```
//*****************************************************************************************************************
// Programa: Simultaneo y Alternancia para bombas
// Autor(es): Brandon Castro.
// Version: 0.8.2
//*****************************************************************************************************************
// Fecha: 09-02-2024
//*****************************************************************************************************************
// 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 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;  // TBD
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

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

void interrupt(){
     temp = PORTC;
     temp = temp << 6;
     // 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();       // Configuraciones iniciales del MCU
     InitInterrupt(); //       ''        de interrupciones del MCU
     once = TRUE;     // Seteo de la condicion del lazo
     flag_init = 1;

     // Lazo infinito
     do{
          Events(); // Iniciamos las
          State();  // funciones
     }while(1);
}

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

void State(){

     switch(state){
          case 0: // S0 - Todo apagado
               M1 = 0;
               M2 = 0;
               M3 = 0;
               sn_GoTo = 0;
               // Todos los grupos en 0?
               if((GT1 == 0) && (GT2 == 0) && (GT3 == 0)){
                    GT3 = 1; // Si, iniciamos en GT3 (para pasar a GT1)
               }
               // Tenemos señal de flanco positivo 1?
               if(sn_PosEdge_1){
                    state = 7; // Si, pasamos a estado 7
               }
               break;
          case 1: // S1 - Grupo de trabajo 1 110
               M1 = 1;
               M2 = 1;
               M3 = 0;
               GT1 = 1;
               GT2 = 0; // Ponemos en 0 todos los demas GT's
               GT3 = 0; // para alternar en estado 7
               sn_GoTo = 0;
               // Tenemos señal de flanco negativo 1?
               if(sn_NegEdge_1){
                    // Si, pasamos a estado 5
                    state = 5;
                    sn_GoTo = 1; // Ponemos en 1 la señal de transicion
               }
               // Tenemos señal de flanco positivo 2?
               if(sn_PosEdge_2){
                    // Si, pasamos a estado 4
```

```
                state = 4;
            }
            break;
        case 2: // S2 - Grupo de trabajo 2 011
            M1 = 0;
            M2 = 1;
            M3 = 1;
            GT1 = 0; // Ponemos en 0 todos los
            GT2 = 1;
            GT3 = 0; // demas GT's para alternar en estado 7
            sn_GoTo = 0;
            // Tenemos señal de flanco negativo 1?
            if(sn_NegEdge_1){
                // Si, pasamos a estado 5
                state = 5;
                sn_GoTo = 1; // Ponemos en 1 la señal de transicion
            }
            // Tenemos señal de flanco positivo 2?
            if(sn_PosEdge_2){
                // Si, pasamos a estado 4
                state = 4;
            }
            break;
        case 3: // S3 - Grupo de trabajo 3 101
            M1 = 1;
            M2 = 0;
            M3 = 1;
            GT1 = 0; // Ponemos en 0 todos los demas GT's
            GT2 = 0; // para alternar en estado 7
            GT3 = 1;
            sn_GoTo = 0;
            // Tenemos señal de flanco negativo 1?
            if(sn_NegEdge_1){
                // Si, pasamos a estado 5
                state = 5;
                sn_GoTo = 1; // Ponemos en 1 la señal de transicion
            }
            // Tenemos señal de flanco positivo 2?
            if(sn_PosEdge_2){
                // Si, pasamos a estado 4
                state = 4;
            }
            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){
                // Si, pasamos a estado 6
                state = 6;
                sn_GoTo = 1; // Ponemos en 1 la señal de transicion
            }
            break;
        case 5: // S5 - Estado de transicion para flanco negativo 1
            // Tenemos señal de transicion?
            if(sn_GoTo){
                // Si, pasamos a estado 0
                state = 0;
            }
            break;
        case 6: // S6 - Estado de transicion para flanco negativo 2
            // Tenemos señal de transicion?
            if(sn_GoTo){
                // Tenemos señal de GT1?
                if(GT1){
                    // Si, pasa a estado 2
```

```
                    state = 2;
               }
               // Tenemos señal de GT2?
               if(GT2){
                    // Si, pasa a estado 3
                    state = 3;
               }
               // Tenemos señal de GT3?
               if(GT3){
                    // Si, pasa a estado 1
                    state = 1;
               }
          }
          break;
     case 7: // S7 - Estado de transicion para flanco positivo
          // Tenemos señal de GT1 y GT2 junto con GT3 en 0?
          if((GT1 == 1) && (GT2 == 0) && (GT3 == 0)){
               // Si, pasa a estado 2
               state = 2;
               GT2 = 0; // Apaga la señal de GT1 (pq lo puse?)
               GT3 = 0;
          }
          // Tenemos señal de GT2 y GT1 junto con GT3 en 0?
          if((GT1 == 0) && (GT2 == 1) && (GT3 == 0)){
               // Si, pasa a estado 3
               state = 3;
               GT1 = 0;
               GT3 = 0;
          }
          // Tenemos señal de GT3 y GT1 junto con GT2 en 0?
          if((GT1 == 0) && (GT2 == 0) && (GT3 == 1)){
               // Si, pasa a estado 1
               state = 1;
               GT1 = 0;
               GT2 = 0;
          }
     }

}

//*********************************************************************
// 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?
          if(SWITCH1 == 0){
               // 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?
     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?
        if(SWITCH2 == 0){
            // 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 interupt
//**********************************************************************

void InitInterrupt(){

    PIE0 = 0x30;    // Enable bit de IOC (Interrupt on Change)
    PIR0 = 0x00;    // Limpiamos la bandera de IOC
    /*T0CON0 = 0x90;
    T0CON1 = 0x46;
    TMR0H = 0x9C;
    TMR0L = 0x40;*/
    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
    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;
    CM2CON0 = 0x00;

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

}
```