HW #5

ECEN 621

Mrinmoy Sarkar

Date: 10/1/2019

LaunchPad Implementation:

```
#include "msp.h"


/**
 * main.c
 */

#define LED_RED BIT0
#define LED_BLUE BIT2
#define S1 BIT1
#define A0 ((P5->IN & S1) == 0x00)
#define B0ON (P1->OUT |= LED_RED)
#define B0OFF (P1->OUT &= ~LED_RED)
#define B1ON (P2->OUT |= LED_BLUE)
#define B1OFF (P2->OUT &= ~LED_BLUE)
#define TIMER_PERIOD 9600000  // equivalent to 200ms

enum DMstates {DMinitstate,DM0,DM1,DM2} DMstate;
enum ILstates {ILinitstate,IL0,IL1,IL2} ILstate;
enum BLstates {BLinitstate,BL0,BL1} BLstate;

unsigned char mnt;
unsigned char cnt;
unsigned char flag;


void SysTick_Init(void);
void initPorts(void);
void TicFctDM(void);
void TicFctIL(void);
void TicFctBL(void);




void main(void)
{
        WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;             // stop watchdog
timer

        DMstate = DMinitstate;
        ILstate = ILinitstate;
```

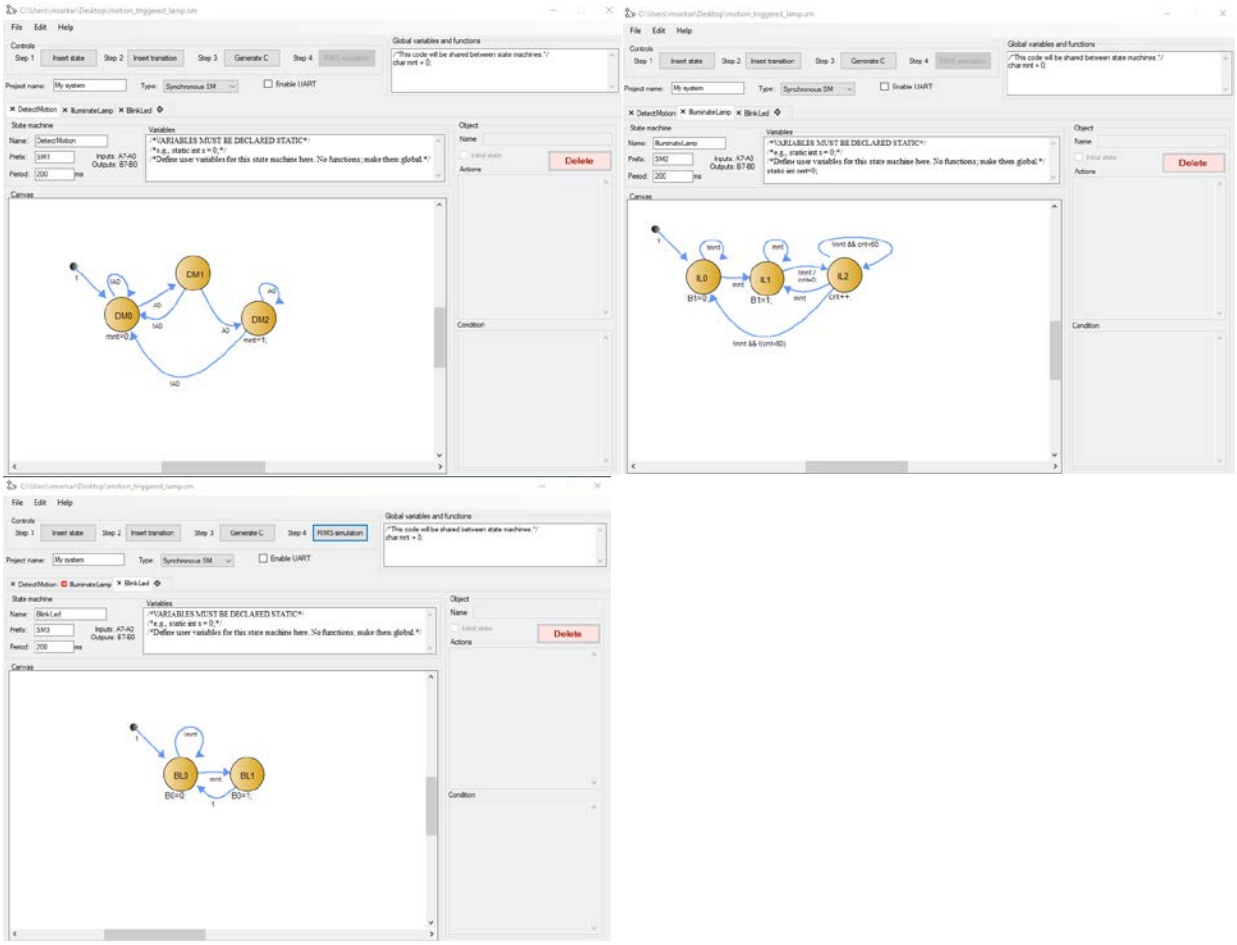
```
        BLstate = BLinitstate;
        mnt = 0;
        cnt = 0;
        flag = 1;
        initPorts();
        SysTick_Init();
        __enable_irq();
        while(1)
        {
          TicFctDM();
          TicFctIL();
          TicFctBL();
          while(flag);
          flag = 1;

        }

}


void SysTick_Handler(void){
  flag = 0;
}

void SysTick_Init(void)
{
   while(PCM->CTL1 & PCM_CTL1_PMR_BUSY);
   PCM->CTL0 = PCM_CTL0_KEY_VAL | PCM_CTL0_AMR_1;
   while(PCM->CTL1 & PCM_CTL1_PMR_BUSY);
   FLCTL->BANK0_RDCTL = (FLCTL->BANK0_RDCTL & ~(FLCTL_BANK0_RDCTL_WAIT_MASK)) |
FLCTL_BANK0_RDCTL_WAIT_1;
   FLCTL->BANK1_RDCTL = (FLCTL->BANK1_RDCTL & ~(FLCTL_BANK1_RDCTL_WAIT_MASK)) |
FLCTL_BANK1_RDCTL_WAIT_1;
   CS->KEY = CS_KEY_VAL;
   CS->CTL0 |= CS_CTL0_DCORSEL_5;
   CS->KEY = 0;

   SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Msk | SysTick_CTRL_ENABLE_Msk;
   SysTick->LOAD = TIMER_PERIOD;
   SysTick->VAL = 0;
   SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;

}
```

```
void initPorts(void)
{
   P5->DIR &= ~S1;
   P5->REN = S1;
   P5->OUT = S1;
   P1->DIR = LED_RED;
   P1->OUT = 0x00;
   P2->DIR = LED_BLUE;
   P2->OUT = 0x00;
}

void TicFctDM(void)
{
   switch(DMstate)
   {
   case DMinitstate:
      DMstate = DM0;
      break;
   case DM0:
      if(A0)
      {
         DMstate = DM1;
      }
      break;
   case DM1:
      if(!A0)
      {
         DMstate = DM0;
      }
      else if(A0)
      {
         DMstate = DM2;
      }
      break;
   case DM2:
      if(!A0)
      {
         DMstate = DM0;
      }
      break;
   }
   switch(DMstate)
   {
```

```
   case DM0:
      mnt=0;
      break;
   case DM1:
      break;
   case DM2:
      mnt=1;
      break;
   }
}



void TicFctIL(void)
{
   switch(ILstate)
   {
   case ILinitstate:
      ILstate = IL0;
      break;
   case IL0:
      if(mnt)
      {
         ILstate = IL1;
      }
      break;
   case IL1:
      if(!mnt)
      {
         ILstate = IL2;
         cnt = 0;
      }
      break;
   case IL2:
      if(mnt)
      {
         ILstate = IL1;
      }
      else if(!mnt && !(cnt<50))
      {
         ILstate = IL0;
      }
      break;
   }
```

```
   switch(ILstate)
   {
   case IL0:
      B1OFF;
      break;
   case IL1:
      B1ON;
      break;
   case IL2:
      cnt++;
      break;
   }
}


void TicFctBL(void)
{
   switch(BLstate)
     {
     case BLinitstate:
        BLstate = BL0;
        break;
     case BL0:
        if(mnt)
        {
           BLstate = BL1;
        }
        break;
     case BL1:
        BLstate = BL0;
        break;
     }
     switch(BLstate)
     {
     case BL0:
        B0OFF;
        break;
     case BL1:
        B0ON;
        break;
     }
}
```

RIBS Model:



Generated C code:

```
/*
This code was automatically generated using the Riverside-Irvine State machine Builder tool
Version 2.8 --- 10/1/2019 18:6:10 PST
*/

#include "rims.h"

/*This code will be shared between state machines.*/
char mnt = 0;
unsigned char TimerFlag = 0;
void TimerISR() {
```

```
  TimerFlag = 1;
}


enum SM1_States { SM1_DM0, SM1_DM1, SM1_DM2 } SM1_State;

TickFct_DetectMotion() {
  /*VARIABLES MUST BE DECLARED STATIC*/
/*e.g., static int x = 0;*/
/*Define user variables for this state machine here. No functions; make them global.*/
  switch(SM1_State) { // Transitions
    case -1:
      SM1_State = SM1_DM0;
      break;
    case SM1_DM0:
      if (A0) {
        SM1_State = SM1_DM1;
      }
      else if (!A0) {
        SM1_State = SM1_DM0;
      }
      break;
    case SM1_DM1:
      if (!A0) {
        SM1_State = SM1_DM0;
      }
      else if (A0) {
        SM1_State = SM1_DM2;
      }
      break;
    case SM1_DM2:
      if (A0) {
        SM1_State = SM1_DM2;
      }
      else if (!A0) {
        SM1_State = SM1_DM0;
      }
      break;
    default:
      SM1_State = SM1_DM0;
    } // Transitions

  switch(SM1_State) { // State actions
    case SM1_DM0:
```

```
      mnt=0;
      break;
    case SM1_DM1:
      break;
    case SM1_DM2:
      mnt=1;
      break;
    default: // ADD default behaviour below
      break;
  } // State actions
}

enum SM2_States { SM2_IL0, SM2_IL1, SM2_IL2 } SM2_State;

TickFct_IlluminateLamp() {
  /*VARIABLES MUST BE DECLARED STATIC*/
/*e.g., static int x = 0;*/
/*Define user variables for this state machine here. No functions; make them global.*/
static int cnt=0;
  switch(SM2_State) { // Transitions
    case -1:
      SM2_State = SM2_IL0;
      break;
    case SM2_IL0:
      if (!mnt) {
        SM2_State = SM2_IL0;
      }
      else if (mnt) {
        SM2_State = SM2_IL1;
      }
      break;
    case SM2_IL1:
      if (mnt) {
        SM2_State = SM2_IL1;
      }
      else if (!mnt) {
        SM2_State = SM2_IL2;
        cnt=0;
      }
      break;
    case SM2_IL2:
      if (!mnt && cnt<50) {
        SM2_State = SM2_IL2;
      }
```

```
      else if (!mnt && !(cnt<50)) {
        SM2_State = SM2_IL0;
      }
      else if (mnt) {
        SM2_State = SM2_IL1;
      }
      break;
    default:
      SM2_State = SM2_IL0;
    } // Transitions

  switch(SM2_State) { // State actions
    case SM2_IL0:
      B1=0;
      break;
    case SM2_IL1:
      B1=1;
      break;
    case SM2_IL2:
      cnt++;
      break;
    default: // ADD default behaviour below
      break;
  } // State actions
}

enum SM3_States { SM3_BL0, SM3_BL1 } SM3_State;

TickFct_BlinkLed() {
  /*VARIABLES MUST BE DECLARED STATIC*/
/*e.g., static int x = 0;*/
/*Define user variables for this state machine here. No functions; make them global.*/
  switch(SM3_State) { // Transitions
    case -1:
      SM3_State = SM3_BL0;
      break;
    case SM3_BL0:
      if (mnt) {
        SM3_State = SM3_BL1;
      }
      else if (!mnt) {
        SM3_State = SM3_BL0;
      }
      break;
```

```
    case SM3_BL1:
      if (1) {
        SM3_State = SM3_BL0;
      }
      break;
    default:
      SM3_State = SM3_BL0;
    } // Transitions

  switch(SM3_State) { // State actions
    case SM3_BL0:
      B0=0;
      break;
    case SM3_BL1:
      B0=1;
      break;
    default: // ADD default behaviour below
      break;
  } // State actions
}
int main() {
  B = 0; //Init outputs
  TimerSet(200);
  TimerOn();
  SM1_State = -1;
  SM2_State = -1;
  SM3_State = -1;
  while(1) {
    TickFct_DetectMotion();
    TickFct_IlluminateLamp();
    TickFct_BlinkLed();
    while (!TimerFlag);
    TimerFlag = 0;
  }
}
```