HW #5

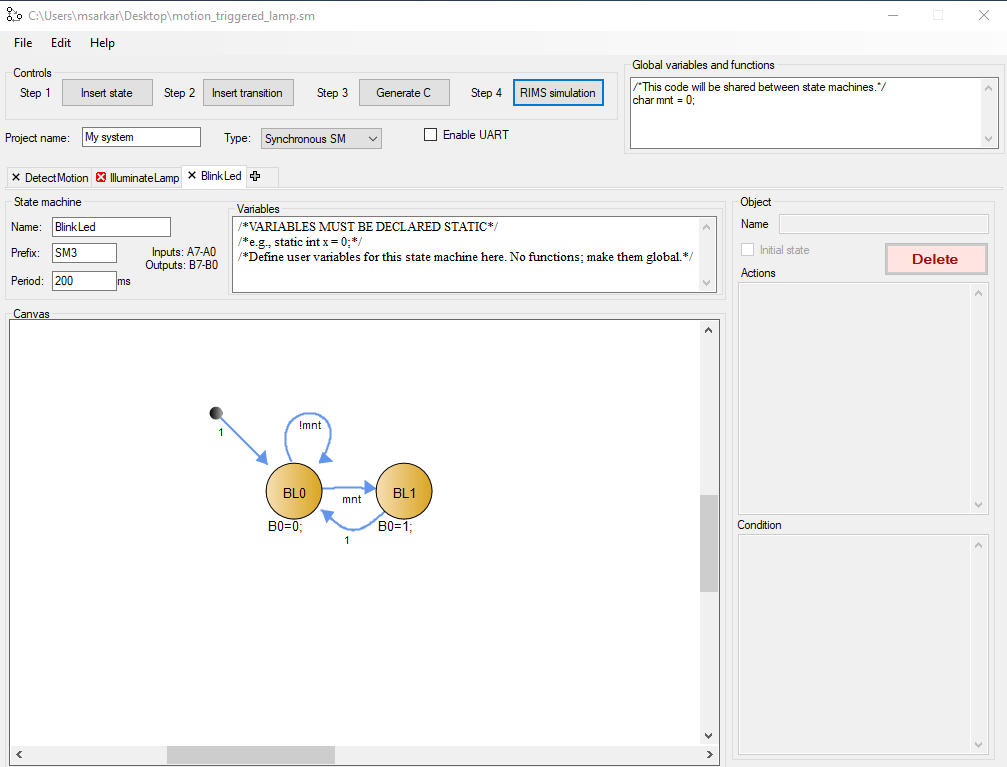
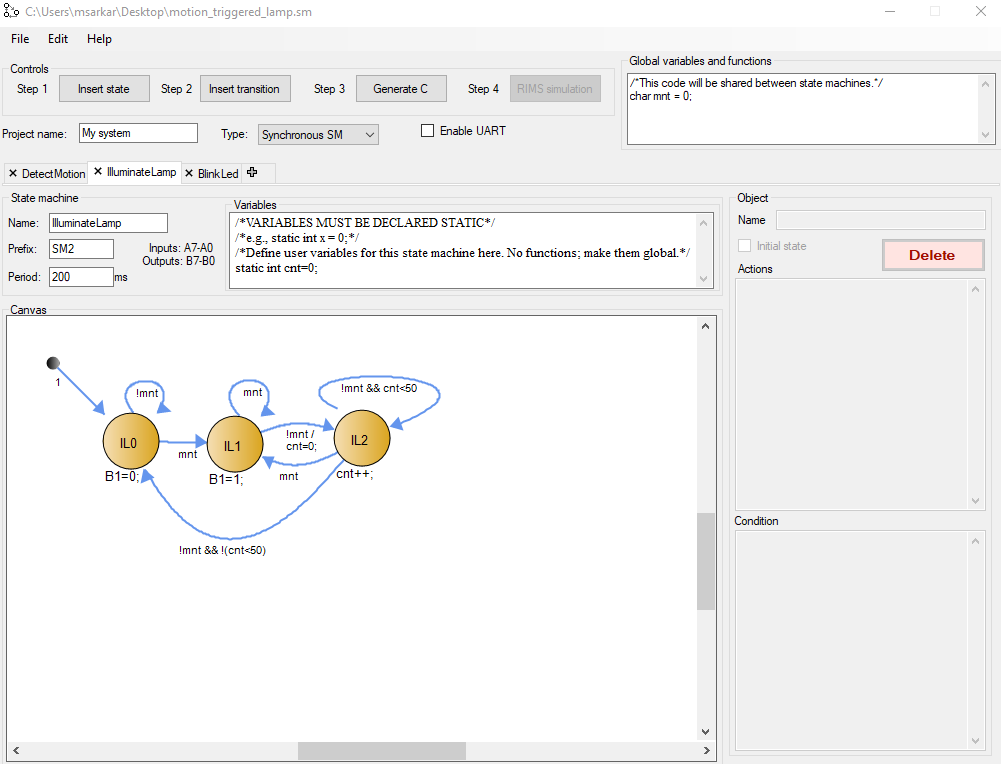
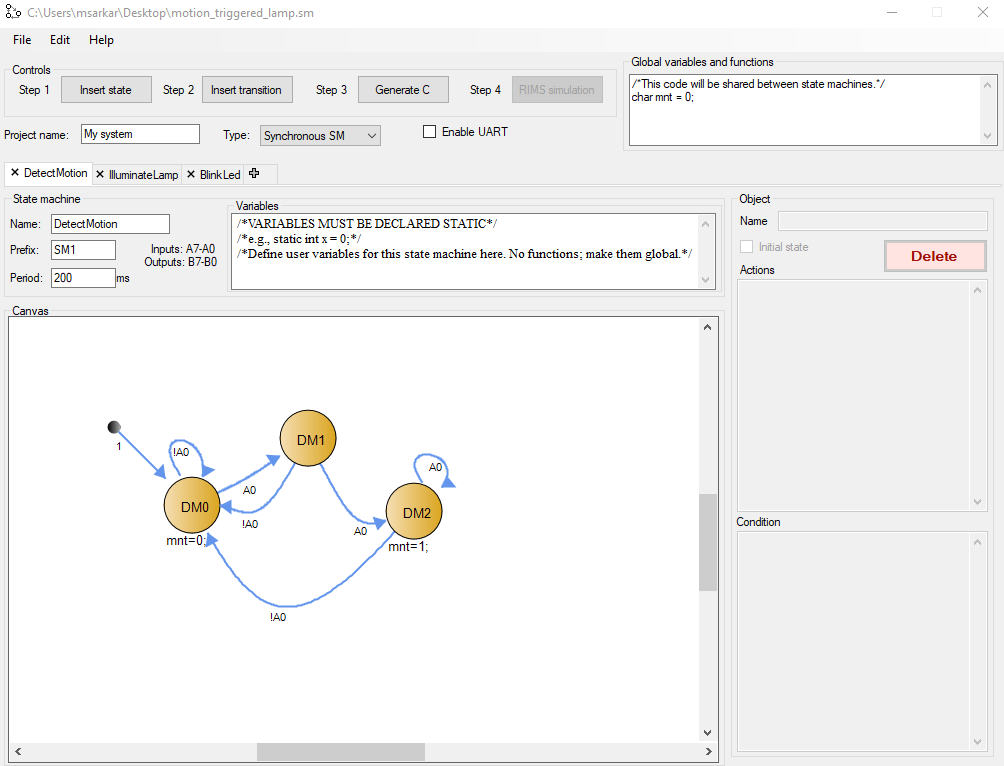
ECEN 621

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;  }  } |