```
//
*****************************************************************************************************************
*
// Programa: Simultaneo y Alternancia para bombas
// Autor(es): Brandon Castro.
// Version: 0.8.4
//
*****************************************************************************************************************
*
// Fecha: 14-02-2024
//
*****************************************************************************************************************
*
// Declaración de Constantes Generales.
//
*****************************************************************************************************************
*

#define LED LATA.F4 // LED de la placa en A4
#define M1 LATA.F5 // Actuador 1 en A5
#define M2 LATE.F0 // Actuador 2 en R0
#define M3 LATE.F1 // Actuador 3 en R1
#define M4 LATE.F2 // Actuador 4 en R2
#define SWITCH1 PORTC.F0 // Pastilla en C0
#define SWITCH2 PORTC.F1
#define TRUE 1
#define FALSE 0
#define RESET asm{reset} // Por si necesitamos reiniciar el PIC en alguna parte

//*****************************************************************
// Variables
//*****************************************************************

bit flag01;    //
bit flag02;    // Banderas varias
bit flag_init; //
bit clock0; // Bandera de reloj
bit interruptC0; // flag interrupcion en C0
bit interruptC1; // flag interrupcion en C1
bit sn_PosEdge_1; // Bandera de señal para transicion positiva en C0
bit sn_PosEdge_2; // Bandera de señal para transicion positiva en C1
bit sn_NegEdge_1; // Bandera de señal para transicion negativa en C0
bit sn_NegEdge_2; // Bandera de señal para transicion negativa en C1
bit once; // Bandera para lazo de control
bit GT1; // Bandera de señal para grupo de trabajo 1
bit GT2; // Bandera de señal para grupo de trabajo 1
bit GT3; // Bandera de señal para grupo de trabajo 1
bit sn_GoTo; // Bandera de señal para señal intermedia
int i;  //
volatile int counter = 0; // Contador
int last_state = 0; // Basura
short unsigned int sm_state = 0;      // Basura
short unsigned int state = 0;         // Variable de barrido de la FSM
short unsigned int current_state = 0; // Basura
short unsigned int next_state;        // Basura
short unsigned int cases = 0;         // Basura
int temp = 0; // Temporal?

//*****************************************************************
// Prototipos
//*****************************************************************

void InitMCU(); // Configuracion inicial MCU
void InitInterrupt(); // Configuracion interrupciones MCU
void State(); // FSM
void Events(); // Rutina de decision sentido de flanco
```

```
int blink(int *_next_state);

//*********************************************************************
// Rutina de interrupcion
//*********************************************************************

void interrupt(){
     temp = PORTC;
     temp = temp << 6;
     if(PIR0.TMR0IF){
          TMR0H = 0xE8;       // Timer para cada segundo y medio?
          TMR0L = 0x49;       //
          PIR0.TMR0IF = 0;
          counter++;
          if(counter >= 2){
               clock0 = 1;
               counter = 0;
          }
     }
     // Tenemos bandera de IOC en C0? y el bit de enable en IOC esta en 1?
     if((IOCCF.B0 == 1) && (IOCIE_bit == 1)){
          IOCCF.B0 = 0; // Limpiamos la bandera de IOC
          interruptC0 = 1; // Ponemos en 1 la bandera de interrupcion en C0
     }
     // Tenemos bandera de IOC en C1? y el bit de enable en IOC esta en 1?
     if((IOCCF.B1 == 1) && (IOCIE_bit == 1)){
          IOCCF.B1 = 0; // Limpiamos la bandera de IOC
          interruptC1 = 1; // Ponemos en 1 la bandera de interrupcion en C0
     }

}

//*********************************************************************
// Programa principal
//*********************************************************************

void main(){

     InitMCU();       // MCU pin/reg config
     InitInterrupt(); // MCU interrupt config

     /* Supuestamente el siguiente if es para iniciar las banderas de los
      * Grupos de trabajo en cuanto inicie el PIC, en teoria nadamas se ejecuta una vez
      * pero se ejecuta una vez tras otra (razon no se) y el problema es no poder
      * declarar la bandera desde el inicio (o maybe con un define?) pero tmbn el
      * problema es que si quito los registros dentro de la funcion State no se ejecuta
      * el programa como lo hace ahora (se hace un barrido entre los estados 1, 2 y 3)
      */
     if(flag_init){
          GT1 = 1;
          GT2 = 0;
          GT3 = 3;
          flag_init = 0;
     }

     // while loop
     do{
          Events(); // Initialize
          State();  // functions
     }while(1);

}

//*********************************************************************
// Clock signal from TMR0 (For testing purposes, must delete later)
//*********************************************************************
```

```
int blink(int *_next_state){
     if(clock0){
          if(state == next_state){
          }
          else{
               state = next_state;
          }
          LED = 0;
          clock0 = 0;
     }
     return state;
}

//*****************************************************************
// FSM
//*****************************************************************

void State(){

     blink(next_state);

     switch(state){
          case 0: // S0 - Todo apagado
               M1 = 0;
               M2 = 0;
               M3 = 0;
               M4 = 1;
               sn_GoTo = 0;
               // Tenemos señal de flanco positivo 1?
               if((sn_PosEdge_1 == 1) && (clock0 == 1)){
                    next_state = 6; // Si, pasamos a estado 6
               }
               else{
                    //next_state = 0; (Revisar de nuevo)
               }
               break;
          case 1: // S1 - Grupo de trabajo 1 110
               M1 = 1;
               M2 = 1;
               M3 = 0;
               GT1 = 1;
               GT2 = 0; // Si comentarizo esto se rompe el codigo
               GT3 = 0; // (why tho???)
               // Tenemos señal de flanco negativo 1?
               if((sn_NegEdge_1 == 1) && (clock0 == 1)){
                    // Si, pasamos a estado 5
                    next_state = 0;
                    //sn_GoTo = 1; // Ponemos en 1 la señal de transicion
               }
               // Tenemos señal de flanco positivo 2?
               else if((sn_PosEdge_2 == 1) && (clock0 == 1)){
                    // Si, pasamos a estado 4
                    next_state = 4;
               }
               // Si no,
               else{
                    // Quedate en estado 1
                    //next_state = 1; (Revisar de nuevo)
               }
               break;
          case 2: // S2 - Grupo de trabajo 2 011
               M1 = 0;
               M2 = 1;
               M3 = 1;
               GT1 = 0; // Trouble
```

```
            GT2 = 1;
            GT3 = 0; // Here comes trouble
            // Tenemos señal de flanco negativo 1?
            if((sn_NegEdge_1 == 1) && (clock0 == 1)){
                  // Si, pasamos a estado 5
                  next_state = 0;
            }
            // Tenemos señal de flanco positivo 2?
            else if((sn_PosEdge_2 == 1) && (clock0 == 1)){
                  // Si, pasamos a estado 4
                  next_state = 4;
            }
            // Si no,
            else{
                  // Quedate en estado 2
                  //next_state = 2; (Revisar de nuevo)
            }
            break;
      case 3: // S3 - Grupo de trabajo 3 101
            M1 = 1;
            M2 = 0;
            M3 = 1;
            GT1 = 0; // Way way more
            GT2 = 0; // trouble
            GT3 = 1;
            // Tenemos señal de flanco negativo 1?
            if((sn_NegEdge_1 == 1) && (clock0 == 1)){
                  // Si, pasamos a estado 5
                  next_state = 0;
            }
            // Tenemos señal de flanco positivo 2?
            else if((sn_PosEdge_2 == 1) && (clock0 == 1)){
                  // Si, pasamos a estado 4
                  next_state = 4;
            }
            // Si no,
            else{
                  // Quedate en estado 3
                  //next_state = 3; (Revisar de nuevo)
            }
            break;
      case 4: // S4 - Grupo de trabajo 4 111
            M1 = 1;
            M2 = 1;
            M3 = 1;
            // Tenemos señal de flango negativo 2?
            if((sn_NegEdge_2 == 1) && (clock0 == 1)){
                  // Si, pasamos a estado 5
                  next_state = 5;
                  sn_GoTo = 1; // Ponemos en 1 la señal de transicion
            }
            // Si no,
            else{
                  // Quedate en estado 4
                  //next_state = 4; (Revisar de nuevo)
            }
            break;
      case 5: // S5 - Estado de transicion para flanco negativo 2
            // Tenemos señal de transicion?
            if((sn_GoTo == 1) && (GT1 == 1) && (clock0 == 1)){
                  next_state = 2;
            }
            else if((sn_GoTo == 1) && (GT2 == 1) && (clock0 == 1)){
                  next_state = 3;
            }
            else if((sn_GoTo == 1) && (GT3 == 1) && (clock0 == 1)){
```

```
                    next_state = 1;
               }
               // Si no,
               else{
                    // Regresamos a estado 4
                    //next_state = 4; (Revisar de nuevo)
               }
               break;
          case 6: // S6 - Estado de transicion para flanco positivo
               M4 = 0;
               if(sn_PosEdge_1){
                    // Tenemos señal de GT1 y GT2 junto con GT3 en 0?
                    if((GT1 == 1) && (clock0 == 1)){
                         // Si, pasa a estado 2
                         next_state = 2;
                         GT2 = 0; // DO NOT
                         GT3 = 0; // DELETE !!!!
                    }
                    // // Tenemos señal de GT2 y GT1 junto con GT3 en 0?
                    else if((GT2 == 1) && (clock0 == 1)){
                         // Si, pasa a estado 3
                         next_state = 3;
                         GT1 = 0; // DO NOT
                         GT3 = 0; // DELETE !!!!
                    }
                    // Tenemos señal de GT3 y GT1 junto con GT2 en 0?
                    else if((GT3 == 1) && (clock0 == 1)){
                         // Si, pasa a estado 1
                         next_state = 1;
                         GT1 = 0; // DO NOT
                         GT2 = 0; // DELETE !!!!
                    }
                    // Si no,
                    else{
                         // Pasa a estado 6
                         //next_state = 6; (Revisar de nuevo)
                    }
               }
               break;
     }

}

//*****************************************************************
// Rutina de decision sentido de flanco
//*****************************************************************

void Events(){
     // Tenemos señal de bandera de interrupcion en C0?
     if(interruptC0){
          // Si, el estado de SWITCH1 es 1?
          if(SWITCH1 == 1){
               // Si, ponemos en 0 la señal de flanco positivo 1 y en 1 la de flanco negativo 1
               sn_PosEdge_1 = 0;
               sn_NegEdge_1 = 1;
          }
          // Si, el estado de SWITCH1 es 0?
          else{
               // Si, ponemos en 1 la señal de flanco positivo 1 y en 0 la de flanco negativo 1
               sn_PosEdge_1 = 1;
               sn_NegEdge_1 = 0;
          }
          interruptC0 = 0; // Limpiamos la bandera de interrupcion en C0
     }
     // Tenemos señal de bandera de interrupcion en C1?
     else if(interruptC1){
```

```
          // Si, el estado de SWITCH2 es 1?
          if(SWITCH2 == 1){
               // Si, ponemos en 0 la señal de flanco positivo 2 y en 1 la de flanco negativo 2
               sn_PosEdge_2 = 0;
               sn_NegEdge_2 = 1;
          }
          // Si, el estado de SWITCH2 es 0?
          else{
               // Si, ponemos en 1 la señal de flanco positivo 2 y en 0 la de flanco negativo 2
               sn_PosEdge_2 = 1;
               sn_NegEdge_2 = 0;
          }
          interruptC1 = 0; // Limpiamos la bandera de interrupcion en C1
     }

}

//*********************************************************************
// Setup bits de configuracion interrupt
//*********************************************************************

void InitInterrupt(){

     PIE0 = 0x30;    // Enable bit de IOC (Interrupt on Change)
     PIR0 = 0x00;    // Limpiamos la bandera de IOC
     T0CON0 = 0x90;
     T0CON1 = 0x46;
     TMR0H = 0xE8;
     TMR0L = 0x49;
     IOCCN = 0x03;   // Activamos las banderas de IOC en Transicion negativa para C0 y C1
     IOCCP = 0x03;   // Activamos las banderas de IOC en Transicion positiva para C0 y C1
     IOCCF = 0x00;   // Limpiamos la bandera de IOC
     PIR0.TMR0IF = 0;
     INTCON = 0xC0;  // Activamos bits de interrupt globales (GIE) y por perifericos (PIE)

}

//*********************************************************************
// Setup del MCU
//*********************************************************************

void InitMCU(){

     ADCON1 = 0x0F; // Desactivamos ADC
     ANSELC = 0;    // Ponemos en modo digital al puerto C
     ANSELE = 0;    //                 ''                 E
     ANSELA = 0;    //                 ''                 A

     TRISC = 0x03;  // Ponemos en modo de entrada a C0 y C1, los demas como salida
     TRISE = 0x00;  // Ponemos en modo salida al puerto E
     TRISA = 0x80;  //                 ''                A

     PORTC = 0x00;  // Ponemos en linea baja en puerto C
     PORTE = 0x00;  //                 ''             E
     PORTA = 0x10;  // Ponemos en linea alta en A4

     LATC = 0x00;   // Dejamos en cero el registro del puerto C
     LATE = 0x00;   //                 ''                      E
     LATA = 0x10;   // Dejamos en 1 al pin A4

     WPUC = 0x03;   // Activamos el pull-up interno de C0 y C1
     INLVLC = 0x03; // Desactivamos valores TTL para C0 y C1 asumiento valores CMOS
     CM1CON0 = 0x00; // Desactivamos el comparador 1
     CM2CON0 = 0x00; // Desactivamos el comparador 2

     once = TRUE;   // Seteo de la condicion para lazo
```

```
}
```