# Lab 10

# Setting Real Time Clock

In this week’s lab we will build a real time clock using the timer0 code from Lab8.

Use the functions created in Lab8 and Lab9 to design a real time clock with the ability to set Hours, Minutes, Seconds, and am/pm independently. Make sure that your code conforms to the following specification:

**Software:**

Implement a real time clock.

* The time is displayed to the nearest second on the LCD display.
* You must display Hours, minutes and seconds on the first line in the following format: HH:MM:SS Xm where X = a or p
* Timer 0 is to be used to give a 10ms ' tickCount ' using an interrupt.
* RF3 will flash every second.
* You must be able to set the time from the keypad. Hours, Minutes, Seconds, and am/pm must be set independently.
* The clock code must be your own.

You must have completely commented code. Also include the following comment block.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Student Name: \*\*\*\*\*\*Include this comment block\*\*\*\*\*\*

File Name:

Description:

Demo Date: \*\*\*\*\*\*fill in the marking check list\*\*\*\*\*\*

Marking Check List

Good Comments:

Good Modular Code:

Unsigned Char used correctly:

Format HH:MM:SS Xm correct:

Increments each Second:

10ms Tick correct:

Set time logical and complete:

Only valid characters are display:

Clock is accurate:

Am/pm roll over is correct:

Hours per day correct:

Bugs:

Mark:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Screen shot of your new code:

**“setRTCmain.c”**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Student Name: Aaron Huinink

File Name: setRTCmain.c

Description: Implements a settable RTC on the LCD

Demo Date: March 21, 2023

Marking Check List

Good Comments:

Good Modular Code:

Unsigned Char used correctly:

Format HH:MM:SS Xm correct:

Increments each Second:

10ms Tick correct:

Set time logical and complete:

Only valid characters are display:

Clock is accurate:

Am/pm roll over is correct:

Hours per day correct:

Bugs:

Mark:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// ============================ INCLUDES/DEFINES =================================//

#include "RealTimeClock.h"

#include "keypad18f.h"

#include <xc.h>

// ============================ MAIN =================================//

void main(void){

// ============================ SETUP =================================//

// ---- VARIABLES ---- //

// time variable to store the current time

time current\_time;

current\_time.hours = 12;

current\_time.minutes = 0;

current\_time.seconds = 0;

current\_time.meridian = 'a';

// position variable to store the current position

unsigned char position = 0;

// indexer array for indexing cursor position

char indexer[12] = "HH:MM:SS Xm";

// key variable to store key inputs

char key;

// ---- HARDWARE INIT ---- //

// initialize LCD

LCDinit();

// initialize RTC run flag

nSetRTC = 1;

// ============================ MAIN LOOP =================================//

while(1){

// when the nSetRTC flag is de-asserted

if(nSetRTC){

RTCrun(&current\_time); // run the clock

};

// transition state

LCD\_HOME;

position = 0;

// if the interrupt on RB4 occurs

while(!nSetRTC){ // while the nSet flag is asserted

key = keyScan(); // get a key press from the keypad

// deal with keypad inputs

switch (key) {

case 'A': // "Advance cursor"

LCDgoto(++position); // move cursor right

break;

case 'B': // "Backspace cursor"

if(position != 0){

LCDgoto(--position); // move cursor left if not at pos 0

}

break;

case 'C': // clear

current\_time.hours = 12;

current\_time.minutes = 0;

current\_time.seconds = 0;

current\_time.meridian = 'a'; // reset to midnight

nSetRTC = 1; // de-assert nSet

break;

case 'D': // enter

nSetRTC = 1; // de-assert nSet;

break;

case '#':

break;

case '\*': // toggle am/pm

current\_time.meridian = (current\_time.meridian == 'a')? 'p' : 'a';

LCDgoto(0x09); // go to am/pm

LCDprintc(current\_time.meridian); // put the new value to the display

LCDgoto(position); // go back to original cursor position

break;

default: // an integer

// match cursor position to attribute

switch(indexer[position]){

case 'H': // if hours position

if ((position % 3)==0){ // if tens digit

current\_time.hours = (current\_time.hours % 10) + 10\*(key-'0');

}else{

current\_time.hours -= current\_time.hours % 10; // remove ones

current\_time.hours += key - '0'; // add key

};

current\_time.hours = (current\_time.hours - 1) % 12 + 1;

LCDprintc(key); // put the new value to the display

position++; // increase the position tracker

break;

case 'M': // if minutes position

if ((position % 3)==0){ // if tens digit

current\_time.minutes = (current\_time.minutes % 10) + 10\*(key-'0');

}else{

current\_time.minutes -= current\_time.minutes % 10; // remove ones

current\_time.minutes += key - '0'; // add key value

};

current\_time.minutes = current\_time.minutes % 60; // prevent illegal states

LCDprintc(key); // put the new value to the display

position++; // increase the position tracker

break;

case 'S': // if seconds position

if ((position % 3)==0){ // if tens digit

current\_time.seconds = (current\_time.seconds % 10) + 10\*(key-'0');

}else{

current\_time.seconds -= current\_time.seconds % 10; // remove ones

current\_time.seconds += key - '0'; // add key

};

current\_time.seconds = current\_time.seconds % 60; // prevent illegal states

LCDprintc(key); // put the new value to the display

position++; // increase the position tracker

break;

default:

break;

}

break;

}

}

}

}

**“RealTimeClock.h”**

/\*

\* File: RealTimeClock.h

\* Author: Aaron Huinink

\*

\* Created on March 20, 2023, 6:31 PM

\*/

#ifndef REALTIMECLOCK\_H

#define REALTIMECLOCK\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

#include <xc.h>

#include <pic18f57q43.h>

#include "C:\Users\a\_hui\OneDrive - Camosun College\term2\ecet165\_embedded\_mc\labs\lab8\timer0.h"

#include "C:\Users\a\_hui\OneDrive - Camosun College\term2\ecet165\_embedded\_mc\labs\lab6\lcd18f.h"

#include "C:\Users\a\_hui\OneDrive - Camosun College\term2\ecet165\_embedded\_mc\labs\lab8\timer0config.h"

unsigned char nSetRTC = 1;

// typedef struct for time variables

typedef struct realtime{

char hours;

char minutes;

char seconds;

char meridian;

}time;

// ----- setINT ----- //

/\*

\* attaches an interrupt to pin RB4 that allows you to set the RTC clock

\* ARGS: [void]

\* RETURNS: [void]

\*/

// interrupt attach prototype

void \_\_interrupt(irq(IOC),low\_priority)setINT(void);

// interrupt service routine protoype

void setINT(void);

// ----- RTCinit ----- //

/\*

\* initializes a real time clock

\* ARGS: [

\* start\_time<\*time> : pointer to time variable containing the time to begin at;

\* ]

\* RETURNS: [void]

\*/

void RTCrun(time \*start\_time);

#ifdef \_\_cplusplus

}

#endif

#endif /\* REALTIMECLOCK\_H \*/

**“RealTimeClock.c”**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Student Name: Aaron Huinink

File Name: RealTimeClock.c

Description: Implements a real time clock on the lcd display

Marking Check List

Good Comments:

Good Modular Code:

Unsigned Char used correctly:

Format HH:MM:SS Xm correct:

Increments each Second:

10ms Tick correct:

Only valid characters are display:

Clock is accurate:

Am/pm roll over is correct:

Hours per day correct:

Bugs:

Mark:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "RealTimeClock.h"

void RTCrun(time \*start\_time) {

// array of pointers for printf statement

char\* current\_time[4] = {&start\_time->hours, &start\_time->minutes, &start\_time->seconds, &start\_time->meridian};

// enable RTC to run

nSetRTC = 1;

// initialize timer0

timerInit();

// initialize RB4 as input

TRISBbits.TRISB4 = 1; // set RB4 to input

ANSELBbits.ANSELB4 = 0; // set RB4 to digital input

WPUBbits.WPUB4 = 1; // set weak pullup on RB4

// while timerInit enables INTCON0.GIE, IPEN bit must be enabled to prevent

// interrupt conflicts with timer0

INTCON0bits.IPEN = 1;

// attach negative-edge IOC to RB4 button

PIE0bits.IOCIE = 1; // enable interrupt on change

IOCBN4 = 1; // enable negative edge interrupt on RB4

IOCBF = 0x00; // clear port b ioc flags.

void \_\_interrupt(irq(IOC),high\_priority)setINT(void); // attach IOC

// main loop

while(nSetRTC){

// check 10ms timer0 interrupt variable

if(tick\_count > 99){ // if a second has past

tick\_count -= 100; // remove 100 10ms cycles from the count

(start\_time)->seconds++; // increment seconds

if((start\_time)->seconds > 59){ // if a minute has past

(start\_time)->seconds = 0; // rollover seconds

(start\_time)->minutes++; // increment minutes

if((start\_time)->minutes>59){ // if an hour has past

(start\_time)->minutes = 0; // rollover minutes

(start\_time)->hours++; // increment hours

if((start\_time)->hours>11){ // if it's past 11am or 11pm

if((start\_time)->hours >12){ // if it's past midnight or noon

(start\_time)->hours = 1; // rollover hours

}else{ // if its midnight or noon

(start\_time)->meridian = ( // toggle am/pm

((start\_time)->meridian == 'a')? 'p' : // if am then pm

'a' // otherwise am

);

}

}

}

};

// print time to lcd

LCD\_HOME;

LCDprintf("%i2:%i2:%i2 %cm", current\_time);

}

};

};

void setINT(void){

IOCBN4 = 0; // disable RB4 edge interrupt because of switch bounce

IOCBF4 = 0; // clear rb4 ioc flags

T0CON0bits.EN = 0; // disable timer0 interrupt

nSetRTC = 0; // set nSet variable to 0 to break while loop

return;

}