# Wire Level: TC74

// PROJECT  :TC74WireAccess

// PURPOSE  :ICS3U-E: First look at Wire level access to an I2C Device

// COURSE   :ICS3U

// AUTHOR   :C. D'Arcy

// DATE     :2023 01 27

// MCU      :\*

// STATUS   :Working

// REFERENCE:[http://darcy.rsgc.on.ca/ACES/Datasheets/TC74.pdf#page=8](http://darcy.rsgc.on.ca/ACES/Datasheets/TC74.pdf%23page%3D8)

#include <Wire.h>

#define TC74ADDRESS 0x4D //0x4D(B100\_1101 or 0x48(B100\_1000)

#define DURATION 1000 //pacing

void setup() {

  **Serial**.begin(9600);

  while(!**Serial**);

  Wire.begin();             // Initiate the Wire library

  **Serial**.println("I2C TC74 Temperature Sensor ...");

}

void loop() {

  Wire.beginTransmission(TC74ADDRESS);// Begin transmission to the TC74

  Wire.write(0);       // Ask for data (starting) in (base) register 0: Temperature

  Wire.endTransmission();             // End the transmission, then...

  Wire.requestFrom(TC74ADDRESS,1);    // ...request number of bytes from register

  while (Wire.available() == 0);      // pause & wait for the data to be sent

  int8\_t celsius = Wire.read();       // read the byte(s) returned on the I2C bus

  float fahrenheit = celsius \* 1.8 + 32;    //or round()

  //Publish

  **Serial**.print(String(celsius));      //

  **Serial**.write(0xC2);                 // <https://www.utf8-chartable.de/>

  **Serial**.write(0xB0);                 // UTF-8 degree symbol

  **Serial**.print("C\t");                //

  **Serial**.print(String(fahrenheit,1));

  **Serial**.write(0xC2);                 // UTF-8 degree symbol

  **Serial**.write(0xB0);                 //

  **Serial**.print("F\n");                //

  delay(DURATION);

}

# Wire Level DS1307 RTC

// PROJECT  :WireLevelRTC

// PURPOSE  :Part of ICS3U-E's Wire Level I2C Introduction

// AUTHOR   :C. D'Arcy

// DATE     :2018 04 24. 2020 02 08. 2023 01 27.

// uC       :328p     //use TinyWire.h for 84 & 85

// REFERENCE:[http://darcy.rsgc.on.ca/ACES/TEI3M/WireLevelRTCExercises.html#time](http://darcy.rsgc.on.ca/ACES/TEI3M/WireLevelRTCExercises.html%23time)

#include <Wire.h>

struct timeDate {

  uint8\_t hours;

  uint8\_t minutes;

  uint8\_t seconds;

  uint8\_t date;

  uint8\_t mon;

  uint8\_t YEAR;

};

#define RTCADDRESS 0x68

char buff[40];

void setup() {

  **Serial**.begin(9600);

  **Serial**.println("DS1307RTC Read Test");

  **Serial**.println("~~~~~~~~~~~~~~~~~~~");

  /\*pinMode(A2, OUTPUT);

  pinMode(A3, OUTPUT);

  digitalWrite(A2, LOW);

  digitalWrite(A3, HIGH);\*/

  Wire.begin();

  while (true) {

    Wire.beginTransmission(RTCADDRESS);

    Wire.write(0);

    Wire.endTransmission();

    Wire.requestFrom(RTCADDRESS, 7);

    while (!Wire.available());

    uint8\_t hours;

    uint8\_t minutes;

    uint8\_t seconds;

    uint8\_t date;

    uint8\_t mon;

    uint8\_t YEAR;

    getAll(hours, minutes, seconds, date, mon, YEAR);

    sprintf(buff, "Time = %02d:%02d:%02d,", bcd2dec(hours), bcd2dec(minutes), bcd2dec(seconds));

    **Serial**.print(buff);

    sprintf(buff, " Data (D/M/Y) = %2d/%02d/%02d", bcd2dec(date), bcd2dec(mon), bcd2dec(YEAR));

    **Serial**.println(buff);

    delay(1000);

  }

}

uint8\_t bcd2dec(uint8\_t bcd) {

  return 10 \* (bcd >> 4) + (bcd & 0xF);

}

void getAll(uint8\_t &ho, uint8\_t &mi, uint8\_t &se, uint8\_t &da, uint8\_t &mo, uint8\_t &ye) {

  se = Wire.read();

  mi = Wire.read();

  ho = Wire.read();

  Wire.read();        // ignore DAY

  da = Wire.read();

  mo = Wire.read();

  ye = Wire.read();

}

void loop() {}

# RTC AND RAM ADDRESS MAP (taken from the DS1307 Datasheet, pp.7-8)

**Table 2** shows the address map for the DS1307 RTC and RAM registers. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multibyte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.



# Wire Level DS1307 SQW Plotter

// PROJECT  :WireLevelSQWPlotter

// PURPOSE  :Plotting the Square Wave output from the DS1307RTC

// AUTHOR   :C. D'Arcy

// DATE     :2020 12 10. Confirmed 2023 01 27.

// STATUS   :Working

// REFERENCE:<http://darcy.rsgc.on.ca/ACES/Datasheets/DS1307.pdf>

#include <Wire.h>

#define RTCADDRESS  0x68

#define SQWREGISTER 7

#define SQWE        4

#define RS0         0

#define RS1         1

#define Hz1         0<<RS1|0<<RS0   // 1 Hz

#define Hz4k        0<<RS1|1<<RS0   // 4.096 Hz (2^12)

#define Hz8k        1<<RS1|0<<RS0   // 8.192 Hz (2^13)

#define Hz32k       1<<RS1|1<<RS0   // 32.768 Hz (2^15)

#define INTERRUPTPIN 2

volatile uint8\_t count = 0;

void setup() {

  **Serial**.begin(9600);     //reduce to 1200 to slow plotter down....

/\*  pinMode(A2, OUTPUT);

  pinMode(A3, OUTPUT);

  digitalWrite(A2, LOW);

  digitalWrite(A3, HIGH);\*/

  attachInterrupt(digitalPinToInterrupt(INTERRUPTPIN), SQW\_pulse, FALLING);

  pinMode(INTERRUPTPIN, INPUT\_PULLUP);

  Wire.begin();

  setSQW(Hz1);

}

uint8\_t setSQW(uint8\_t frequency) {

  Wire.beginTransmission(RTCADDRESS);

  Wire.write(SQWREGISTER);

  Wire.write(1 << SQWE | frequency); // RS1:RS0=00 selects 1Hz

  Wire.endTransmission();

  // optional: read it back for verification/confirmation...

  Wire.beginTransmission(RTCADDRESS);

  Wire.write(SQWREGISTER);

  Wire.endTransmission();

  Wire.requestFrom(RTCADDRESS, 1);

  while (!Wire.available());

  uint8\_t value = Wire.read();

  return value;

}

void SQW\_pulse () {

  count++;

}

void loop() { **Serial**.println(count%2);

# Wire Level RTC/LCD Code Shell

// PROJECT  :WireLevelRTCLCD

// PURPOSE  :Stage 3 (of 4) for the Wire-level I2C Introduction for ICS3U

// AUTHOR   :C. D'Arcy

// DATE     :2020 02 13

// uC       :328p

// HARDWARE :DS1307 RTC Breakout and LCD Appliance

// STATUS   :Shell to be completed

// REFERENCE:[http://darcy.rsgc.on.ca/ACES/TEI3M/WireLevelRTCExercises.html#time](http://darcy.rsgc.on.ca/ACES/TEI3M/WireLevelRTCExercises.html%23time)

#include <Wire.h>

#define RTCADDRESS 0x00           //DS1307 I2C Address?

#include <**LiquidCrystal**.h>        //LCD Library...

#define LCD\_COLUMNS 16            //Number of columns in Character LCD screen

#define LCD\_ROWS    2             //Number of rows on LCD screen

//LCD appliance-compatible pin assignments...

uint8\_t RS = 7, EN = 6, D4 = 5, D5 = 4, D6 = 3, D7 = 2;

**LiquidCrystal** lcd(RS, EN, D4, D5, D6, D7);

**char buff[LCD\_COLUMNS];**      //convenient output buffer for formatted prints

**struct timeDate** {            //tight binding of RTC register data into a struct

  uint8\_t seconds;

  uint8\_t minutes;

  uint8\_t hours;

  uint8\_t date;

  uint8\_t mon;

  uint8\_t YEAR;

} tD;

**void setup()** {

  lcd.begin(LCD\_COLUMNS, LCD\_ROWS);       //initialize LCD screen

  lcd.clear();                            //erase the LCD screen

  lcd.setCursor(0, 0);                    //position display cursor at home

  lcd.print("DATE:");                     //provide a label on the first line

  lcd.setCursor(0, 1);                    //position cursor at start of second row

  lcd.print("TIME:");                     //provide a label on the second line

  pinMode(A2, OUTPUT);                    //provide supply for

  pinMode(A3, OUTPUT);                    // the DS1307 breakout board

  digitalWrite(A2, LOW);                  // inserted as an appliance

  digitalWrite(A3, HIGH);                 //

  Wire.begin();                           //initialize a Wire session

}

uint8\_t BCD2bin(uint8\_t bcd) {

  return 0;                    //compose an expression that converts BCD to binary

}

//the body of this method needs to be developed...

**void getAll(timeDate &td)** { //'**Address Of**' operator, **&**, is the key to no **return**!

??

}

**void loop()** {

  Wire.beginTransmission(RTCADDRESS);

  Wire.write(0);

  Wire.endTransmission();

  Wire.requestFrom(RTCADDRESS, 7);

  //while (!Wire.available()); **//uncomment when going live…**

  getAll(tD);

  sprintf(buff, "%2d/%02d/%02d", tD.date, tD.mon, tD.YEAR);

  lcd.setCursor(5, 0);

  lcd.print(buff);

  sprintf(buff, "%02d:%02d:%02d", tD.hours, tD.minutes, tD.seconds);

  lcd.setCursor(5, 1);

  lcd.print(buff);

  delay(1000); **//to be replaced in the next Stage**

}

# RTC AND RAM ADDRESS MAP (taken from the DS1307 Datasheet, pp.7-8)

**Table 2** shows the address map for the DS1307 RTC and RAM registers. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multibyte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

