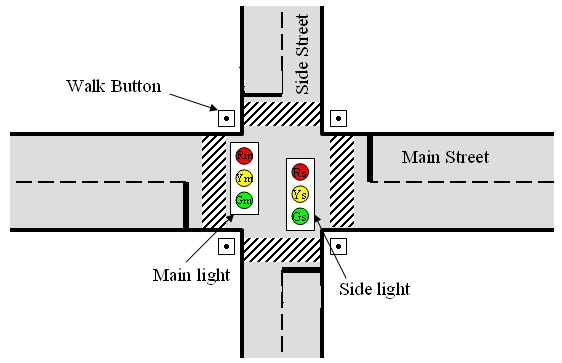
COE 306, Term 171

Introduction to Embedded Systems

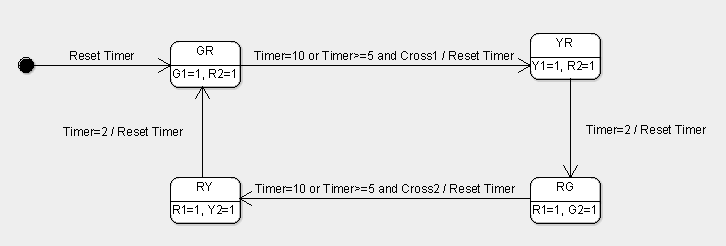
**Assignment# 4 Solution**

**Due date: Saturday, Nov. 25, 2017**

# It is required to design a digital system that controls the traffic lights at an intersection. It receives inputs from all four corners indicating pedestrians that want to cross. In absence of crossing requests, it should allow each direction 10 seconds of green light, followed by 2 seconds of yellow light while the other traffic light will be red light (i.e., for 12 seconds). In presence of crossing requests at or after 5 seconds, immediately proceed with yellow. Use two buttons Cross1 and Cross2 to indicate request for crossing across the main street and side street respectively. Use a pair of Red, Yellow and Green leds for each street.



## Show the state diagram of your traffic light controller.



## Implement your traffic light controller and include your code along with a link for a video demo illustrating its correct functionality.

Two solutions are done one based on using two timers and interrupts and the other is based on using one timer and timer polling in addition to displaying the waiting time on seven segment display.

**First Solution**:

**#ifdef** \_\_USE\_CMSIS

**#include** "LPC17xx.h"

**#endif**

**#include** <cr\_section\_macros.h>

**#define** GR 0

**#define** YR 1

**#define** RG 2

**#define** RY 3

**int** state=0;

**int** Cross1=0, Cross2=0;

**int** timer0=0; timer1=0;

**void** **TIMER0\_IRQHandler**() {

timer0=1;

LPC\_TIM0->IR |= 1;

}

**void** **TIMER1\_IRQHandler**() {

timer1=1;

LPC\_TIM1->IR |= 1;

}

**void** **EINT0\_IRQHandler**()

{

**if** (state==GR) Cross1=1;

printf("Cross1 pressed\n");

**for** (**int** j=0; j<1000000; j++); // to avoid effect of bouncing

LPC\_SC->EXTINT |= 1;

}

**void** **EINT1\_IRQHandler**()

{

**if** (state==RG) Cross2=1;

printf("Cross2 pressed\n");

**for** (**int** j=0; j<1000000; j++); // to avoid effect of bouncing

LPC\_SC->EXTINT |= 2;

}

**void** **SetTimer0**(uint32\_t delayInSec) {

LPC\_TIM0->TCR = 0x02; /\* reset timer \*/

LPC\_TIM0->MR0 = delayInSec\*2000 \* (12500000 / 1000 - 1);

LPC\_TIM0->MCR = 0x05; /\* stop timer on match and enable interrupt\*/

LPC\_TIM0->TCR = 0x01; /\* start timer \*/

}

**void** **SetTimer1**(uint32\_t delayInSec) {

LPC\_TIM1->TCR = 0x02; /\* reset timer \*/

LPC\_TIM1->MR0 = delayInSec\*2000 \* (12500000 / 1000 - 1);

LPC\_TIM1->MCR = 0x05; /\* stop timer on match and enable interrupt\*/

LPC\_TIM1->TCR = 0x01; /\* start timer \*/

}

**int** **main**(**void**) {

LPC\_GPIO0->FIODIR |= 7<<7; // set pins 0.7, 0.8, 0.9 for GYR for Main Street TL

LPC\_GPIO0->FIODIR |= 7<<23; // set pins 0.23, 0.24, 0.25 for GYR for Side Street TL

LPC\_PINCON->PINSEL4 |= (1<<20); // using pin p2.10 for cross1

LPC\_PINCON->PINSEL4 |= (1<<22); // using pin p2.11 for cross2

LPC\_SC->EXTMODE |= 3;

LPC\_SC->EXTPOLAR |= 3;

NVIC\_EnableIRQ(*EINT0\_IRQn*);

NVIC\_EnableIRQ(*EINT1\_IRQn*);

NVIC\_EnableIRQ(*TIMER0\_IRQn*); // Enable interrupt for timer 0

NVIC\_EnableIRQ(*TIMER1\_IRQn*); // Enable interrupt for timer 1

Cross1=0; Cross2=0;

LPC\_TIM0->PR = 0x00; /\* set prescaler to zero \*/

SetTimer0(10);

timer0=0;

LPC\_TIM1->PR = 0x00; /\* set prescaler to zero \*/

SetTimer1(5);

timer1=0;

**while**(1) {

**switch**(state){

**case** GR:

printf("state GR\n");

LPC\_GPIO0->FIOSET |= (1<<7); LPC\_GPIO0->FIOCLR |= (1<<8);

LPC\_GPIO0->FIOCLR |= (1<<9);

LPC\_GPIO0->FIOCLR |= (1<<23); LPC\_GPIO0->FIOCLR |= (1<<24);

LPC\_GPIO0->FIOSET |= (1<<25);

**if** (timer0 || (timer1 && Cross1)){

state = YR; Cross1=0;

SetTimer0(2);

timer0=0;

}

**else**

state = GR;

**break**;

**case** YR:

printf("state YR\n");

LPC\_GPIO0->FIOCLR |= (1<<7); LPC\_GPIO0->FIOSET |= (1<<8);

LPC\_GPIO0->FIOCLR |= (1<<9);

LPC\_GPIO0->FIOCLR |= (1<<23); LPC\_GPIO0->FIOCLR |= (1<<24);

LPC\_GPIO0->FIOSET |= (1<<25);

**if** (timer0){

state = RG;

SetTimer0(10);

timer0=0;

SetTimer1(5);

timer1=0;

}

**else**

state = YR;

**break**;

**case** RG:

printf("state RG\n");

LPC\_GPIO0->FIOCLR |= (1<<7); LPC\_GPIO0->FIOCLR |= (1<<8);

LPC\_GPIO0->FIOSET |= (1<<9);

LPC\_GPIO0->FIOSET |= (1<<23); LPC\_GPIO0->FIOCLR |= (1<<24);

LPC\_GPIO0->FIOCLR |= (1<<25);

**if** (timer0 || (timer1 && Cross2)){

state = RY; Cross2 = 0;

SetTimer0(2);

timer0=0;

}

**else**

state = RG;

**break**;

**case** RY:

printf("state RY\n");

LPC\_GPIO0->FIOCLR |= (1<<7); LPC\_GPIO0->FIOCLR |= (1<<8);

LPC\_GPIO0->FIOSET |= (1<<9);

LPC\_GPIO0->FIOCLR |= (1<<23); LPC\_GPIO0->FIOSET |= (1<<24); LPC\_GPIO0->FIOCLR |= (1<<25);

**if** (timer0){

state = GR;

SetTimer0(10);

timer0=0;

SetTimer1(5);

timer1=0;

}

**else**

state = RY;

**break**;

}

}

**return** 0 ;

}

**Second Solution**:

**#ifdef** \_\_USE\_CMSIS

**#include** "LPC17xx.h"

**#endif**

**#include** <cr\_section\_macros.h>

**#define** GR 0

**#define** YR 1

**#define** RG 2

**#define** RY 3

**int** state=0;

**int** Cross1=0, Cross2=0;

**void** **EINT0\_IRQHandler**()

{

**if** (state==GR) Cross1=1;

printf("Cross1 pressed\n");

LPC\_SC->EXTINT |= 1;

**for** (**int** j=0; j<1000000; j++); // to avoid effect of bouncing

}

**void** **EINT1\_IRQHandler**()

{

**if** (state==RG) Cross2=1;

printf("Cross2 pressed\n");

LPC\_SC->EXTINT |= 2;

**for** (**int** j=0; j<1000000; j++); // to avoid effect of bouncing

}

**int** **main**(**void**) {

LPC\_GPIO2->FIODIR |= 0x7f; // set pins 2.0 to 2.6 for 7-segment display

LPC\_GPIO0->FIODIR |= 7<<7; // set pins 0.7, 0.8, 0.9 for GYR for Main Street TL

LPC\_GPIO0->FIODIR |= 7<<23; // set pins 0.23, 0.24, 0.25 for GYR for Side Street TL

LPC\_PINCON->PINSEL4 |= (1<<20); // using pin p2.10 for cross1

LPC\_PINCON->PINSEL4 |= (1<<22); // using pin p2.11 for cross2

LPC\_SC->EXTMODE |= 3;

LPC\_SC->EXTPOLAR |= 3;

NVIC\_EnableIRQ(*EINT0\_IRQn*);

NVIC\_EnableIRQ(*EINT1\_IRQn*);

Cross1=0; Cross2=0;

**while**(1) {

**switch**(state){

**case** GR:

printf("state GR\n");

LPC\_GPIO0->FIOSET |= (1<<7); LPC\_GPIO0->FIOCLR |= (1<<8);

LPC\_GPIO0->FIOCLR |= (1<<9);

LPC\_GPIO0->FIOCLR |= (1<<23); LPC\_GPIO0->FIOCLR |= (1<<24);

LPC\_GPIO0->FIOSET |= (1<<25);

**for** (i=1; i<=5; i++){

Display(i);

DelaySec(1);

}

**for** (i=6; i<=10; i++){

**if** (Cross1) { Cross1=0; **break**;}

Display(i);

DelaySec(1);

}

state = YR;

**break**;

**case** YR:

printf("state YR\n");

LPC\_GPIO0->FIOCLR |= (1<<7); LPC\_GPIO0->FIOSET |= (1<<8);

LPC\_GPIO0->FIOCLR |= (1<<9);

LPC\_GPIO0->FIOCLR |= (1<<23); LPC\_GPIO0->FIOCLR |= (1<<24);

LPC\_GPIO0->FIOSET |= (1<<25);

**for** (i=1; i<=2; i++){

Display(i);

DelaySec(1);

}

state = RG;

**break**;

**case** RG:

printf("state RG\n");

LPC\_GPIO0->FIOCLR |= (1<<7); LPC\_GPIO0->FIOCLR |= (1<<8);

LPC\_GPIO0->FIOSET |= (1<<9);

LPC\_GPIO0->FIOSET |= (1<<23); LPC\_GPIO0->FIOCLR |= (1<<24);

LPC\_GPIO0->FIOCLR |= (1<<25);

**for** (i=1; i<=5; i++){

Display(i);

DelaySec(1);

}

**for** (i=6; i<=10; i++){

**if** (Cross2) { Cross2=0; **break**;}

Display(i);

DelaySec(1);

}

state = RY;

**break**;

**case** RY:

printf("state RY\n");

LPC\_GPIO0->FIOCLR |= (1<<7); LPC\_GPIO0->FIOCLR |= (1<<8);

LPC\_GPIO0->FIOSET |= (1<<9);

LPC\_GPIO0->FIOCLR |= (1<<23); LPC\_GPIO0->FIOSET |= (1<<24);

LPC\_GPIO0->FIOCLR |= (1<<25);

**for** (i=1; i<=2; i++){

Display(i);

DelaySec(1);

}

state = GR;

**break**;

}

}

**return** 0 ;

}

**void** **Display**(**int** num){

**int** res; // GFEDCBA

**if**(num == 0)

res = 0x3F;

**else** **if**(num == 1)

res = 0x6;

**else** **if**(num == 2)

res = 0x5B;

**else** **if**(num == 3)

res = 0x4F;

**else** **if**(num == 4)

res = 0x66;

**else** **if**(num == 5)

res = 0x6D;

**else** **if**(num == 6)

res = 0x7D;

**else** **if**(num == 7)

res = 0x7;

**else** **if**(num == 8)

res = 0x7F;

**else** **if**(num == 9)

res = 0x6F;

**else** res = 0x3F;

LPC\_GPIO2->FIOPIN = ~res;

}

**void** **DelaySec**(uint32\_t delayInSec) {

delayInSec = delayInSec \* 2000;

LPC\_TIM0->TCR = 0x02; /\* reset timer \*/

LPC\_TIM0->PR = 0x00; /\* set prescaler to zero \*/

LPC\_TIM0->MR0 = delayInSec \* (12500000 / 1000 - 1);

LPC\_TIM0->IR = 0xff; /\* reset all interrrupts \*/

LPC\_TIM0->MCR = 0x04; /\* stop timer on match \*/

LPC\_TIM0->TCR = 0x01; /\* start timer \*/

**while** (LPC\_TIM0->TCR & 0x01);

}

# Write an embedded software that computes the moving average of the last five samples using a circular buffer. Assume that each sample is 4-bit. Whenever a sample is entered by the user, your program should print the average of the last five samples. Use interrupt to indicate that a new sample is entered. Include your code along with a video link illustrating correct functionality of your program for 10 entered samples.

**ifdef** \_\_USE\_CMSIS

**#include** "LPC17xx.h"

**#endif**

**#include** <cr\_section\_macros.h>

**int** x;

**#define** SIZE 5

**int** buffer[SIZE];

**int** pos;

**void** **init**() {

**for** (**int** i = 0; i < SIZE; i++)

buffer[i] = 0;

pos = SIZE - 1;

}

**void** **put**(**int** value) {

pos = (pos + 1) % SIZE;

buffer[pos] = value;

}

/\* get the ith value earlier from the circular buffer; zero being the newest value \*/

**int** **get**(**int** i) {

**int** index = (pos - i) % SIZE;

**if** (index >=0)

**return** buffer[index];

**else** **return** buffer[SIZE+index];

}

**void** **EINT0\_IRQHandler**()

{

x = LPC\_GPIO2->FIOPIN & 0xf; // Using pins 2.0, 2.1, 2.2 and 2.3 for 4-bit input

printf("Entered new value = %d \n", x);

put(x);

**int** sum=0;

printf("Buffer Content: ");

**for** (**int** k=0; k<SIZE;k++){

printf("%d ", buffer[k]);

sum += get(k);

}

printf("\n");

**float** avg = (**float**) sum /SIZE;

printf("Average = %f \n", avg);

LPC\_SC->EXTINT |= 1; //Clear Interrupt

**for** (**int** j=0; j<1000000; j++); // to avoid effect of bouncing

}

**int** **main**(**void**) {

LPC\_PINCON->PINSEL4 |= (1<<20); // using pin p2.10

NVIC\_EnableIRQ(*EINT0\_IRQn*);

init();

**while**(1) {

}

**return** 0 ;

}