงานของ Switch ดังนั้นแล้ว Microcomputer ก็ไม่จำเป็นต้องรับค่าของ Switch ทั้งหมดพร้อมกัน โดยเลือกรับ Input ทีละตัว หรือทีละชุด ก็ได้ จนครบจำนวนของ Switch ทั้งหมด

ในการทดลอง เป็นการต่อ Switch แบบ Matrix ซึ่ง Port 3 บิต P3.3-P3.6 ถูกใช้ในการเลือกว่าจะอ่านค่าสถานะของ Switch ในแถวใด และ Port 3 บิต P3.0-P3.2 เป็นการเลือกว่าจะอ่านค่าสถานะของ Switch ในหลักใด การเขียนโปรแกรมเพื่อรับค่าสถานะของ Switch ทั้งหมดสามารถแสดงเป็น Pseudo code ได้ดังนี้

1. ตรวจสอบการกดของ column แรกโดยส่ง 1111 1110B
2. ถ้า column แรก ถูกกด ให้ค่าข้อมูลการกด key เป็น 1
    2.1 ส่งข้อมูล ‘1’ ออกไปที่ทุกแถว (1111 0111B) แล้วอ่านค่าเข้ามา ถ้าบิตที่เป็นแถว 1 เป็น ‘0’ บวกค่า 0 เข้ากับค่าข้อมูลการกด key ถ้าบิตที่เป็นแถว 2 เป็น ‘0’ บวกค่า 3 เข้ากับค่าข้อมูลการกด key ถ้าบิตที่เป็นแถว 3 เป็น ‘0’ บวกค่า 6 เข้ากับค่าข้อมูลการกด key ถ้าบิตที่เป็นแถว 4 เป็น ‘0’ บวกค่า 9 เข้ากับค่าข้อมูลการกด key
3. ถ้า column แรกไม่ถูกกด ให้บิตที่เป็น column ต่อไปเป็น ‘0’ (1111 1101B)
4. วนกลับไปทำ ข้อ 2 อีกครั้ง แต่ให้ค่าข้อมูลการกด key เปลี่ยนเป็น 2 ทำจนกว่าจะครบทุก column พร้อมทั้งเปลี่ยนค่าข้อมูลการกด key ไปเรื่อยๆ ตามลำดับ







Row	Pin	Port
3	2	3.6
2	3	3.5
1	5	3.4
0	6	3.3

Column	Pin	Port
2	7	3.2
1	8	3.1
0	9	3.0

การต่ออุปกรณ์

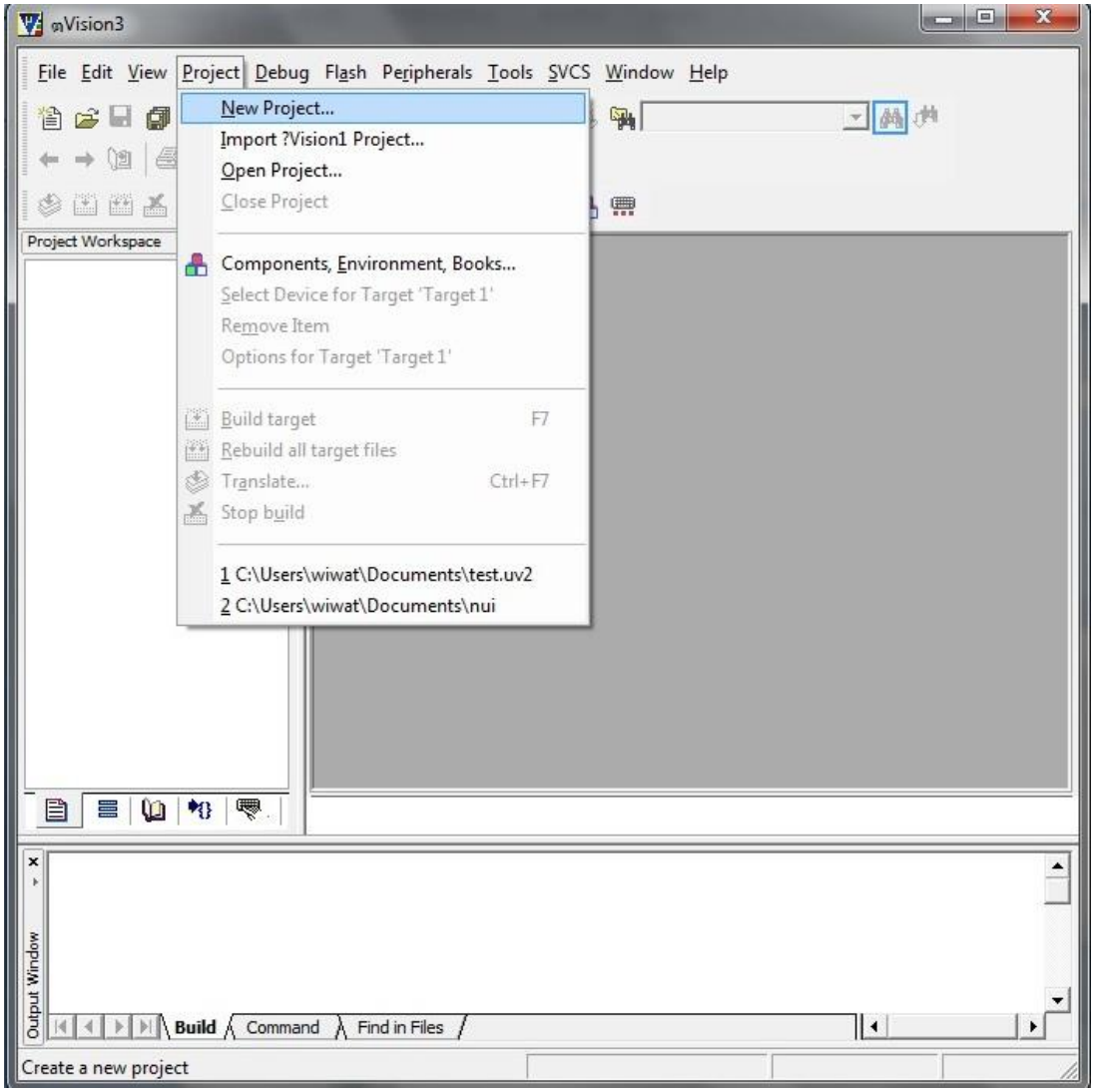
1. ที่ connector ของ keypad ต่อขา 1 เข้ากับขา 2 และขา 4 เข้ากับขา 8

2. ต่อ Port 1 เข้ากับconnector ของ keypad ดังแสดงในตาราง

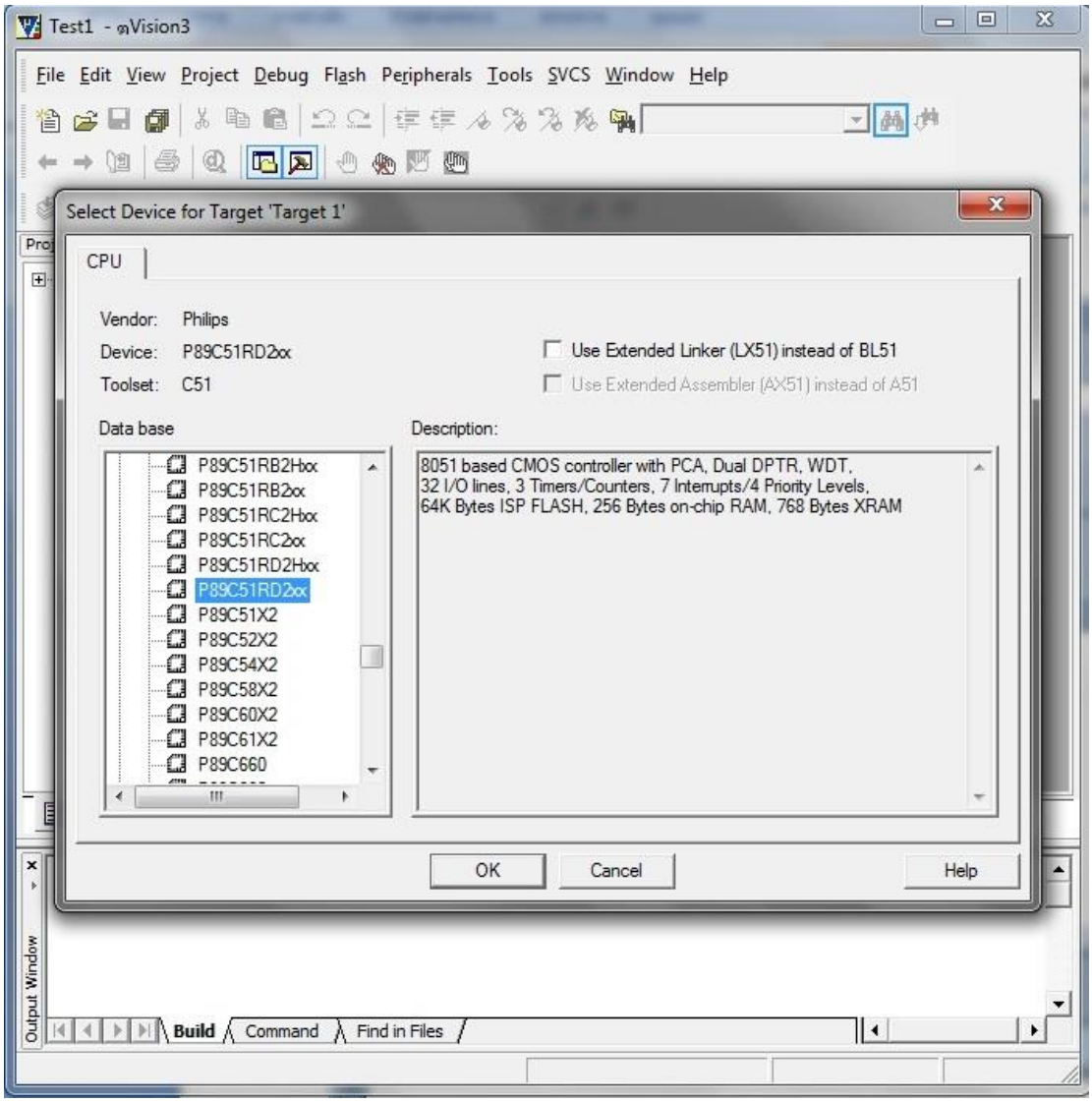
<u>วิธีการทดลอง</u>

1. ใช้ IDE สำหรับพัฒนาชุดคำสั่งของ MCS-51 ด้วยภาษา C โดยใช้ Keil51เพื่อเขียนโปรแกรม Lab05_x.c

2. สร้างโปรเจ็คใหม่โดยเลือก New project ตั้งชื่อเป็น Lab05 แล้วกด ok

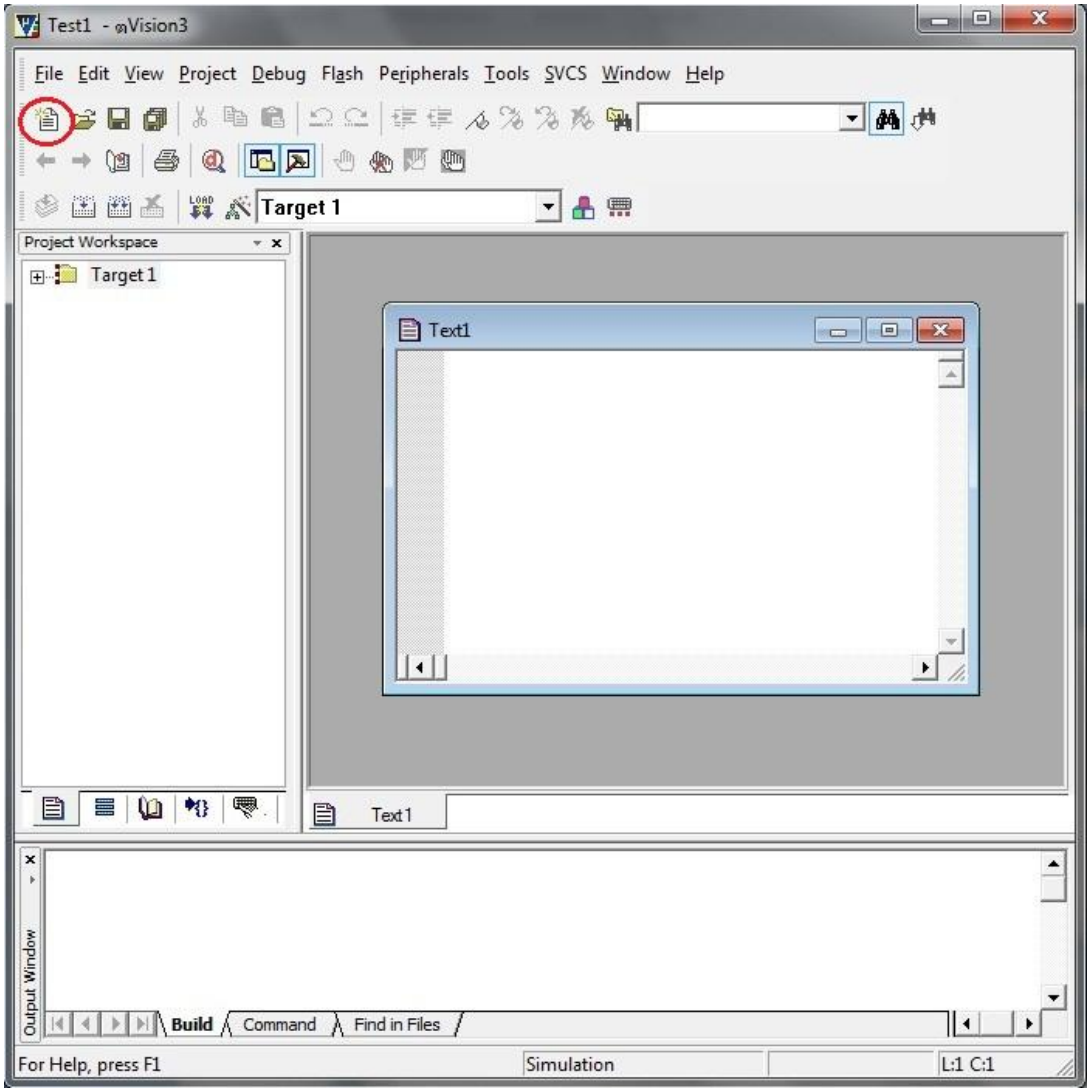

µVision3
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
New Project...
Import ?Vision1 Project...
Open Project...
Close Project
Project Workspace
Components, Environment, Books...
Select Device for Target 'Target 1'
Remove Item
Options for Target 'Target 1'
Build target F7
Rebuild all target files
Translate... Ctrl+F7
Stop build
1 C:\Users\wiwat\Documents\test.uv2
2 C:\Users\wiwat\Documents\nui
Output Window
Build Command Find in Files
Create a new project

3. เลือก CPU : Philips P89C51RD2xx

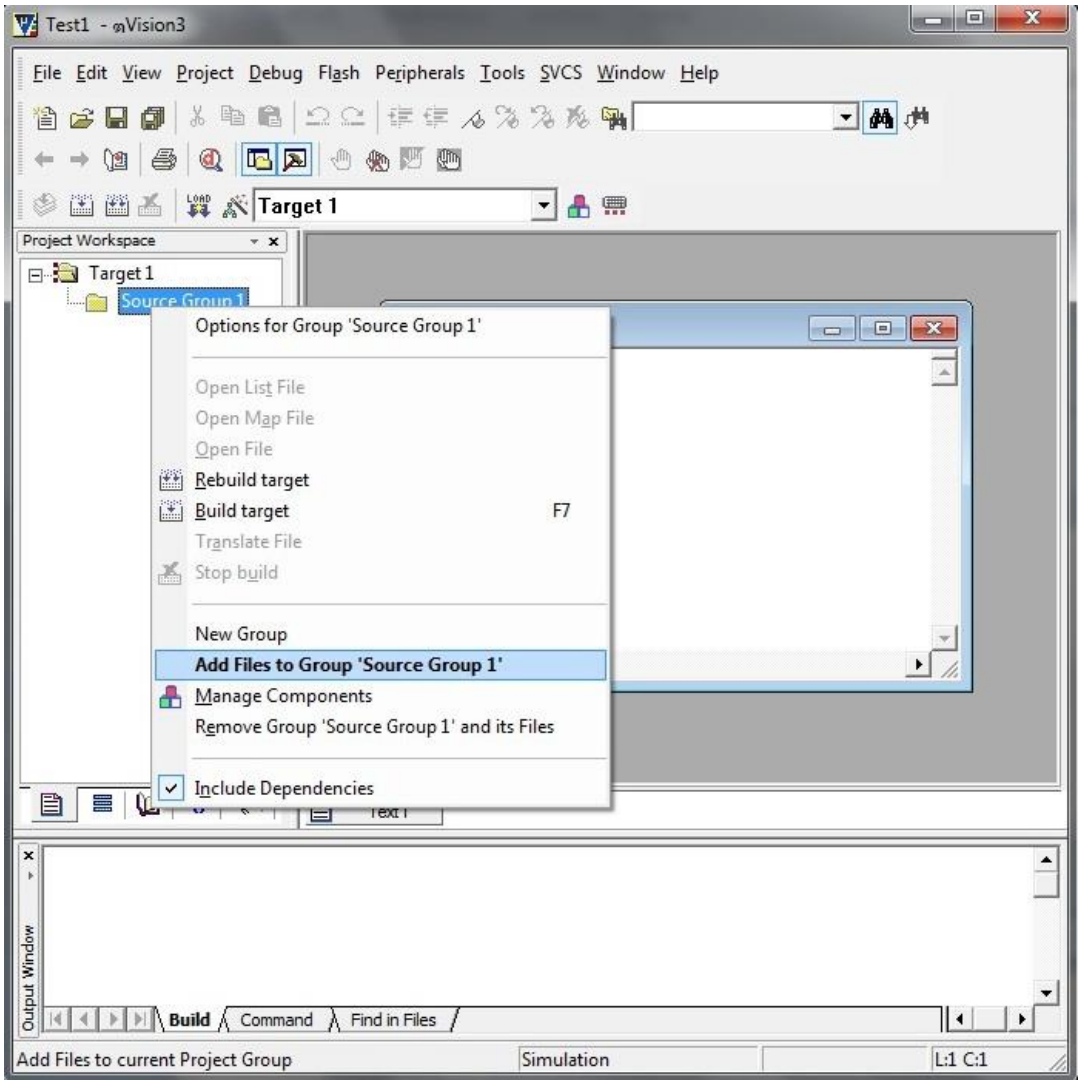


4. จากนั้นคลิ้กที่ปุ่มวงกลมสีแดง เพื่อสร้างหน้าต่างสำหรับเขียนโปรแกรม(หน้าต่าง Text1)

สร้างไฟล์ใหม่ เขียนโปรแกรมและ save โดยใช้ชื่อ Lab05_x.c (x แทนหมายเลขการทดลอง)



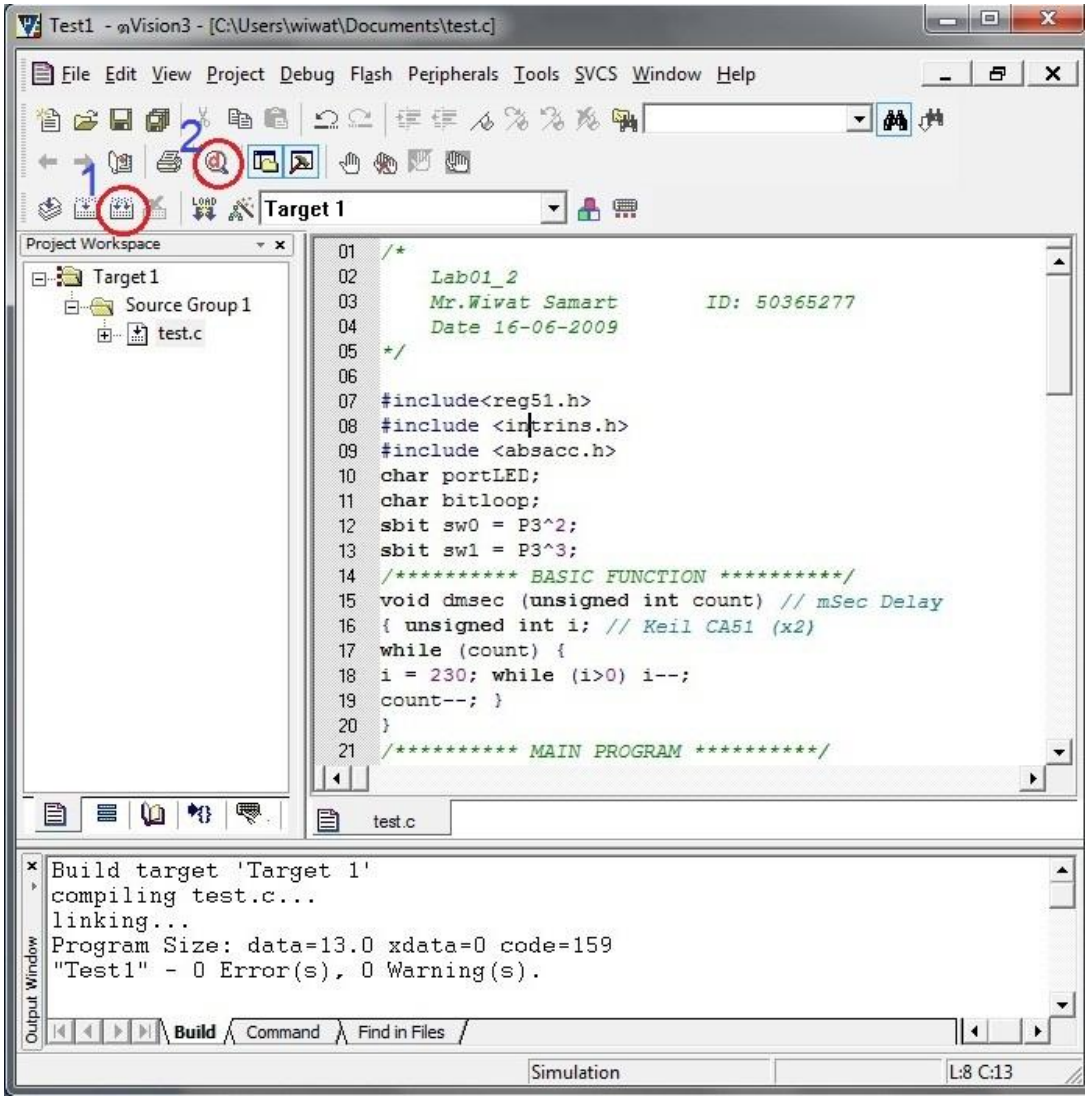
5. คลิ้กขวาที่ Source Group 1 แล้วเลือก Add file to group ‘Source Group 1’ แล้วเลือก File Lab05_x.c



6. เขียนโปรแกรมให้เสร็จแล้วคลิ้กที่ วงกลมที่ 1 เพื่อทำการ compile ถ้าหากว่าไม่พบ Error โดยที่ถ้าเกิด error ขึ้น โปรแกรมจะแสดงว่า error กี่ที่ (ดู windows output ด้านล่างของโปรแกรม) คลิ้กที่ วงกลมที่ 2 เพื่อให้ keil ตรวจสอบและทำการ debug พร้อมทั้งตรวจสอบขนาดของโปรแกรมที่เราสร้างขึ้นและพร้อมสำหรับการทดสอบการทำงานโปรแกรม

7. Build โปรแกรมที่เขียน โดยก่อนหน้านั้น click ขวา ที่ Target1 เลือก Options for Target ‘Target1’ ที่ Tab Output เลือก create hex file โดยเลือกที่ check box ‘Create Hex file’

File ที่ ได้จะมีชื่อเหมือนกับชื่อโปรเจ็ค คือ Lab05.hex สามารถเปลี่ยนแปลงชื่อได้โดยแก้ชื่อ output file ลงในช่อง ‘Name of Executable’



8. เมื่อ Compile และ Build ผ่านโดยไม่มีข้อผิดพลาด (error) ให้ต่อสายอนุกรมจากพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ PC (COM1) ไปยังพอร์ตอนุกรมของบอร์ด MCS-51 **

**ดูการ Download Hex File ให้กับ MCU ด้วยโปรแกรม Flip ในคู่มือ ET-Base51 V3 หน้า 9-15

9. บันทึกผลการทดลอง และอธิบายการทำงานของแต่ละคำสั่งในโปรแกรม และทำออกมาในรูปแบบรายงาน

10. ให้นิสิตทำการทดลองที่เหลือ โดยเมื่อทำการทดลองใหม่ให้สร้างไฟล์ใหม่ เขียนโปรแกรมและ save เป็นชื่อใหม่ เช่น สร้างไฟล์ใหม่ชื่อ Lab05_2.c (2แทนหมายเลขการทดลองที่ 2)

11. จากนั้น Click ที่ Source group แล้ว click ขวาที่ Lab05_1.c ซึ่งอยู่ใน Source group เลือก Remove file ' Lab05_1.c' เพื่อเอา file Lab05_1.c ออกจาก project

12. Click ขวาที่ Source group เลือก Add file to group แล้วเลือก File Lab05_2.c

13. ทำการทดลองที่เหลือเหมือนเดิม

การทดลองที่ 5.1

```
/*******************************************/

/* Example Program For ET-BASE51 V3.0(ED2) */

/* MCU      : AT89C51ED2(XTAL=29.4912 MHz) */

/*          : Frequency Bus = 58.9824 MHz  */

/* Compiler : Keil C51 (V7.50)             */

/* Write By : Eakachai Makarn(ETT CO.,LTD.)*/

/*******************************************/

/* Demo Character LCD(16x2) 4Bit Interface */



/* Include  Section */

#include <reg52.h>
                        // Standard 8052 SFR : File

#include <stdio.h>
                        // sprintf Function



/* AT89C51ED2 SFR */

sfr CKCON = 0x8F;
                        // Clock Control



/* LCD Interface */

#define PORT_LCD  P0                                    //
LCD Interface = Port P0

sbit RS = PORT_LCD^3;
            // RS LCD (0=Instruction,1=Data)

sbit RW = PORT_LCD^2;
            // RW LCD (0=Write,1=Read)

sbit E  = PORT_LCD^1;
            // Enable LCD(Active = "1")



char lcdbuf[16+1];
                        // LCD Display Buffer



/* prototype  section */

void init_lcd(void);
            // Initial Character LCD(4-Bit Interface)

void gotolcd(unsigned char);                            //
Set Cursor LCD

void write_ins(unsigned char);        // Write Instruction
LCD

void write_data(unsigned char);                         //
Write Data LCD

void enable_lcd(void);
            // Enable Pulse

char busy_lcd(void);
            // Read Busy LCD Status

void printlcd(void);
            // Display Message LCD

void delay(unsigned long);
            // Delay Time Function(1..4294967295)
```

```
/*-------------------------------------------

The main C function.  Program execution Here

-------------------------------------------*/

void main(void)

{

  CKCON = 0x01;
                    // Initial X2 Mode (BUS Clock =
58.9824 MHz)

  init_lcd();
                    // Initial LCD



  while(1)

  {

    gotolcd(0);
                    // Set Cursor Line-1

    sprintf(lcdbuf,"ET-BASE51V3(ED2)"); // Display Line-1

    printlcd();

    gotolcd(0x40);
          // Set Cursor Line-2

    sprintf(lcdbuf,"BY..ETT CO.,LTD."); // Display Line-2

    printlcd();

    delay(150000);
          // Display Delay

    gotolcd(0);
                    // Set Cursor Line-1

    sprintf(lcdbuf,"MCS51 High Speed"); // Display Line-1

    printlcd();

    gotolcd(0x40);
          // Set Cursor Line-2

    sprintf(lcdbuf,"58.98MHz Execute"); // Display Line-2

    printlcd();

    delay(150000);
          // Display Delay

  }

}



/*******************************/

/* Initial LCD 4-Bit Interface */

/*******************************/

void init_lcd(void)

{

  unsigned int i;
          // Delay Count



  E = 0;
                    // Start LCD Control (Disable)

  RS = 0;
                    // Default Instruction
```

```
  RW = 0;
                              // Default = Write Direction
  for (i=0;i<10000;i++);                                    //
Power-On Delay (15 mS)


  PORT_LCD &= 0x0F;
          // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= 0x30;
                    // DB5:DB4 = 1:1
  enable_lcd();
                    // Enable Pulse
  for (i=0;i<2500;i++);
          // Delay 4.1mS


  PORT_LCD &= 0x0F;
          // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= 0x30;
                    // DB5:DB4 = 1:1
  enable_lcd();
                    // Enable Pulse
  for (i=0;i<100;i++);
          // delay 100uS


  PORT_LCD &= 0x0F;
          // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= 0x30;
                    // DB5:DB4 = 1:1
  enable_lcd();
                    // Enable Pulse
  while(busy_lcd());                                        //
Wait LCD Execute Complete


  PORT_LCD &= 0x0F;
          // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= 0x20;
                    // DB5:DB4 = 1:0
  enable_lcd();
                    // Enable Pulse
  while(busy_lcd());                                        //
Wait LCD Execute Complete


  write_ins(0x28);
          // Function Set (DL=0: 4-Bit, N=1: 2 Line, F=0:
5X7)
  write_ins(0x0C);
          // Display on/off Control (D=1 Display ON, C=1
Cursor OFF, B=0 Blink OFF)
  write_ins(0x06);
          // Entry Mode Set (M=1 Increment, S=0 Cursor
Shift)
  write_ins(0x01);
          // Clear Display  (Clear Display, Set DD RAM
Address=0)
}
```

```
/*****************/

/* Set LCD Cursor */

/*****************/

void gotolcd(unsigned char i)

{

 i |= 0x80;

                    // Set DD-RAM Address Command

 write_ins(i);

}



/***************************/

/* Write Instruction to LCD */

/***************************/

void write_ins(unsigned char i)

{

 RS  = 0;

                          // Instruction Select

 RW  = 0;

                          // Write Select



 PORT_LCD &= 0x0F;

        // Clear old LCD Data (Bit[7..4])

 PORT_LCD |= i & 0xF0;

        // Strobe High Nibble Command

 enable_lcd();

                    // Enable Pulse



 PORT_LCD &= 0x0F;

        // Clear old LCD Data (Bit[7..4])

 PORT_LCD |= (i << 4) & 0xF0;                    //
Strobe Low Nibble Command

 enable_lcd();

                    // Enable Pulse



 while(busy_lcd());                                //
Wait LCD Execute Complete

}



/***************************/

/* Write Data(ASCII) to LCD */

/***************************/

void write_data(unsigned char i)

{

 RS  = 1;

                          // Data Select

 RW  = 0;

                          // Write Select
```

```
  PORT_LCD &= 0x0F;
          // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= i & 0xF0;                    //
Strobe High Nibble Data
  enable_lcd();
                  // Enable Pulse


  PORT_LCD &= 0x0F;
          // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= (i << 4) & 0xF0;             //
Strobe Low Nibble Data
  enable_lcd();
                  // Enable Pulse


  while(busy_lcd());                       //
Wait LCD Execute Complete
}

/***********************/
/* Enable Pulse to LCD */
/***********************/
void enable_lcd(void)
          // Enable Pulse
{
  unsigned int i;
          // Delay Count
  E = 1;
                          // Enable ON
  for (i=0;i<500;i++);
  E = 0;
                          // Enable OFF
}


/******************/
/* Wait LCD Ready */
/******************/
char busy_lcd(void)
{
  unsigned char busy_status;                //
Busy Status Read


  RS  = 0;
                          // Instruction Select
  RW  = 1;
                          // Read Direction
  E = 1;
                  // Start Read Busy
  busy_status = PORT_LCD;
          // Read LCD Data
```

```
 if(busy_status & 0x80)                                //
Read & Check Busy Flag
 {
   E = 0;
          // Disable Read
          RW = 0;
                              // Default = Write Direction
   return 1;
                    // LCD Busy Status
 }
 else
 {
   E = 0;
          // Disable Read
          RW = 0;
                              // Default = Write Direction
   return 0;
                    // LCD Ready Status
 }
}



/***************************/
/* Print Data(ASCII) to LCD */
/***************************/
void printlcd(void)
{
  char *p;

  p = lcdbuf;

  do
                                    // Get ASCII & Write to LCD
Until null
  {
    write_data(*p);
           // Write ASCII to LCD
    p++;
                       // Next ASCII
  }
  while(*p != '\0');                                    //
End of ASCII (null)

  return;
}

/*****************************************/
/* Long Delay Time Function(1..4294967295) */
/*****************************************/
```

```
void delay(unsigned long i)
{
  while(i > 0) {i--;}                    //
Loop Decrease Counter
  return;
}
```

การทดลองที่ 5.2

```
/*
File : LAB5_2.c
Description : Keypad and 7-segment Display
Clock : 11.0592 MHz
Hardware: Start c51
Compiler: Keil C Compiler
*/
#include<reg51.h>
#include <intrins.h>
#include <absacc.h>
#define SEGM P0// segment
#define DIGIT P1// Digit,BL,485con,sound,user-
led
/*key_col0 P3^0,
key_col1 P3^1,
key_col2 P3^2,
key_row0 P3^3,
key_row1 P3^4,
key_row2 P3^5
key_row3 P3^6*/
/********* INT-RAM WORKING AREA **********/
unsigned char DISBUF[8], KEY_DATA; // display
buffer
bit R_SHF;
unsigned char code SEGCODE[12]
={0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6
F,0x76,0x3F,0x63};
// segment code {1,2,3,4,5,6,7,8,9,*,0,-}
/********** BASIC FUNCTIONS **********/
void dmsec(unsigned int count2)
{
TMOD = (TMOD & 0xF0) | 0x01;/* Set T/C0 Mode*/
TR0=1;
while (count2)
{if (TF0==1)
```

```
{TR0=0; /* Stop Timer 0*/
TF0=0;
count2--;
TH0 = 0xEC; /* Load Timer withFFFFH- 5000 */
TL0 = 0x77;
TR0 = 1; /* Start Timer 0*/
}
}
}
unsigned char get_key (void) {// get key
unsigned char i,j,x,colx,w,rowx,k;
colx=0xfe;//11111110
rowx=0xf7;//11110111
P3=0x78; //01111000
k=0xff;
x = P3; // check for key press
if (x!=0x78)
{
dmsec(30);
if (x!=0x78)
// new press accepted
{ for (i=0;i<=2;i++)
// look for col
{ P3=colx;
w=P3&0x78;
colx=_crol_(colx,1);
if(w!=0x78)
{
for (j=0;j<=3;j++)
{
if (w==(rowx&0x78)){k=i+3*j+1;}
rowx=_crol_(rowx,1);
}
}
}
}
}
return (k);
}
/********** MAIN **********/
void main (void) {
unsigned int n,i;
dmsec(100);
while (1) {
KEY_DATA = get_key();
if (KEY_DATA!= 0xff){
for (n=0;n<=6;n++){
DISBUF[n]=DISBUF[n+1];
```

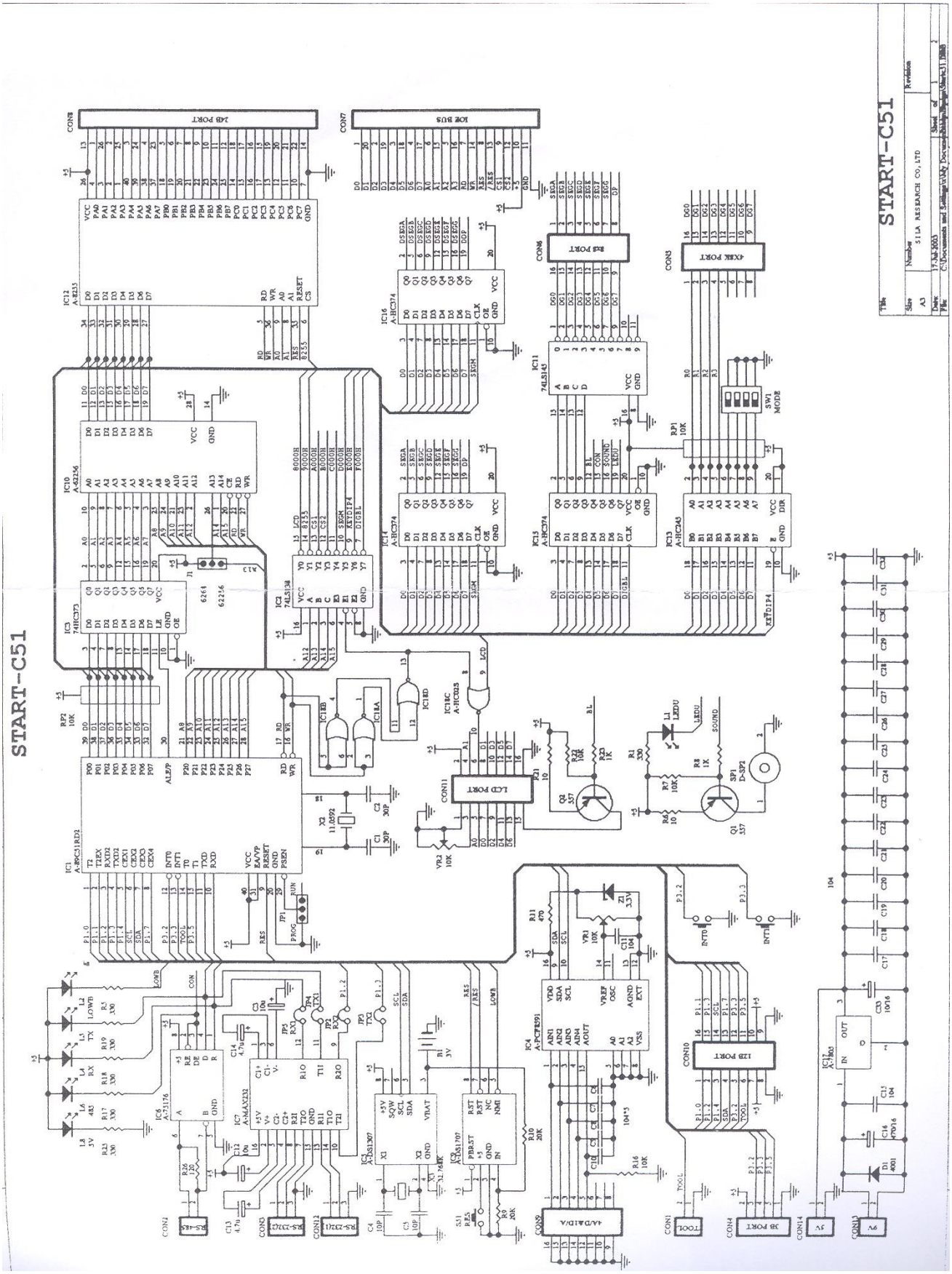
```
}
DISBUF[7]=SEGCODE[KEY_DATA-1];
}
for (i=0;i<=7;i++) {
// scan display
DIGIT = i;
SEGM = DISBUF[i]; // out segment
dmsec(10);
}
}
}
```

Reference

http://www.keil.com/support/man/docs/c166/c166_libref.htm

การบ้าน

1. เขียนโปรแกรมเพื่อให้เมื่อกดคีย์บอร์ดแล้วมีการแสดงผลบนจอ LCD


START-C51
START-C51