การทดลองที่ 7 การใช้งาน Real Time Clock

<u>วัตถุประสงค์</u>

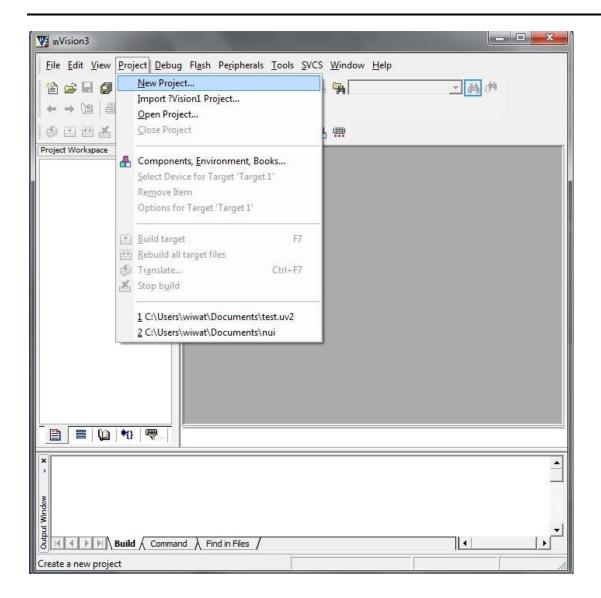
- 1. เพื่อให้นิสิตสามารถเขียนโปรแกรมภาษา C อย่างง่ายในการควบคุมไมโครคอนโทรลเลอร์ได้
- 3. เพื่อให้นิสิตสามารถใช้งาน Real Time Clock เบอร์ DS1307 ได้

<u>อุปกรณ์ในการทดลอง</u>

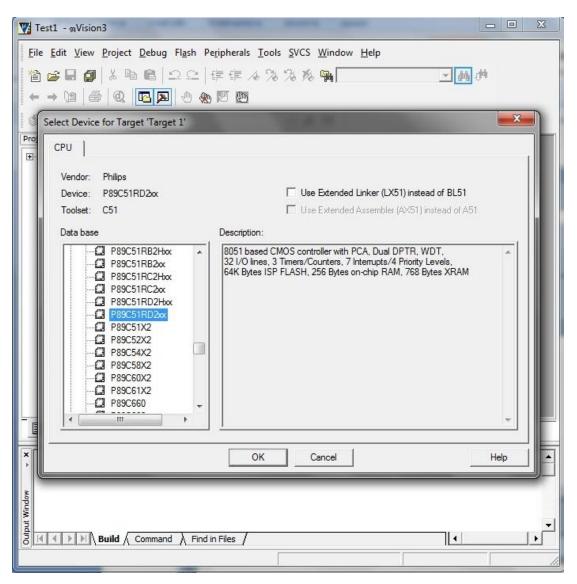
- 1. ชุดทดลองไมโครคอนโทรลเลอร์
- 2. เครื่องคอมพิวเตอร์ PC พร้อมโปรแกรมสำหรับการเขียนและคอมไพล์ภาษา C โปรแกรม Keil51 v.xx
- 3. สายต่อพอร์ตอนุกรม

การใช้งาน Keil51

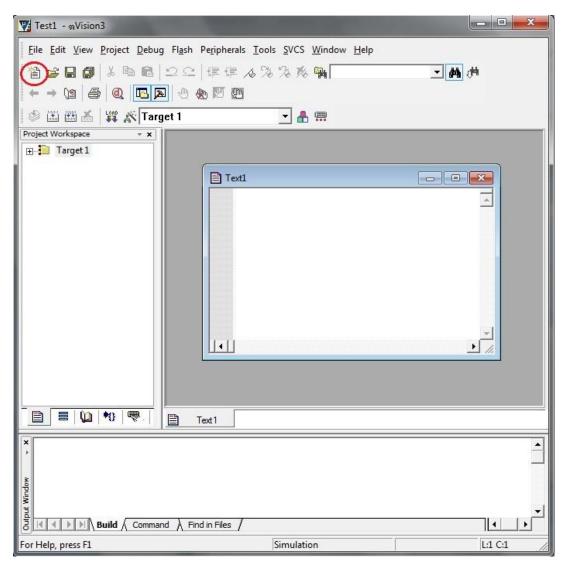
- 1. ใช้ IDE สำหรับพัฒนาชุดคำสั่งของ MCS-51 ด้วยภาษา C โดยใช้ Keil51เพื่อเขียนโปรแกรม Lab07_x.c
- 2. สร้างโปรเจ็คใหม่โดยเลือก New project ตั้งชื่อเป็น Lab07 แล้วกด ok



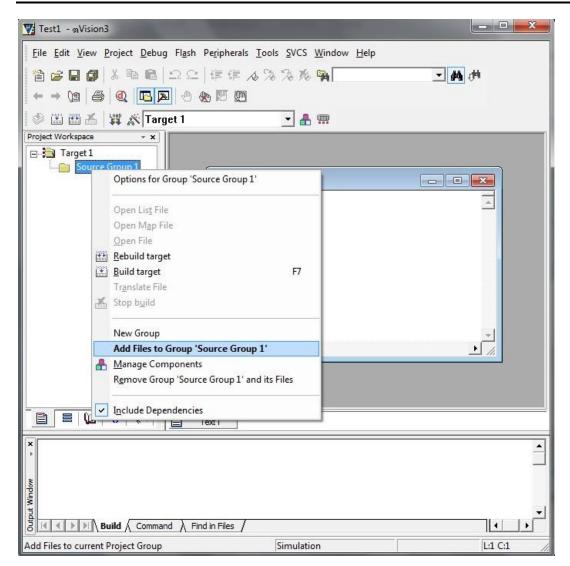
3. เลือก CPU : Philips P89C51RD2xx



4. จากนั้นคลิ้กที่ปุ่มวงกลมสีแดง เพื่อสร้างหน้าต่างสำหรับเขียนโปรแกรม(หน้าต่าง Text1) สร้างไฟล์ใหม่ เขียนโปรแกรมและ save โดยใช้ชื่อ Lab07_x.c (x แทนหมายเลขการทดลอง)

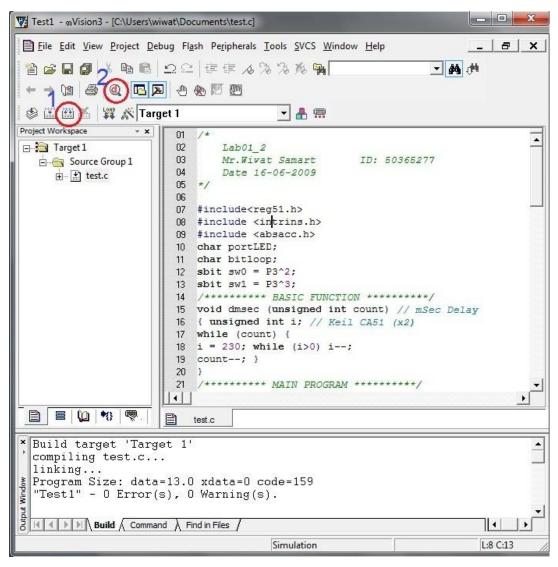


5. คลิ้กขวาที่ Source Group 1 แล้วเลือก Add file to group 'Source Group 1' แล้วเลือก File Lab07_x.c



- 6. เขียนโปรแกรมให้เสร็จแล้วคลิ้กที่ วงกลมที่ 1 เพื่อทำการ compile ถ้าหากว่าไม่พบ Error โดยที่ถ้าเกิด error ขึ้น โปรแกรมจะแสดงว่า error กี่ที่ (ดู windows output ด้านล่างของโปรแกรม) คลิ้กที่ วงกลมที่ 2 เพื่อให้ keil ตรวจสอบ และทำการ debug พร้อมทั้งตรวจสอบขนาดของโปรแกรมที่เราสร้างขึ้นและพร้อมสำหรับการทดสอบการทำงานโปรแกรม
- 7. Build โปรแกรมที่เขียน และให้เลือก option → create hex file โดยเลือกที่ check box Hex file ที่ ได้จะมีชื่อเหมือกับ ชื่อโปรเจ็ค คือ Lab07.hex

Lab 7



- 6. บันทึกผลการทดลอง และอธิบายการทำงานของแต่ละคำสั่งในโปรแกรม และทำออกมาในรูปแบบรายงาน
- 7. ให้นิสิตทำการทดลองที่เหลือ โดยเมื่อทำการทดลองใหม่ให้สร้างไฟล์ใหม่ เขียนโปรแกรมและ save เป็นชื่อใหม่ เช่น สร้างไฟล์ใหม่ชื่อ Lab07_2.c (2แทนหมายเลขการทดลองที่ 2)
- 8. จากนั้น Click ที่ Source group แล้ว click ขวาที่ Lab07_1.c ซึ่งอยู่ใน Source group เลือก Remove file ' Lab07_1.c' เพื่อเอา file Lab07_1.c ออกจาก project
- 9. Click ขวาที่ Source group เลือก Add file to group แล้วเลือก File Lab07_2.c
- 10. ทำเช่นนี้จนครบทุกการทดลอง

การทดลองที่ 7.1

```
/***************
                                                           1200 = 0x80(256-128=128)
/* Example Program For ET-BASE51 V3.0(ED2) */
                                                           2400 = 0 \times C0(256 - 94 = 192)
/* MCU : AT89C51ED2(XTAL=29.4912 MHz) */
                                                           4800 = 0 \times E0(256-32=224)
       : Frequency Bus = 58.9824 MHz */
                                                           9600 = 0 \times F0(256-16=240)
/* Compiler : Keil C51 (V7.50)
                                   */
                                                           19200 = 0xF8(256-8=248)
/* Write By : Eakachai Makarn(ETT CO.,LTD.)*/
                                                           38400 = 0 \times FC(256-4=252)
/*********************************/
                                                         */
// Demo I2C RTC(DS1307) Interrupt Mode
                                                         sbit scl = P3^7;
// SCL = P3.7
                                                                          // I2C SCL Signal
// SDA = P3.6
                                                         sbit sda = P3^6;
// INT = P3.3(INT1)
                                                                          // I2C SDA Signal
/* Include Section */
                                                         /* Pototype Section */
#include <reg52.h>
                                                         void ds1307_write_byte(unsigned char,unsigned
                 // Standard 8052 SFR: File
                                                         char);
                                                                          // Setup RTC
#include <stdio.h>
                         // For printf I/O functions
                                                         unsigned char ds1307_read_byte(unsigned char);
                                                                  // Read RTC
/* AT89C51ED2 SFR */
                                                         void i2c_send_byte(unsigned char);
sfr CKCON = 0x8F;
                                                                 // Write Byte I2C
                 // Clock Control
                                                         unsigned char i2c_get_byte(void);
                                                                                   // Read Byte I2C
/* Timer1 Baudrate (29.4912 MHz x 2 )
                                                         void delay_i2c(void);
 Reload = 256 - [58.9824 \text{ MHz} / (384 \times \text{Baud})]
                                                                 // Short Delay For I2C Bus Interface
```

```
void delay(unsigned long);
       // Delay Time Function(1..4294967295)
The main C function. Program execution Here
*/
void main (void)
  unsigned char j;
       unsigned char last_second;
  CKCON = 0x01;
               // Initial X2 Mode (58.9824 MHz)
       /* Initial MCS51 Serial Port */
  TMOD \&= 0x0F;
       // Reset old Timer1 Mode Config
  TMOD = 0x20;
       // Update Timer1 = 8 Bit Auto Reload
  SCON = 0x50;
               // Serial Port Mode 1 (N,8,1)
  ES = 0;
       // Disable Serial Interupt
```

```
ET1 = 0;
        // Disable Timer1 Interrupt
  PCON &= 0x7F;
        // SMOD1 = 0 (Disable Double Baudrate)
  TH1 = 0xF0;
// Setup Timer1 Baudrate 9600BPS / 58.9824 MHz
  TL1 = 0xF0;
  TR1 = 1;
        // Start Timer1 Generate Baudrate
  TI = 1;
        // Set TI to send First char of UART
        scl = 1;
        // Makesure I2C Stop Condition
        sda = 1;
        ds1307_write_byte(7,0x80);
        // Disable Square Wave Output
        ds1307_write_byte(2,0);
        // Set Hour
                      = 00
        ds1307_write_byte(1,0);
        // Set Minute = 00
```

```
ds1307_write_byte(0,0);
         // Start RTC Clock + Set Second = 00
         printf ("Hello This is ET-BASE51
V3.0(ED2)\n");
                          // Display Start Menu
         printf ("Demo Test RTC:DS1307 Polling
Read\n");
         printf ("HR:MN:SC\n");
         delay(100);
         while(1)
         {
   printf("\r");
                  // Restart Print
   j = ds1307\_read\_byte(2);
                 // Get Hour
    printf("%02BX",j);
    printf(":");
          j = ds1307\_read\_byte(1);
                  // Get Minute
          printf("%02BX",j);
   printf(":");
```

```
j = ds1307\_read\_byte(0);
                // Get Second
         last_second = j;
                // Save Second Reference
         printf("%02BX",j);
         do
// Repeat Get Second until Second Change
            j = ds1307\_read\_byte(0);
        // Get Second
     }
   while(last_second == j);
        // Repeat if Second Not Change
        }
}
/* Write Data 1-Byte to DS1307 */
/*********
void ds1307_write_byte(unsigned char
ds1307_address,unsigned char ds1307_data)
```

```
{
                                                           sda = 1;
 sda = 0;
                                                                                   // Release SDA
                // I2C Start Condition
                                                           scl = 1;
 scl = 0;
                                                                                   // Start ACK Clock
 delay_i2c();
                                                           delay_i2c();
                                                           while(sda) {}
 i2c_send_byte(0xD0);
                                                           scl = 0;
// Send ID Code DS1307, Write (1101000+0)
                                                                                   // End ACK Clock
 sda = 1;
                                                           delay_i2c();
                         // Release SDA
                                                           i2c_send_byte(ds1307_data);
 scl = 1;
                                                                  // Send DS1307 Data
                         // Start ACK Clock
                                                           sda = 1;
 delay_i2c();
                                                                                            // Release SDA
 while(sda) {}
 scl = 0;
                                                           scl = 1;
                 // End ACK Clock
                                                                                   // Start ACK Clock
                                                           delay_i2c();
 delay_i2c();
                                                           while(sda) {}
 i2c_send_byte(ds1307_address);
                                                           scl = 0;
                                                                                   // End ACK Clock
        // Send DS1307 Address
                                                           delay_i2c();
```

```
sda = 0;
                                                        scl = 0;
                                                        delay_i2c();
                        // Stop Bit(End of Data)
                                                        i2c_send_byte(0xD0);
 scl = 1;
                                // I2C Stop
                                                       // Send ID Code DS1307, Write (1101000+0)
Condition
                                                        sda = 1;
 delay_i2c();
                                                                                       // Release SDA
 sda = 1;
                                                        scl = 1;
                                                                               // Start ACK Clock
return;
}
                                                        delay_i2c();
                                                        while(sda) {}
/**********************/
                                                        scl = 0;
/* Read Data 1-Byte From DS1307 */
                                                                               // End ACK Clock
/*********
                                                        delay_i2c();
unsigned char ds1307_read_byte(unsigned char
ds1307_address)
                                                        i2c_send_byte(ds1307_address);
{
                                                               // Send DS1307 Address
 unsigned char ds1307_data;
                                                        sda = 1;
       // Dummy Byte
                                                                                       // Release SDA
                                                        scl = 1;
 sda = 0;
                                                                               // Start ACK Clock
                        // I2C Stat condition
```

```
delay_i2c();
                                                            sda = 1;
 while(sda) {}
                                                                                             // Release SDA
                                                            scl = 1;
 scl = 0;
                                                                                     // Start ACK Clock
                         // End ACK Clock
                                                            delay_i2c();
 delay_i2c();
                                                            while(sda) {}
 scl = 1;
                                                            scl = 0;
                                  // I2C Stop
                                                                                    // End ACK Clock
Condition
                                                            delay_i2c();
 delay_i2c();
 sda = 1;
                                                            ds1307_data = i2c_get_byte();
                                                                                                      //
                                                           Read 1-Byte From DS1307
 // New Start For Read //
 sda = 0;
                                                            sda = 1;
                                  // I2C Stat
condition
                                                                   // Send Stop Bit (End of Read Data)
 scl = 0;
                                                            scl = 1;
 delay_i2c();
                                                                           // Start Stop Bit Clock
                                                            delay_i2c();
 i2c_send_byte(0xD1);
                                                            scl = 0;
// Send ID Code DS1307, Read (1101000+1)
                                                                                     // End Stop Bit Clock
                                                            delay_i2c();
```

```
scl = 1;
                                // I2C Stop
Condition
 delay_i2c();
 sda = 1;
 return ds1307_data;
}
/*********************/
/* Send Data 8 Bit to I2C Bus */
/********/
void i2c_send_byte(unsigned char i)
{
 char j;
                        // Bit Counter Send
 for(j = 0;j < 8;j++)
                                                        }
                // 8-Bit Counter Send Data
 {
                                                      }
```

```
if((i \& 0x80) == 0x80)
      // Send MSB First
 {sda = 1;}
      // Send Data = 1
else
 {sda = 0;}
               // Send Data = 0
scl = 1;
               // Release SDA
delay_i2c();
scl = 0;
                        // Next Bit Send
delay_i2c();
i <<= 1;
      // Shift Data For Send (MSB <- LSB)
return;
```

```
/*************************/
/* Get Data 8 Bit From I2C Bus */
                                                         if(sda == 1)
/**************************/
                                                          {
unsigned char i2c_get_byte(void)
                                                                 i = 0x01;
                                                                                // Save Bit Data = 1
unsigned char i;
                                                                }
                // Result Byte Buffer
                                                         else
                                                          {
char j;
                                                                 i &= 0xFE;
                // Bit Counter Read Data
                                                                                // Save Bit Data = 0
for(j = 0; j < 8; j++)
                                                                }
                                                           scl = 0;
                // 8-Bit Counter Read Data
{
                                                                        // Next Bit Read
 i <<= 1;
                                                         delay_i2c();
                                 // Shift Result
Save (MSB <- LSB)
                                                         return i;
                                                       }
 sda = 1;
                                                       /******************************/
                        // Release Data
                                                       /* Delay For I2C Bus Device Interface */
 scl = 1;
                                                        // Strobe Read SDA
                                                       void delay_i2c(void)
 delay_i2c();
```

การทดลองที่ 7.2

```
/************************************/
                                                             /* Include Section */
/* Example Program For ET-BASE51 V3.0(ED2) */
                                                             #include <reg52.h>
                                                             // Standard 8052 SFR: File
/* MCU : AT89C51ED2(XTAL=29.4912 MHz) */
                                                             #include <stdio.h>
       : Frequency Bus = 58.9824 MHz */
                                                             // For printf I/O functions
/* Compiler : Keil C51 (V7.50) */
                                                             /* AT89C51ED2 SFR */
/* Write By : Eakachai Makarn(ETT CO.,LTD.)*/
                                                             sfr CKCON = 0x8F;
/**********************************/
                                                             // Clock Control
// Demo I2C RTC(DS1307) Interrupt Mode
                                                             /* Timer1 Baudrate (29.4912 MHz x 2 )
// SCL = P3.7
                                                               Reload = 256 - [58.9824 \text{ MHz} / (384 \times \text{Baud})]
// SDA = P3.6
                                                               1200 = 0 \times 80(256 - 128 = 128)
// INT = P3.3(INT1)
                                                               2400 = 0 \times C0(256-94=192)
```

```
4800 = 0 \times E0(256-32=224)
                                                             void delay_i2c(void);
                                                                                       // Short Delay For I2C Bus
 9600 = 0 \times F0(256-16=240)
                                                             Interface
 19200 = 0 \times F8(256 - 8 = 248)
 38400 = 0xFC(256-4=252)
                                                             The main C function. Program execution Here
sbit scl = P3^7;
                                                             void main (void)
// I2C SCL Signal
sbit sda = P3^6;
                                                               unsigned char j;
// I2C SDA Signal
// Bit Boolean 0=False,1=True //
                                                               CKCON = 0x01;
bit int1_status;
                         // INT1 Interrupt Flag Status
                                                                     // Initial X2 Mode (58.9824 MHz)
/* Pototype Section */
void ds1307_write_byte(unsigned char,unsigned char);
                                                                      /* Initial MCS51 Serial Port */
                 // Setup RTC
                                                               TMOD \&= 0x0F;
unsigned char ds1307_read_byte(unsigned char);
                                                                                       // Reset old Timer1 Mode Config
                         // Read RTC
                                                               TMOD = 0x20;
void i2c_send_byte(unsigned char);
                                                                              // Update Timer1 = 8 Bit Auto Reload
                                           // Write Byte I2C
                                                               SCON = 0x50;
unsigned char i2c_get_byte(void);
                                                                                       // Serial Port Mode 1 (N,8,1)
                                  // Read Byte I2C
                                                               ES = 0;
                                                                                       // Disable Serial Interupt
```

```
ET1 = 0;
                      // Disable Timer1 Interrupt
PCON \&= 0x7F;
              // SMOD1 = 0 (Disable Double Baudrate)
TH1 = 0xF0;
      // Setup Timer1 Baudrate 9600BPS / 58.9824 MHz
TL1 = 0xF0;
TR1 = 1;
                      // Start Timer1 Generate Baudrate
TI = 1;
              // Set TI to send First char of UART
      /* Setup INT1 Interrupt Control */
      EX1 = 1;
              // Enable External INT1
      IT1 = 1;
      // INT1 Trig on Falling Edge only
      EA = 1;
              // Enable Global Interrupt
      scl = 1;
      // Makesure I2C Stop Condition
```

```
sda = 1;
ds1307_write_byte(7,0x90);
        // Enable Square Wave Output = 1Hz
ds1307_write_byte(2,0);
        // Set Hour
                         = 00
ds1307_write_byte(1,0);
        // Set Minute = 00
ds1307_write_byte(0,0);
        // Start RTC Clock + Set Second = 00
printf ("Hello This is ET-BASE51 V3.0(ED2)\n");
        // Display Start Menu
printf ("Demo Test RTC:DS1307 Interrupt Read\n");
printf ("HR:MN:SC\n");
while(1)
// Loop Continue
 while(int1_status == 1)
                 // Wait Interrupt Complete
  printf("\r");
                          // Restart Print
```

```
j = ds1307\_read\_byte(2);
                                                                sda = 0;
                                   // Get Hour
     printf("%02BX",j);
     printf(":");
                                                                scl = 0;
    j = ds1307\_read\_byte(1);
                                                                delay_i2c();
                                   // Get Minute
     printf("%02BX",j);
                                                                i2c_send_byte(0xD0);
     printf(":");
           j = ds1307\_read\_byte(0);
                                                                sda = 1;
                                   // Get Second
           printf("%02BX",j);
                 int1\_status = 0;
                                                                scl = 1;
                                   // Reset Status
          }
                                                                delay_i2c();
                                                                while(sda) {}
}
                                                                scl = 0;
/***********************/
/* Write Data 1-Byte to DS1307 */
                                                                delay_i2c();
/************************/
void ds1307_write_byte(unsigned char
ds1307_address,unsigned char ds1307_data)
```

```
// I2C Start Condition
      // Send ID Code DS1307, Write (1101000+0)
              // Release SDA
      // Start ACK Clock
      // End ACK Clock
i2c_send_byte(ds1307_address);
                      // Send DS1307 Address
```

```
sda = 1;
                                                          delay_i2c();
                                                          sda = 0;
              // Release SDA
scl = 1;
                                                                        // Stop Bit(End of Data)
      // Start ACK Clock
                                                          scl = 1;
delay_i2c();
while(sda) {}
                                                                        // I2C Stop Condition
scl = 0;
                                                          delay_i2c();
                                                          sda = 1;
      // End ACK Clock
delay_i2c();
                                                        return;
i2c_send_byte(ds1307_data);
                              // Send DS1307 Data
                                                        /*************************/
sda = 1;
                                                        /* Read Data 1-Byte From DS1307 */
              // Release SDA
                                                        /*********
scl = 1;
                                                        unsigned char ds1307_read_byte(unsigned char
                                                        ds1307_address)
      // Start ACK Clock
delay_i2c();
                                                          unsigned char ds1307_data;
while(sda) {}
                                                        // Dummy Byte
scl = 0;
                                                          sda = 0;
      // End ACK Clock
```

```
// I2C Stat condition
                                                              scl = 1;
 scl = 0;
                                                                     // Start ACK Clock
 delay_i2c();
                                                              delay_i2c();
 i2c_send_byte(0xD0);
                                                              while(sda) {}
// Send ID Code DS1307, Write (1101000+0)
                                                              scl = 0;
 sda = 1;
                                                                     // End ACK Clock
                 // Release SDA
                                                              delay_i2c();
 scl = 1;
        // Start ACK Clock
                                                              scl = 1;
 delay_i2c();
                                                                              // I2C Stop Condition
 while(sda) {}
                                                              delay_i2c();
 scl = 0;
                                                              sda = 1;
        // End ACK Clock
                                                              // New Start For Read //
 delay_i2c();
                                                              sda = 0;
 i2c_send_byte(ds1307_address);
                                                                             // I2C Stat condition
                         // Send DS1307 Address
                                                              scl = 0;
 sda = 1;
                                                              delay_i2c();
                 // Release SDA
```

```
i2c_send_byte(0xD1);
                                                         scl = 0;
      // Send ID Code DS1307,Read (1101000+1)
                                                               // End Stop Bit Clock
sda = 1;
                                                         delay_i2c();
              // Release SDA
scl = 1;
                                                         scl = 1;
      // Start ACK Clock
                                                                       // I2C Stop Condition
delay_i2c();
                                                         delay_i2c();
while(sda) {}
                                                         sda = 1;
scl = 0;
                                                         return ds1307_data;
      // End ACK Clock
delay_i2c();
ds1307_data = i2c_get_byte();
                      // Read 1-Byte From DS1307
                                                        /*********
                                                        /* Send Data 8 Bit to I2C Bus */
sda = 1;
                                                        /*********
                                                       void i2c_send_byte(unsigned char i)
              // Send Stop Bit (End of Read Data)
scl = 1;
                                                        char j;
      // Start Stop Bit Clock
                                                               // Bit Counter Send
delay_i2c();
```

```
for(j = 0; j < 8; j++)
                                                            return;
                       // 8-Bit Counter Send Data
{
 if((i \& 0x80) == 0x80)
                       // Send MSB First
                                                          /*********
  {sda = 1;}
                                                          /* Get Data 8 Bit From I2C Bus */
                       // Send Data = 1
                                                          /*********
 else
                                                         unsigned char i2c_get_byte(void)
  {sda = 0;}
                               // Send Data = 0
                                                          unsigned char i;
                                                                                  // Result Byte Buffer
 scl = 1;
                                                          char j;
      // Release SDA
                                                                  // Bit Counter Read Data
 delay_i2c();
                                                          for(j = 0; j < 8; j++)
 scl = 0;
                                                                                  // 8-Bit Counter Read Data
      // Next Bit Send
 delay_i2c();
                                                           i <<= 1;
                                                                          // Shift Result Save (MSB <- LSB)
i <<= 1;
                                                           sda = 1;
      // Shift Data For Send (MSB <- LSB)
                                                                  // Release Data
```

```
scl = 1;
                                               //
Strobe Read SDA
                                                        /* Delay For I2C Bus Device Interface */
 delay_i2c();
                                                        void delay_i2c(void)
 if(sda == 1)
  {
                                                         unsigned char i;
         i = 0x01;
                                                         i = 100;
       // Save Bit Data = 1
                                                                        // Delay Counter
        }
                                                         while(i > 0) {i--;}
 else
                                                                                // Loop Decrease Counter
  {
                                                         return;
        i \&= 0xFE;
       // Save Bit Data = 0
                                                        /**********
        }
                                                        /* External INT1 Service Function */
                                                        /* Trig By RTC:DS1307 Every 1Hz */
 scl = 0;
                       // Next Bit Read
 delay_i2c();
                                                        void external1 (void) interrupt 2 using 1
                                                                                // External-INT1 : Bank-1
}
return i;
```

```
int1_status = 1;

// Set Interrupt Status
```

การบ้าน

1. เขียนโปรแกรมนับความเร็วรอบมอเตอร์โดยใช้ Real Time Clock เป็นฐานเวลา แสดงผลทางจอ LCD

การส่งงาน

ส่ง code ภาษา C มาทาง email และ เขียนอธิบาย program พอเข้าใจส่งพร้อม Flow chart และแสดงผล

Reference

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