// include the library code:

#include <LiquidCrystal.h>

#include <Wire.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(P2\_0, P2\_1, P2\_2, P2\_3, P2\_4, P2\_5);

const byte MTT\_I2C\_ADDR = 0x48;

const byte MTT\_CMD\_START\_CONVERT = 0x51;

const byte MTT\_CMD\_STOP\_CONVERT = 0x22;

const byte MTT\_CMD\_READ\_TEMP = 0xAA;

const byte MTT\_CMD\_ACCESS\_TH = 0xA1;

const byte MTT\_CMD\_ACCESS\_TL = 0xA2;

const byte MTT\_CMD\_ACCESS\_CONFIG = 0xAC;

const byte MTT\_CMD\_SOFTWARE\_POR = 0x54;

const byte MTT\_CONFIG\_DEFAULT = 0x8C;

const byte RTC\_I2C\_ADDR = 0x6F;

const byte RTC\_ST\_SEC = 0x00;

const byte RTC\_MINS = 0x01;

const byte RTC\_HRS = 0x02;

const byte RTC\_DAY = 0x03;

const byte RTC\_DATE = 0x04;

const byte RTC\_MONTH = 0x05;

const byte RTC\_YEAR = 0x06;

const byte RTC\_CONTROL= 0x07;

const char DayOfWeek[7][4] = {"Sun","Mon","Tue","Wed","Thu","Fri",

"Sat"};

const char MonthName[12][4] = {"Jan","Feb","Mar","Apr","May",

"Jun","Jul","Aug","Sep","Oct",

"Nov","Dec"};

byte mybuff[9]; // Working data for forming command sequences

byte value;

byte leapYear = 0, monthLength = 0;

byte mybuffTemp[3]; // Working data for forming command sequences

byte RwholeValue;

byte RfractionValue;

int fracC;

byte wholeF;

int fracF;

char Temp[7];

byte state;

int CorF = 0;

int convertFractionalPart(byte fC) {

int theValue=0;

int AMOUNT=5000;

for(int i=0; i<4; i++) {

if(0x80 & fC)

theValue += AMOUNT;

AMOUNT >>= 1;

fC <<= 1;

}

return theValue;

}

void convertCtoF(byte wC, int fC, byte \*wF, int \*fF ) {

long int CK, FK;

CK = wC\*10000L+fC;

FK = (9\*CK)/5+320000L;

\*wF = FK/10000L;

\*fF = FK%10000L;

}

int convertToDec(byte value)

{

return (value & 0x0f) + ((value & 0xf0) >> 4) \*10;

}

byte convertToHexish(byte value)

{

return (value/10)<<4 | (value%10);

}

void setup() {

//Set up the RTC

//---------------------------------------------------------------------------------------

Wire.setModule(0); // Required to select MSP430G2553IN20 pins 14/15 for I2C

Wire.begin(); // Initialize connection to I2C bus as master.

// Disable RTC in order to allow for unambiguous set-up of time/date.

mybuff[0]=RTC\_ST\_SEC; mybuff[1]=0x00;

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(mybuff,2);

Wire.endTransmission();

// Set minutes, hours, day, date, month, year

mybuff[0]=RTC\_MINS; // Address to begin writing into RTC

mybuff[1]=0x49; // Minute count is 49 (BCD)

mybuff[2]=0x19; // 24hr format, hour is 19

mybuff[3]=0x01; // First day of the week

mybuff[4]=0x19; // 19th day of the month

mybuff[5]=0x01; // 1st month of the year

mybuff[6]=0x79; // year 2089

mybuff[7]=0x43; // No alarms, square wave output at 32.768 kHz

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(mybuff,8);

Wire.endTransmission();

// Enable RTC with initialized seconds at 00 (BCD-ish), see Datasheet

mybuff[0]=RTC\_ST\_SEC; mybuff[1]=0x80;

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(mybuff,2);

Wire.endTransmission();

//Set up the Thermometer

//---------------------------------------------------------------------------------------

Wire.setModule(0);

Wire.begin(); // Initialize connection to I2C bus as master.

// Set MTT to the factory default configuration.

mybuffTemp[0]=MTT\_CMD\_ACCESS\_CONFIG; mybuffTemp[1]=MTT\_CONFIG\_DEFAULT;

Wire.beginTransmission(MTT\_I2C\_ADDR);

Wire.write(mybuffTemp,2);

Wire.endTransmission();

// Tell MTT to begin converting temperatures

Wire.beginTransmission(MTT\_I2C\_ADDR);

Wire.write(MTT\_CMD\_START\_CONVERT);

Wire.endTransmission();

//Set up the LCD and buttons

//-----------------------------------------------------------------------------------------

lcd.begin(16, 2);

pinMode(P1\_3, INPUT\_PULLUP);

pinMode(P1\_4, INPUT\_PULLUP);

//Set the initial state

//-----------------------------------------------------------------------------------------

state = 0;

}

void loop() {

if(state == 0)

{

//Take care of the RTC

//---------------------------------------------------------------------------------------

// HOURS

//--------------------------------------------------------------------------------

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_HRS);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read();

sprintf(Temp,"%02d",convertToDec(value));

lcd.setCursor(0, 0);

lcd.print(Temp);

// MINUTES

//--------------------------------------------------------------------------------

lcd.setCursor(2, 0);

lcd.print(":");

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_MINS);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read();

sprintf(Temp,"%02d",convertToDec(value));

lcd.setCursor(3, 0);

lcd.print(Temp);

// SECONDS

//--------------------------------------------------------------------------------

lcd.setCursor(5, 0);

lcd.print(":");

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_ST\_SEC);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read();

sprintf(Temp,"%02d",(value & 0x0f) + ((value & 0x70) >> 4) \*10);

lcd.setCursor(6, 0);

lcd.print(Temp);

// MONTH

//--------------------------------------------------------------------------------

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_MONTH);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read();

lcd.setCursor(0, 1);

lcd.print(MonthName[ convertToDec(0x1f & value) - 1] );

// DAY

//---------------------------------------------------------------------------------------

lcd.setCursor(3, 1);

lcd.print("-");

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_DATE);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read();

sprintf(Temp,"%02d",convertToDec(value));

lcd.setCursor(4, 1);

lcd.print(Temp);

//YEAR

//---------------------------------------------------------------------------------------

lcd.setCursor(6, 1);

lcd.print("-");

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_YEAR);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read();

lcd.setCursor(7, 1);

sprintf(Temp,"20%02d",convertToDec(value));

lcd.print(Temp);

// DAY OF WEEK

//---------------------------------------------------------------------------------------

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_DAY);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read();

lcd.setCursor(12, 1);

lcd.print(DayOfWeek[(value & 0x0f)-1]);

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//Take care of the thermometer

//-------------------------------------------------------------------------------------

Wire.beginTransmission(MTT\_I2C\_ADDR);

Wire.write(MTT\_CMD\_READ\_TEMP);

Wire.endTransmission();

Wire.requestFrom(MTT\_I2C\_ADDR,2);

RwholeValue=Wire.read();

RfractionValue=Wire.read();

fracC=convertFractionalPart(RfractionValue);

convertCtoF(RwholeValue,fracC,&wholeF,&fracF);

if(digitalRead(P1\_3) == LOW)

{

CorF = !CorF;

delay(500);

}

if(CorF == 1)

{

sprintf(Temp,"%d.%01d C",RwholeValue,round(convertFractionalPart(RfractionValue)/1000));

lcd.setCursor(10, 0);

lcd.print(Temp);

}

else

{

sprintf(Temp,"%d.%01d F",wholeF,round(fracF/1000));

lcd.setCursor(10, 0);

lcd.print(Temp);

}

// Check for request to set starting values

if(digitalRead(P1\_4) == LOW)

{

state++;

delay(500);

}

}

else if(state == 1) // Stop clock and GET YEAR

{

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_YEAR);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=convertToDec(Wire.read());

lcd.clear();

lcd.setCursor(0, 0);

sprintf(Temp,"Set year: 20%02d",value);

lcd.print(Temp);

state++;

}

else if(state == 2) // SET YEAR

{

lcd.setCursor(12,0);

if(digitalRead(P1\_3) == LOW)

{

if(value < 99)

value++;

else

value = 1;

sprintf(Temp,"%02d",value);

lcd.print(Temp);

delay(200);

}

// Check for request to set starting values

if(digitalRead(P1\_4) == LOW)

{

state++;

if(value%4 == 0)

leapYear = 1;

else

leapYear = 0;

mybuff[6]=convertToHexish(value);

delay(500);

}

}

else if(state == 3) // GET MONTH

{

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_MONTH);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=convertToDec(Wire.read() & 0x1f);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Set month: ");

lcd.print(MonthName[value-1]);

state++;

}

else if(state == 4) // SET MONTH

{

lcd.setCursor(11,0);

if(digitalRead(P1\_3) == LOW)

{

if(value < 12)

value++;

else

value = 1;

lcd.print(MonthName[value-1]);

delay(400);

}

// Check for request to set starting values

if(digitalRead(P1\_4) == LOW)

{

if(value == 2)

{

if(leapYear == 1)

monthLength = 29;

else

monthLength = 28;

}

else if(value == 9 || value == 4 || value == 6 || value == 11)

monthLength = 30;

else

monthLength = 31;

state++;

if(leapYear == 1)

mybuff[5] = 0x20 | convertToHexish(value);

else

mybuff[5] = convertToHexish(value);

delay(500);

}

}

else if(state == 5) // GET DAY

{

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_DATE);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=convertToDec(Wire.read());

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Set day: ");

sprintf(Temp,"%02d",value);

lcd.print(Temp);

state++;

}

else if(state == 6) // SET DAY

{

lcd.setCursor(9,0);

if(digitalRead(P1\_3) == LOW)

{

if(value < monthLength)

value++;

else

value = 1;

sprintf(Temp,"%02d",value);

lcd.print(Temp);

delay(300);

}

// Check for request to set starting values

if(digitalRead(P1\_4) == LOW)

{

state++;

mybuff[4]=convertToHexish(value);

delay(500);

}

}

else if(state == 7) // GET DAY OF THE WEEK

{

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_DAY);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=Wire.read() & 0x0f;

lcd.setCursor(0, 1);

lcd.print(value);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Set Weekday: ");

lcd.print(DayOfWeek[value-1]);

state++;

}

else if(state == 8) // SET DAY OF THE WEEK

{

lcd.setCursor(13,0);

if(digitalRead(P1\_3) == LOW)

{

if(value < 7)

value++;

else

value = 1;

lcd.print(DayOfWeek[value-1]);

delay(500);

}

// Check for request to set starting values

if(digitalRead(P1\_4) == LOW)

{

state++;

mybuff[3]=value & 0x0f;

delay(500);

}

}

else if(state == 9) // GET HOUR

{

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_HRS);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=convertToDec(Wire.read());

lcd.clear();

lcd.setCursor(0, 0);

sprintf(Temp,"Set hour: %02d",value);

lcd.print(Temp);

state++;

}

else if(state == 10) // SET HOUR

{

lcd.setCursor(10,0);

if(digitalRead(P1\_3) == LOW)

{

if(value < 23)

value++;

else

value = 0;

sprintf(Temp,"%02d",value);

lcd.print(Temp);

delay(400);

}

// Check for request to set starting values

if(digitalRead(P1\_4) == LOW)

{

state++;

mybuff[2]=convertToHexish(value);

delay(500);

}

}

else if(state == 11) // GET MINUTES

{

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(RTC\_MINS);

Wire.endTransmission();

Wire.requestFrom(RTC\_I2C\_ADDR,1);

value=convertToDec(Wire.read());

lcd.clear();

lcd.setCursor(0, 0);

sprintf(Temp,"Set minute: %02d",value);

lcd.print(Temp);

state++;

}

else if(state == 12) // SET MINUTES

{

lcd.setCursor(12,0);

if(digitalRead(P1\_3) == LOW)

{

if(value < 59)

value++;

else

value = 0;

sprintf(Temp,"%02d",value);

lcd.print(Temp);

delay(200);

}

// Check for request to set starting values

if(digitalRead(P1\_4) == LOW)

{

state++;

mybuff[1]=convertToHexish(value);

delay(500);

}

}

else if(state == 13)

{

// Set minutes, hours, day, date, month, year

mybuff[0] = RTC\_MINS;

mybuff[7] = 0x43;

Wire.beginTransmission(RTC\_I2C\_ADDR);

Wire.write(mybuff,8);

Wire.endTransmission();

// Go back to clock and temp display

lcd.clear();

state = 0;

}

}

Sketch uses 14,352 bytes (87%) of program storage space. Maximum is 16,384 bytes.

Global variables use 262 bytes (51%) of dynamic memory, leaving 250 bytes for local variables. Maximum is 512 bytes.