Ryan Clark

Project 2

1. Program status: Working

2. Port number and pin number you have used for input/output

sbit h0 = P1^0; h0-3 are used for the hours

sbit h1 = P1^1;

sbit h2 = P1^2;

sbit h3 = P1^3;

sbit sw0 = P1^4; Switch 0

sbit sw1 = P1^5; Switch 1

sbit sw2 = P1^6; Switch 2

sbit apm = P1^7; am/pm

sbit m0 = P2^0; Minutes are P2^0-5

sbit m1 = P2^1;

sbit m2 = P2^2;

sbit m3 = P2^3;

sbit m4 = P2^4;

sbit m5 = P2^5;

sbit cm0 = P2^6; Current Mode

sbit cm1 = P2^7;

sbit s0 = P3^0; Seconds

sbit s1 = P3^1;

sbit s2 = P3^2;

sbit s3 = P3^3;

sbit s4 = P3^4;

sbit s5 = P3^5;

sbit blink = P3^6; Alarm Blink

3. Program source code

//Project #2 Ryan Clark

#include "Main.H"

#include "Simple\_EOS.H"

#include "Stopwatch.H"

#include "Time.H"

#include "Elapsed\_time.H"

#include "Ports.H"

void main(void)

{

//P1=0xFF;

//P2=0xFF;

//P3=0xFF;

// Inits watch set to clock mode by default

watch\_Init(clock\_state);

time\_Init();

//prepare for elapsed time measurement

Elapsed\_Time\_Init();

// Set up simple EOS (10 ms ticks)

sEOS\_Init\_Timer2(10);

while(1) // Super Loop

{

// Enter idle mode to save power

sEOS\_Go\_To\_Sleep();

}

}

// Project #2, Ryan Clark

#include "Main.H"

#include "Elapsed\_time.H"

#include "Time.H"

#include "Stopwatch.H"

#include "Ports.H"

// Easy to change logic here

#define ON 0

#define OFF 1

static watch\_State watch\_state\_G;

void watch\_Init(const watch\_State Start\_state)

{

watch\_state\_G = Start\_state; // Decide on initial state

}

void watch\_Update(void)

{

switch (watch\_state\_G)

{

case clock\_state: // clock mode by default

{

displaytime(); //displays clock time

cm0=0; //sets mode

cm1=0;

if (sw0 == ON)

{

watch\_state\_G = stop\_state;

sw0 = OFF; // reset to be pressed again

sw1 = OFF;

sw2 = OFF;

}

break;

}

case stop\_state: //stop watch set mode

{

cm0=1;//clockmode

cm1=0;

sw1=OFF;

Elapsed\_Time\_Init(); //resets time to 00:00:00

while(sw0==OFF){

if(sw1==ON){

while(sw1==ON){

Elapsed\_Time\_Update();

edisplaytime(); //displays updated stop watch time hit sw1 to turn off

}

}

if(sw2==ON){

sw2=OFF;

Elapsed\_Time\_Init(); //sets time back to 00:00:00

}

}

if (sw0 == ON) // switches to new state if sw0 is pressed

{

sw1=OFF;

sw2=OFF;

watch\_state\_G = alarm\_state;//changes alarm state

sw0 = OFF; // reset to be pressed again

}

break;

}

case alarm\_state: // alarm set mode

{

cm0=0;

cm1=1; //updates current mode

//time\_Update();

sw1=OFF;

while(sw0==OFF){

if(sw1==ON){ //Press sw1 to enter alarm

alarm();

}

}

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

watch\_state\_G =time\_state; // changes to new state

sw0 = OFF; // reset to be pressed again

}

break;

}

case time\_state: // time set mode

{

cm0=1;//sets mode

cm1=1;

sw1=OFF;

while(sw0==OFF){

if(sw1==ON){ //Press sw1 to enter alarm

changetime();

}

}

if (sw0 == ON)

{

sw1=OFF

sw2=OFF;

watch\_state\_G = clock\_state;

sw0 = OFF; // reset to be pressed again

}

break;

}

}

}

/\*------------------------------------------------------------------\*-

// Project #2, Ryan Clark

-\*------------------------------------------------------------------\*/

#include "Main.H"

#include "Simple\_EOS.H"

#include "Elapsed\_time.H"

#include "Stopwatch.H"

#include "Time.H"

void sEOS\_ISR() interrupt INTERRUPT\_Timer\_2\_Overflow

{

TF2 = 0; // Must manually reset the T2 flag

Elapsed\_Time\_Update();

// Call update function

time\_Update();

watch\_Update();

}

void sEOS\_Init\_Timer2(const tByte TICK\_MS)

{

tLong Inc;

tWord Reload\_16;

tByte Reload\_08H, Reload\_08L;

// Timer 2 is configured as a 16-bit timer,

// which is automatically reloaded when it overflows

T2CON = 0x04; // Load Timer 2 control register

// Number of timer increments required (max 65536)

Inc = ((tLong)TICK\_MS \* (OSC\_FREQ/1000)) / (tLong)OSC\_PER\_INST;

// 16-bit reload value

Reload\_16 = (tWord) (65536UL - Inc);

// 8-bit reload values (High & Low)

Reload\_08H = (tByte)(Reload\_16 / 256);

Reload\_08L = (tByte)(Reload\_16 % 256);

// Used for manually checking timing (in simulator)

//P2 = Reload\_08H;

//P3 = Reload\_08L;

TH2 = Reload\_08H; // Load Timer 2 high byte

RCAP2H = Reload\_08H; // Load Timer 2 reload capt. reg. high byte

TL2 = Reload\_08L; // Load Timer 2 low byte

RCAP2L = Reload\_08L; // Load Timer 2 reload capt. reg. low byte

// Timer 2 interrupt is enabled, and ISR will be called

// whenever the timer overflows.

ET2 = 1;

// Start Timer 2 running

TR2 = 1;

EA = 1; // Globally enable interrupts

}

/\*-----------------------------------------------------------------\*-

sEOS\_Go\_To\_Sleep()

-\*------------------------------------------------------------------\*/

void sEOS\_Go\_To\_Sleep(void)

{

PCON |= 0x01; // Enter idle mode (generic 8051 version)

}

// Project #2, Ryan Clark

#include "Main.H"

#include "Elapsed\_time.H"

#include "Ports.H"

#include <stdio.h>

tByte Houe;//elapsed time variables

tByte Mine;

tByte Sece;

void Elapsed\_Time\_Init(void)

{

Houe = 0;

Mine = 0;

Sece = 0;

s0 = 0;

s1 = 0;

s2 = 0;

s3 = 0;

s4 = 0;

s5 = 0;

//minutes

m0 = 0;

m1 = 0;

m2 = 0;

m3 = 0;

m4 = 0;

m5 = 0;

//hours

h0 = 0;

h1 = 0;

h2 = 0;

h3 = 0;

}

//display

void edisplaytime()

{ // update clock display

s0 = (Sece & (1 << 0))!=0;

s1 = (Sece & (1 << 1))!=0;

s2 = (Sece & (1 << 2))!=0;

s3 = (Sece & (1 << 3))!=0;

s4 = (Sece & (1 << 4))!=0;

s5 = (Sece & (1 << 5))!=0;

//minutes

m0 = (Mine & (1 << 0))!=0;

m1 = (Mine & (1 << 1))!=0;

m2 = (Mine & (1 << 2))!=0;

m3 = (Mine & (1 << 3))!=0;

m4 = (Mine & (1 << 4))!=0;

m5 = (Mine & (1 << 5))!=0;

//hours

h0 = (Houe & (1 << 0))!=0;

h1 = (Houe & (1 << 1))!=0;

h2 = (Houe & (1 << 2))!=0;

h3 = (Houe & (1 << 3))!=0;

if(Houe <12 || Houe == 24)

{

apm = 0;

}

else{

apm = 1;

}

}

void Elapsed\_Time\_Update(void)

{

if (++Sece == 60)

{

Sece = 0;

if (++Mine == 60)

{

Mine = 0;

if (++Houe == 24)

{

Houe = 0;

}

}

}

edisplaytime();

}

//endoffile

// Project #2, Ryan Clark

#include "Main.H"

#include "Time.H"

#include "Ports.H"

#include <stdio.h>

// ------ Public variable definitions ------------------------------

tByte Hou\_G;

tByte Min\_G;

tByte Sec\_G;

tByte tapm;

tByte tHou\_G;

tByte tMin\_G;

tByte tSec\_G;

int temp = 0; /\* Temporary variable for calculations \*/

int temp1=0;

int temp2=0;

#define ON 0

#define OFF 1

void time\_Init(void)

{

Hou\_G = 10;

Min\_G = 30;

Sec\_G = 0;

s0 = 0;

s1 = 0;

s2 = 0;

s3 = 0;

s4 = 0;

s5 = 0;

//minutes

m0 = 0;

m1 = 1;

m2 = 1;

m3 = 1;

m4 = 1;

m5 = 0;

//hours

h0 = 0;

h1 = 1;

h2 = 0;

h3 = 1;

}

//display

void displaytime()

{

// update clock display

s0 = (Sec\_G & (1 << 0))!=0;

s1 = (Sec\_G & (1 << 1))!=0;

s2 = (Sec\_G & (1 << 2))!=0;

s3 = (Sec\_G & (1 << 3))!=0;

s4 = (Sec\_G & (1 << 4))!=0;

s5 = (Sec\_G & (1 << 5))!=0;

//minutes

m0 = (Min\_G & (1 << 0))!=0;

m1 = (Min\_G & (1 << 1))!=0;

m2 = (Min\_G & (1 << 2))!=0;

m3 = (Min\_G & (1 << 3))!=0;

m4 = (Min\_G & (1 << 4))!=0;

m5 = (Min\_G & (1 << 5))!=0;

//hours

h0 = (Hou\_G & (1 << 0))!=0;

h1 = (Hou\_G & (1 << 1))!=0;

h2 = (Hou\_G & (1 << 2))!=0;

h3 = (Hou\_G & (1 << 3))!=0;

if(Hou\_G <12 || Hou\_G == 24)// set to am or pm

{

apm = 0;

}

else{

apm = 1;

}

}

void changetime(void) //Change clock time function

{

while(sw0==OFF){

while(sw1==1){ //0 for am 1 for pm

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

break;

}

apm=sw2;

displaytime();

}

sw1=1; //1 = OFF

while(sw1==1){ // increments hours press switch one to exit loop

if(sw2==0){

Hou\_G++;

sw2=1;

displaytime();

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

break;

}

}

}

sw1=1;

while(sw1==1){ // increments minutes press sw1 to exit

if(sw2==0){

Min\_G++;

sw2=1;

displaytime();

}

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

break;

}

}

sw1=1;

while(sw1==1){ // increments seconds press sw1 to exit

if(sw2==0){

Sec\_G++;

sw2=1;

displaytime();

}

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

break;

}

}

sw1=1;

}

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

}

}

void time\_Update(void) //updates time

{

if (++Sec\_G == 60)

{

Sec\_G = 0;

if (++Min\_G == 60)

{

Min\_G = 0;

if (++Hou\_G == 24)

{

Hou\_G = 0;

}

}

}

displaytime();

}

void alarm(void) //alarm

{

while(sw0==ON){

while(sw1==OFF){ //0 for am 1 for pm

tapm=sw2;

}

sw1=OFF;

while(sw1==OFF){ // increments hours press switch one to exit loop

if(sw2==ON){

tHou\_G++;

sw2=OFF;

}

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

break;

}

}

sw1=OFF;

while(sw1==OFF){ // increments minutes press sw1 to exit

if(sw2==ON){

tMin\_G++;

sw2=OFF;

}

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

break;

}

}

sw1=OFF;

while(sw1==OFF){ // increments seconds press sw1 to exit

if(sw2==ON){

tSec\_G++;

sw2=OFF;

}

if (sw0 == ON)

{

sw1=OFF;

sw2=OFF;

sw0 = OFF; // reset to be pressed again

break;

}

}

sw1=OFF;

if(tapm == apm){

if(tHou\_G==Hou\_G){

if(tMin\_G==Min\_G){

if(tSec\_G==Sec\_G){

if(blink==1) ///Blinks when alarm goes off

{

blink=0;

}

else

{

blink=1;

}

}

}

}

}

}

if (sw0 == ON)

{

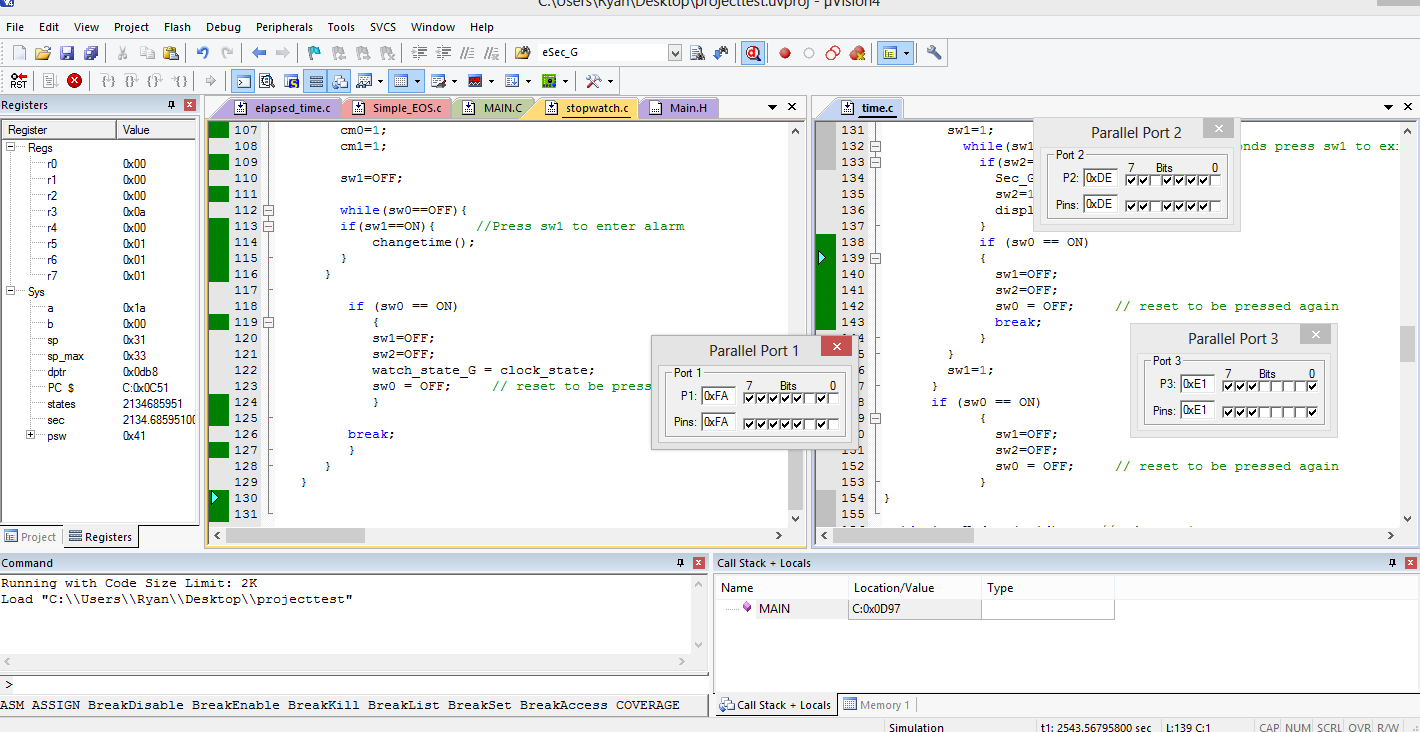
sw1=OFF;

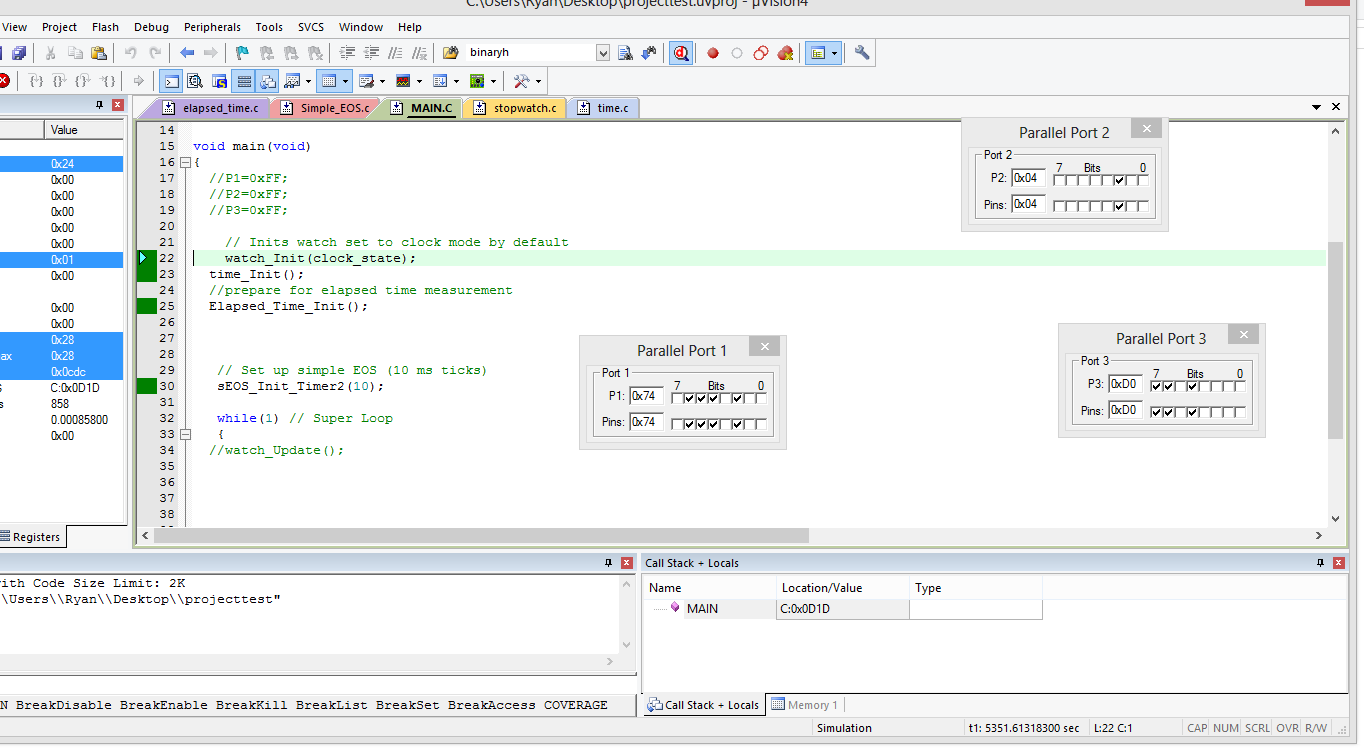
sw2=OFF;

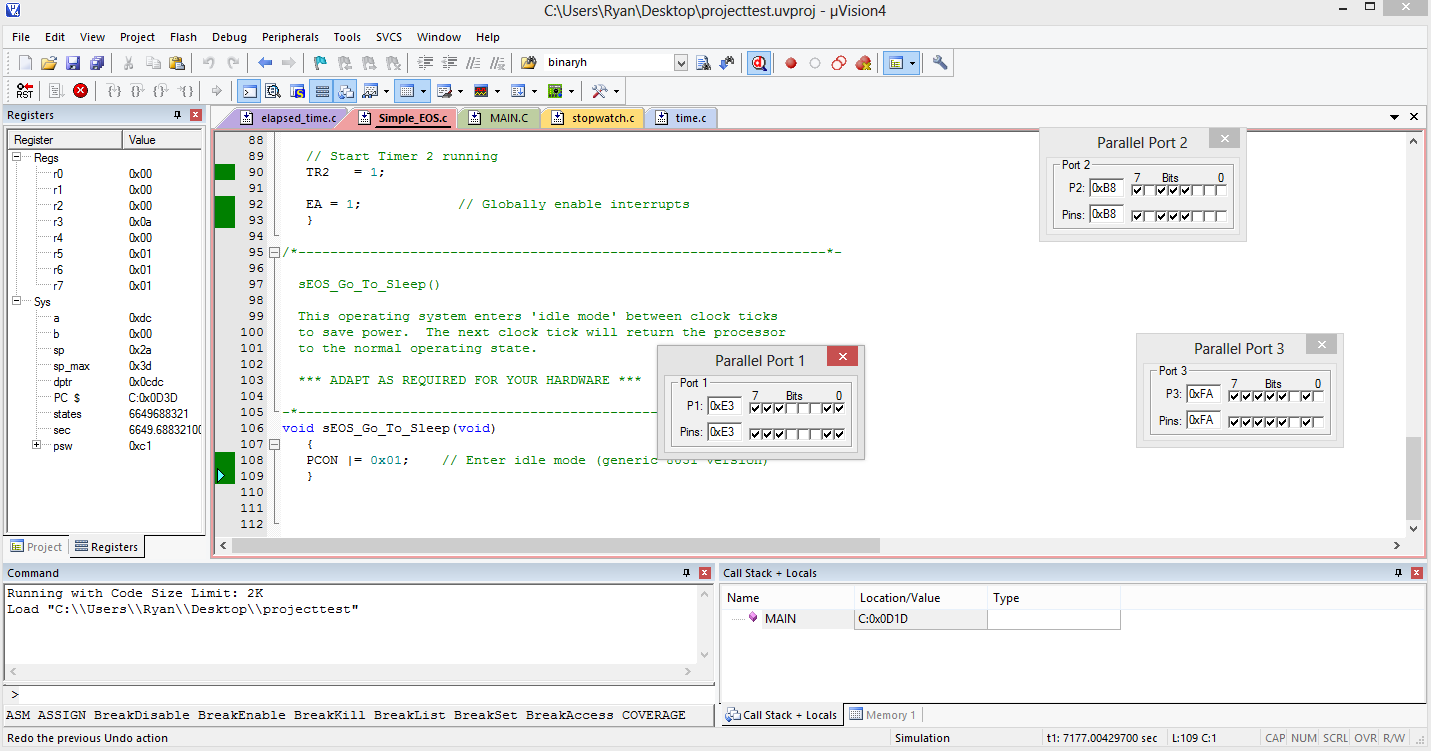
sw0 = OFF; // reset to be pressed again

}

}

// End of file

4. 



5. This project forced me to make use of headers more than I ever have before. I feel like that was a valuable learning experience since most of my CS projects have not made it seem necessary. Keeping all the functions and headers organized helped in keeping the code clear to be able to follow the flow of the program. When I was able to get the clock to finally operate it was extremely satisfying.

At one point I was stuck while using uvision2. I would receive an error but could not find out why it would not load a file. When I switched to uvision4 was able to figure out my file was a little too large and fix it.