Instituto Politécnico Nacional

Escuela Superior de Cómputo

Introducción a los Microcontroladores

Montes Meza Adbel Anahí

“Ejercicios Básicos con un PIC”

Práctica 1

3CM13

Equipo 10

Montaño Reyes Jennifer

Pardo Alanis Arturo Isaac

Ortega Alcocer Humberto Alejandro

21 de Noviembre del 2022

Práctica 1 (Índice)

1. Objetivo.

En esta práctica se busca realizar 3 ejercicios básicos con nuestro microcontrolador, el *dsPIC30F3013*. Estos ejercicios tendrán como objetivo general mostrar las capacidades, cualidades y funcionamiento del dispositivo, así como el flujo de trabajo que se deberá seguir con el microcontrolador para llevar a cabo el desarrollo de aplicaciones y módulos a futuro.

Cada uno de los ejercicios planteados por la profesora cuentan con un objetivo particular, sin embargo, en lo general podemos decir que se buscará establecer flujos de ejecución con su circuitería asociada para llevar a cabo la operación de un programa simple de E/S, una calculadora y el uso de un *display* de siete segmentos de cátodo común.

1.1. Objetivo del ejercicio uno.

En este ejercicio se buscará leer una entrada a partir de un *dipswitch* y mostrar la misma información en un arreglo de leds correspondiente. El microcontrolador actuará como un “replicador”, en tanto el dispositivo leerá la entrada, realizará un corrimiento de bits para ajustar el dato a los puertos disponibles y, finalmente, la mostrará en los LEDs correspondientes.

1.2. Objetivo del ejercicio dos.

Este ejercicio consistirá en realizar una calculadora básica. Nuestro microcontrolador deberá recibir un *código de operación* (previamente definido por la profesora) con el cuál se procederá a realizar la operación en cuestión. Los códigos de operación definidos son:

Código de Operación (Binario)	**Operación**
00	Suma
01	Resta
10	Multiplicación
11	División

Así, nuestro microcontrolador leerá en dos puertos la combinación del código de operación y realizará la operación en cuestión con un valor fijo en el código. Este ejercicio cuenta con dos variaciones (que serán llamadas A y B), las cuales difieren en el modo de realizar las operaciones particulares de Multiplicación y División.

1.2.1. Objetivo del ejercicio dos (versión A)

En esta versión usaremos las funciones integradas en el lenguaje ensamblador para realizar la Multiplicación y División, es decir: `MUL` y `DIV.`

1.2.2. Objetivo del ejercicio dos (versión B)

En esta versión usaremos corrimientos de bits para realizar las operaciones ya que, como sabemos, un corrimiento *a la izquierda* es equivalente a realizar una multiplicación, y un corrimiento *a la derecha* es equivalente a realizar una división.

1.3. Objetivo del ejercicio tres

En este ejercicio se desarrollará un programa en ensamblador que muestre el Número de Boleta del IPN de algún integrante del equipo basándose en un código de operación de entrada. Para esto, hemos seleccionado el número de boleta de ***Humberto Alcocer*** cuyo número de boleta es ***2016630495***.

Usaremos un dipswitch para introducir el código del número a mostrar y se mostrará, en un display de siete segmentos de cátodo común, el dígito correspondiente del número de boleta en cuestión.

La tabla que define el código asociado a cada dígito en el número de la boleta será la siguiente:

Código (Binario)	Dígito de la Boleta
0000	2
0001	0
0010	1
0011	6
0100	6
0101	3
0110	0
0111	4
1000	9
1001	5

1.4. Objetivo del ejercicio cuatro

En este ejercicio se tomará como base el ejercicio #3 (mostrar el número de boleta de un integrante del equipo) pero añadiendo retardos entre cada “paso” del proceso para así no depender del *dip-switch* y que se muestre todo seguido.

2. Lista de Materiales

Para la realización de la práctica se requieren de distintos materiales tanto físicos (hardware) como virtuales o lógicos (software). El motivo de esto es que realizaremos el flujo de trabajo desde la programación (usando un IDE proporcionado por la profesora) hasta el armado del circuito que mostrará la ejecución del código de manera práctica.

2.1. Materiales Físicos (Hardware)

Los materiales requeridos para esta práctica son:

- 1 DSPIC30f3013.
- 1 Pickit 3.
- 1 Zócalo de programación.
- 14 Resistencias de 1kOhm.
- 1 Display de siete segmentos de cátodo común.
- 1 Dip switch de 4 bits.
- 2 Push button.
- 7 LEDs.
- Cable tipo jumper (a discreción) para pruebas y *debug*.
- Cable electrónico (a discreción).
- Fuente de alimentación 5V.
- Protoboard.
- Computadora Personal con puerto USB.

2.2. Materiales Virtuales (Software)

Los elementos virtuales o lógicos empleados para el desarrollo de la práctica son:

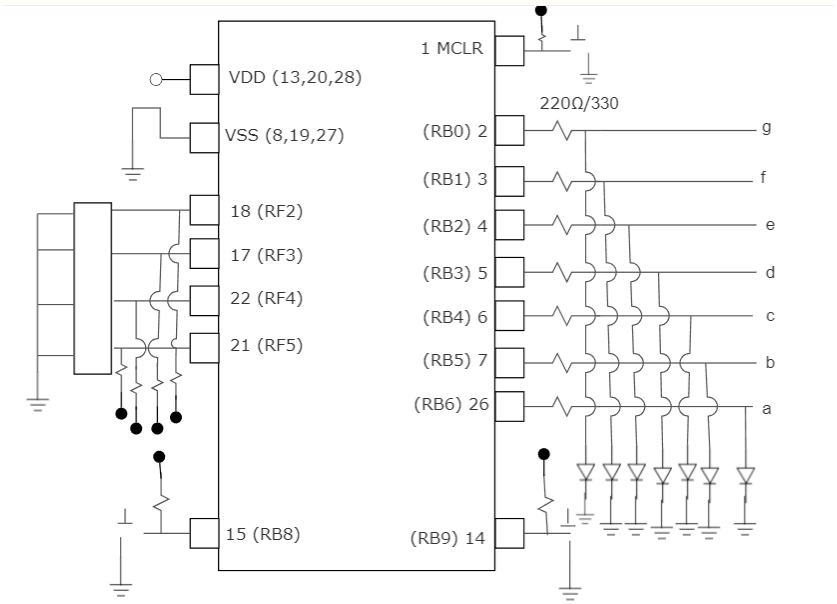
- Sistema Operativo Microsoft Windows 10, 11 o superior.
- MPLAB X IDE v6.05.

3. Desarrollo

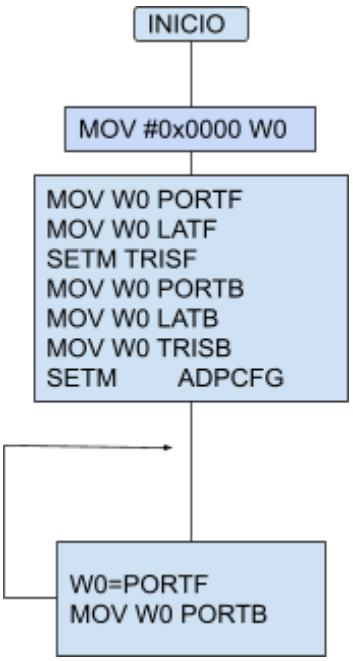
3.1. Primer ejercicio

El primer ejercicio consiste en colocar el valor de la entrada (PORTF) en la salida (PORTB) agregando un corrimiento de dos bits a la derecha para que coincida apropiadamente y se muestre en un arreglo de LEDs.

3.1.1. Diagrama de Bloques (Esquemático)



3.1.2. Diagrama de Flujo (UML)



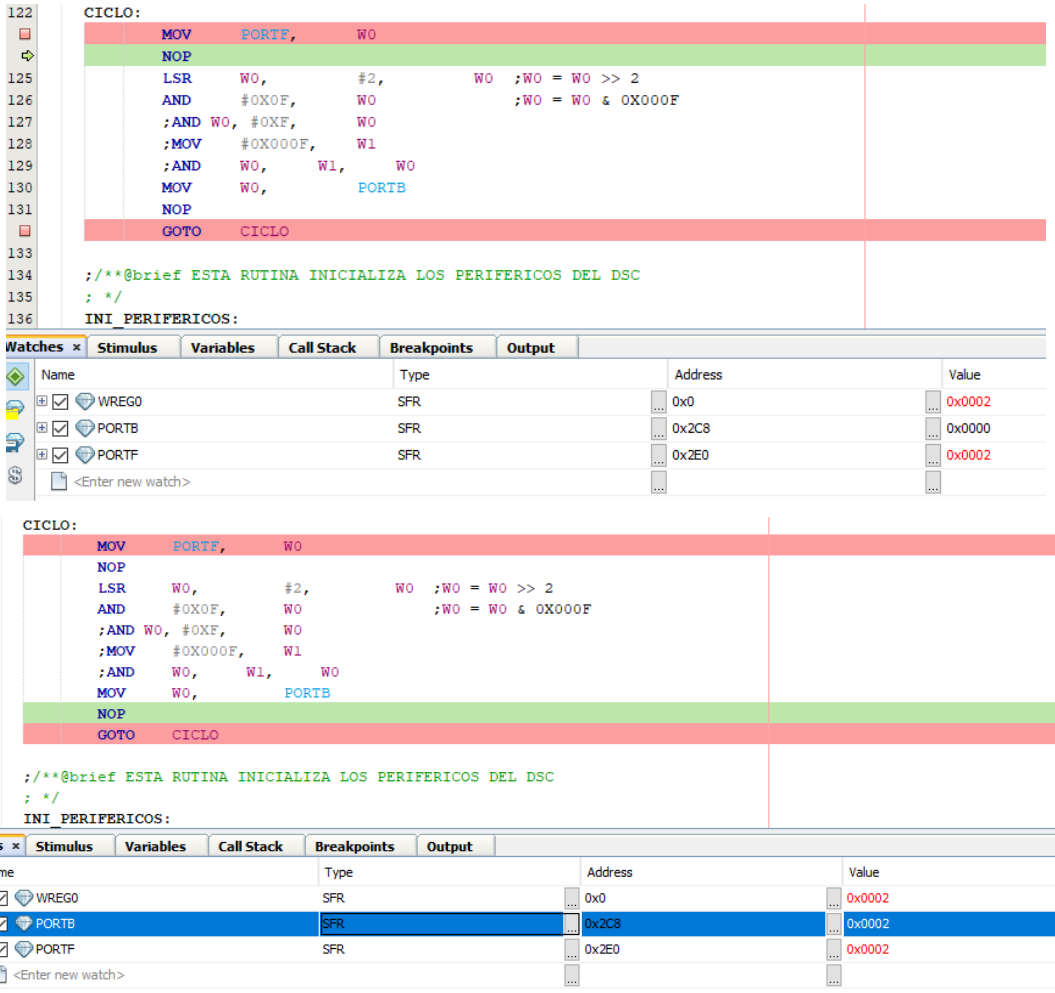
3.1.3. Código

```
;/**@brief ESTE PROGRAMA MUESTRA LOS BLOQUES QUE FORMAN UN PROGRAMA
; * EN ENSAMBLADOR, LOS BLOQUES SON:
; * BLOQUE 1. OPCIONES DE CONFIGURACIÓN DEL DSC: OSCILADOR, WATCHDOG,
; *     BROWN OUT RESET, POWER ON RESET Y CODIGO DE PROTECCION
; * BLOQUE 2. EQUIVALENCIAS Y DECLARACIONES GLOBALES
; * BLOQUE 3. ESPACIOS DE MEMORIA: PROGRAMA, DATOS X, DATOS Y, DATOS NEAR
; * BLOQUE 4. C�DIGO DE APLICACI�N
; * @device: DSPIC30F4013
; * @oscilLator: FRC, 7.3728MHz
; */
       .equ __30F3013, 1
       .include "p30F3013.inc"
;*****************************************************************************
; BITS DE CONFIGURACIÓN
;*****************************************************************************
;.............................................................................
;SE DESACTIVA EL CLOCK SWITCHING Y EL FAIL-SAFE CLOCK MONITOR (FSCM) Y SE
;ACTIVA EL OSCILADOR INTERNO DE 7.3728MHZ(FAST RC) PARA TRABAJAR
;FSCM: PERMITE AL DISPOSITIVO CONTINUAR OPERANDO AUN CUANDO OCURRA UNA FALLA
;EN EL OSCILADOR. CUANDO OCURRE UNA FALLA EN EL OSCILADOR SE GENERA UNA TRAMPA
;Y SE CAMBIA EL RELOJ AL OSCILADOR FRC
;.............................................................................
       config __FOSC, CSW_FSCM_OFF & FRC
;.............................................................................
;SE DESACTIVA EL WATCHDOG
;.............................................................................
       config __FWDT, WDT_OFF
;.............................................................................
;SE ACTIVA EL POWER ON RESET (POR), BROWN OUT RESET (BOR), POWER UP TIMER (PWRT)
;Y EL MASTER CLEAR (MCLR)
;POR: AL MOMENTO DE ALIMENTAR EL DSPIC OCURRE UN RESET CUANDO EL VOLTAJE DE
;ALIMENTACI�N ALCANZA UN VOLTAJE DE UMBRAL (VPOR), EL CUAL ES 1.85V
;BOR: ESTE MODULO GENERA UN RESET CUANDO EL VOLTAJE DE ALIMENTACI�N DECAE
;POR DEBAJO DE UN CIERTO UMBRAL ESTABLECIDO (2.7V)
;PWRT: MANTIENE AL DSPIC EN RESET POR UN CIERTO TIEMPO ESTABLECIDO, ESTO AYUDA
;A ASEGURAR QUE EL VOLTAJE DE ALIMENTACI�N SE HA ESTABILIZADO (16ms)
;.............................................................................
       config __FBORPOR, PBOR_ON & BORV27 & PWRT_16 & MCLR_EN
;.............................................................................
;SE DESACTIVA EL C�DIGO DE PROTECCI�N
;.............................................................................
       config __FGS, CODE_PROT_OFF & GWRP_OFF

;*****************************************************************************
; SECCI�N DE DECLARACI�N DE CONSTANTES CON LA DIRECTIVA .EQU (= DEFINE EN C)
;*****************************************************************************
       .equ MUESTRAS, 64          ;N�MERO DE MUESTRAS

;*****************************************************************************
; DECLARACIONES GLOBALES
;*****************************************************************************
;.............................................................................
;PROPORCIONA ALCANCE GLOBAL A LA FUNCI�N _wreg_init, ESTO PERMITE LLAMAR A LA
;FUNCI�N DESDE UN OTRO PROGRAMA EN ENSAMBLADOR O EN C COLOCANDO LA DECLARACI�N
;"EXTERN"
;.............................................................................
       .global _wreg_init
;.............................................................................
;ETIQUETA DE LA PRIMER LINEA DE C�DIGO
;.............................................................................
       .global __reset
;.............................................................................
;DECLARACI�N DE LA ISR DEL TIMER 1 COMO GLOBAL
;.............................................................................
       .global __T1Interrupt
```

```
;*****************************************************************************
;CONSTANTES ALMACENADAS EN EL ESPACIO DE LA MEMORIA DE PROGRAMA
;*****************************************************************************
       .section .myconstbuffer, code
;.............................................................................
;ALINEA LA SIGUIENTE PALABRA ALMACENADA EN LA MEMORIA
;DE PROGRAMA A UNA DIRECCIÓN MÚLTIPLO DE 2
;.............................................................................
       .palign 2

ps_coeff:
       .hword   0x0002, 0x0003, 0x0005, 0x000A
;short int ps_coeff[] = {0x0002, 0x0003, 0x0005, 0x000A };

;*****************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO X DE LA MEMORIA DE DATOS
;*****************************************************************************
        .section .xbss, bss, xmemory

x_input: .space 2*MUESTRAS        ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;char x_input[128];
;short int x_input[64];
;int x_input[32];
;*****************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO Y DE LA MEMORIA DE DATOS
;*****************************************************************************

         .section .ybss, bss, ymemory

y_input:  .space 2*MUESTRAS       ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;*****************************************************************************
;VARIABLES NO INICIALIZADAS LA MEMORIA DE DATOS CERCANA (NEAR), LOCALIZADA
;EN LOS PRIMEROS 8KB DE RAM
;*****************************************************************************
         .section .nbss, bss, near

var1:     .space 2               ;LA VARIABLE VAR1 RESERVA 1 WORD DE ESPACIO

;*****************************************************************************
;SECCION DE CODIGO EN LA MEMORIA DE PROGRAMA
;*****************************************************************************
.text                                ;INICIO DE LA SECCION DE CODIGO

__reset:
       MOV     #__SP_init,     W15     ;INICIALIZA EL STACK POINTER

       MOV     #__SPLIM_init,  W0      ;INICIALIZA EL REGISTRO STACK POINTER LIMIT
       MOV     W0,             SPLIM

       NOP                             ;UN NOP DESPUES DE LA INICIALIZACION DE SPLIM

       CALL    _WREG_INIT              ;SE LLAMA A LA RUTINA DE INICIALIZACION DE REGISTROS
                                       ;OPCIONALMENTE USAR RCALL EN LUGAR DE CALL
       CALL    INI_PERIFERICOS
CICLO:
       MOV     PORTF,      W0
       NOP
       LSR     W0,         #2,         W0  ;W0 = W0 >> 2
       AND     #0X0F,      W0              ;W0 = W0 & 0X000F
       ;AND W0, #0XF,      W0
       ;MOV    #0X000F,    W1
       ;AND    W0,     W1,     W0
       MOV     W0,         PORTB
       NOP
       GOTO    CICLO

;/**@brief ESTA RUTINA INICIALIZA LOS PERIFERICOS DEL DSC
; */
```

```
INI_PERIFERICOS:
	CLR	PORTF		;PORTF = 0
	NOP
	CLR	LATF		;LATF = 0
	NOP
	SETM	TRISF		;TRISF = 0XFFFF
	NOP

	CLR	PORTB		;PORTB = 0
	NOP
	CLR	LATB		;LATB = 0
	NOP
	CLR	TRISB		;TRISB = 0
	NOP
	SETM	ADPCFG		;ADPCFG = 0XFFFF
				;SE DESHABILITA EL ADC

	RETURN

;/**@brief ESTA RUTINA INICIALIZA LOS REGISTROS Wn A 0X0000
; */
_WREG_INIT:
	CLR	W0
	MOV	W0,			W14
	REPEAT	#12
	MOV	W0,			[++W14]
	CLR	W14
	RETURN

;/**@brief ISR (INTERRUPT SERVICE ROUTINE) DEL TIMER 1
; * SE USA PUSH.S PARA GUARDAR LOS REGISTROS W0, W1, W2, W3,
; * C, Z, N Y DC EN LOS REGISTROS SOMBRA
; */
__T1Interrupt:
	PUSH.S


	BCLR IFS0, #T1IF		;SE LIMPIA LA BANDERA DE INTERRUPCION DEL TIMER 1

	POP.S

	RETFIE				;REGRESO DE LA ISR


.END				;TERMINACION DEL CÓDIGO DE PROGRAMA EN ESTE ARCHIVO
```

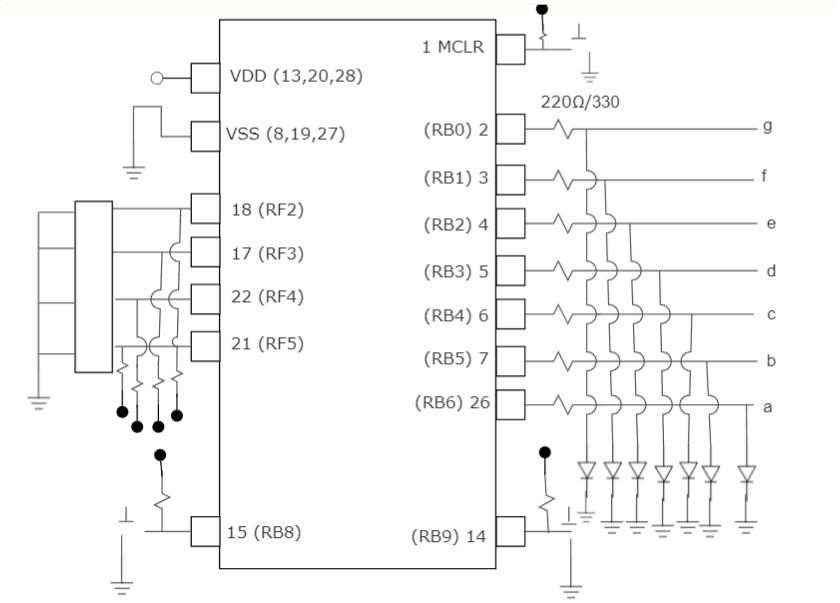
3.1.4. Simulación



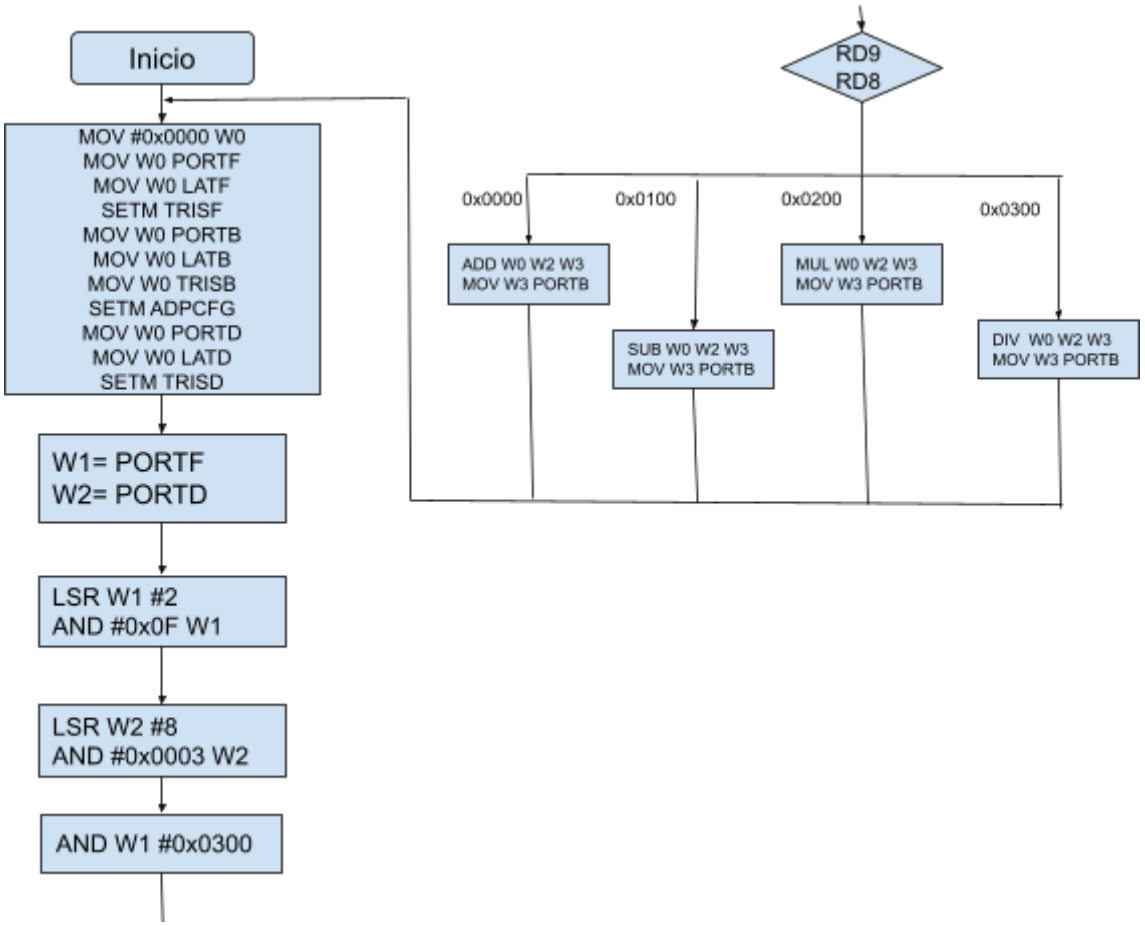
3.2. Segundo ejercicio (versión A)

El segundo ejercicio consiste en programar una calculadora básica, para esta versión (A), usaremos en el código las instrucciones directamente para realizar las operaciones, es decir: `MULT, DIV, SUM` y `DIF`.

3.2.1. Diagrama de Bloques (Esquemático)



3.2.2. Diagrama de Flujo (UML)



3.2.3. Código

```
;/**@brief ESTE PROGRAMA MUESTRA LOS BLOQUES QUE FORMAN UN PROGRAMA
; * EN ENSAMBLADOR, LOS BLOQUES SON:
; * BLOQUE 1. OPCIONES DE CONFIGURACION DEL DSC: OSCILADOR, WATCHDOG,
; *     BROWN OUT RESET, POWER ON RESET Y CODIGO DE PROTECCION
; * BLOQUE 2. EQUIVALENCIAS Y DECLARACIONES GLOBALES
; * BLOQUE 3. ESPACIOS DE MEMORIA: PROGRAMA, DATOS X, DATOS Y, DATOS NEAR
; * BLOQUE 4. C�DIGO DE APLICACI�N
; * @device: DSPIC30F4013
; * @oscilLator: FRC, 7.3728MHz
; */
       .equ __30F3013, 1
       .include "p30F3013.inc"
;*****************************************************************************
; BITS DE CONFIGURACI�N
;*****************************************************************************
;.............................................................................
;SE DESACTIVA EL CLOCK SWITCHING Y EL FAIL-SAFE CLOCK MONITOR (FSCM) Y SE
;ACTIVA EL OSCILADOR INTERNO DE 7.3728MHZ(FAST RC) PARA TRABAJAR
;FSCM: PERMITE AL DISPOSITIVO CONTINUAR OPERANDO AUN CUANDO OCURRA UNA FALLA
;EN EL OSCILADOR. CUANDO OCURRE UNA FALLA EN EL OSCILADOR SE GENERA UNA TRAMPA
;Y SE CAMBIA EL RELOJ AL OSCILADOR FRC
;.............................................................................
       config __FOSC, CSW_FSCM_OFF & FRC
;.............................................................................
;SE DESACTIVA EL WATCHDOG
;.............................................................................
       config __FWDT, WDT_OFF
;.............................................................................
;SE ACTIVA EL POWER ON RESET (POR), BROWN OUT RESET (BOR), POWER UP TIMER (PWRT)
;Y EL MASTER CLEAR (MCLR)
;POR: AL MOMENTO DE ALIMENTAR EL DSPIC OCURRE UN RESET CUANDO EL VOLTAJE DE
;ALIMENTACI�N ALCANZA UN VOLTAJE DE UMBRAL (VPOR), EL CUAL ES 1.85V
;BOR: ESTE MODULO GENERA UN RESET CUANDO EL VOLTAJE DE ALIMENTACI�N DECAE
;POR DEBAJO DE UN CIERTO UMBRAL ESTABLECIDO (2.7V)
;PWRT: MANTIENE AL DSPIC EN RESET POR UN CIERTO TIEMPO ESTABLECIDO, ESTO AYUDA
;A ASEGURAR QUE EL VOLTAJE DE ALIMENTACI�N SE HA ESTABILIZADO (16ms)
;.............................................................................
       config __FBORPOR, PBOR_ON & BORV27 & PWRT_16 & MCLR_EN
;.............................................................................
;SE DESACTIVA EL C�DIGO DE PROTECCI�N
;.............................................................................
       config __FGS, CODE_PROT_OFF & GWRP_OFF

;*****************************************************************************
; SECCI�N DE DECLARACI�N DE CONSTANTES CON LA DIRECTIVA .EQU (= DEFINE EN C)
;*****************************************************************************
       .equ MUESTRAS, 64          ;N�MERO DE MUESTRAS

;*****************************************************************************
; DECLARACIONES GLOBALES
;*****************************************************************************
;.............................................................................
;PROPORCIONA ALCANCE GLOBAL A LA FUNCI�N _wreg_init, ESTO PERMITE LLAMAR A LA
;FUNCI�N DESDE UN OTRO PROGRAMA EN ENSAMBLADOR O EN C COLOCANDO LA DECLARACI�N
;"EXTERN"
;.............................................................................
       .global _wreg_init
;.............................................................................
;ETIQUETA DE LA PRIMER LINEA DE C�DIGO
;.............................................................................
       .global __reset
;.............................................................................
;DECLARACI�N DE LA ISR DEL TIMER 1 COMO GLOBAL
;.............................................................................
       .global __T1Interrupt
```

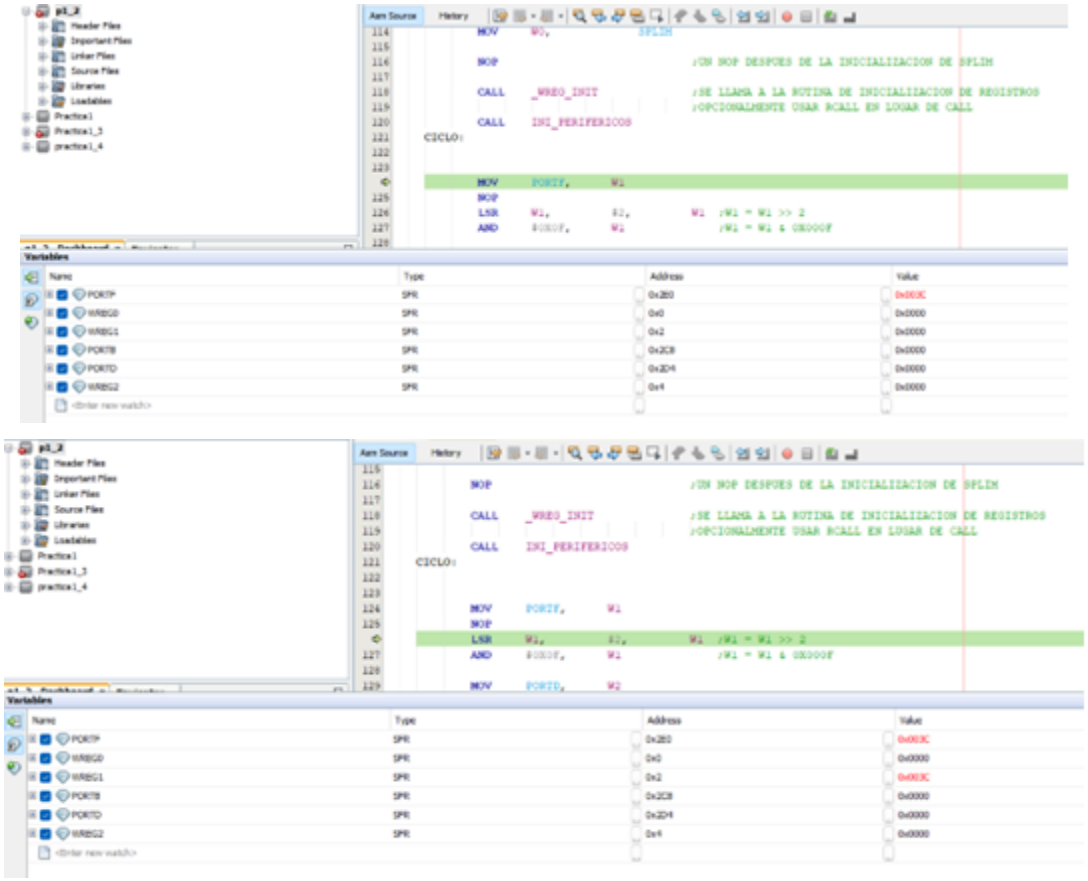
```
;*****************************************************************************
;CONSTANTES ALMACENADAS EN EL ESPACIO DE LA MEMORIA DE PROGRAMA
;*****************************************************************************
       .section .myconstbuffer, code
;.............................................................................
;ALINEA LA SIGUIENTE PALABRA ALMACENADA EN LA MEMORIA
;DE PROGRAMA A UNA DIRECCION MULTIPLO DE 2
;.............................................................................
       .palign 2

ps_coeff:
       .hword   0x0002, 0x0003, 0x0005, 0x000A
;short int ps_coeff[] = {0x0002, 0x0003, 0x0005, 0x000A };

;*****************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO X DE LA MEMORIA DE DATOS
;*****************************************************************************
        .section .xbss, bss, xmemory

x_input: .space 2*MUESTRAS        ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;char x_input[128];
;short int x_input[64];
;int x_input[32];
;*****************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO Y DE LA MEMORIA DE DATOS
;*****************************************************************************

        .section .ybss, bss, ymemory

y_input:  .space 2*MUESTRAS       ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;*****************************************************************************
;VARIABLES NO INICIALIZADAS LA MEMORIA DE DATOS CERCANA (NEAR), LOCALIZADA
;EN LOS PRIMEROS 8KB DE RAM
;*****************************************************************************
        .section .nbss, bss, near

var1:     .space 2               ;LA VARIABLE VAR1 RESERVA 1 WORD DE ESPACIO

;*****************************************************************************
;SECCION DE CODIGO EN LA MEMORIA DE PROGRAMA
;*****************************************************************************
.text                                ;INICIO DE LA SECCION DE CODIGO

__reset:
       MOV     #__SP_init,     W15     ;INICIALIZA EL STACK POINTER

       MOV     #__SPLIM_init,  W0      ;INICIALIZA EL REGISTRO STACK POINTER LIMIT
       MOV     W0,             SPLIM

       NOP                             ;UN NOP DESPUES DE LA INICIALIZACION DE SPLIM

       CALL    _WREG_INIT              ;SE LLAMA A LA RUTINA DE INICIALIZACION DE REGISTROS
                                       ;OPCIONALMENTE USAR RCALL EN LUGAR DE CALL
       CALL    INI_PERIFERICOS
CICLO:

       MOV     PORTF,      W1
       NOP
       LSR     W1,         #2,         W1   ;W1 = W1 >> 2
       AND     #0X0F,      W1               ;W1 = W1 & 0X000F

       MOV     PORTD,      W2
       NOP
       LSR     W2,         #8,         W2   ;W2 = W2 >> 8
       AND     #0X0003,    W2               ;W0 = W0 & 0X0003

       MOV     #3,    W3
```

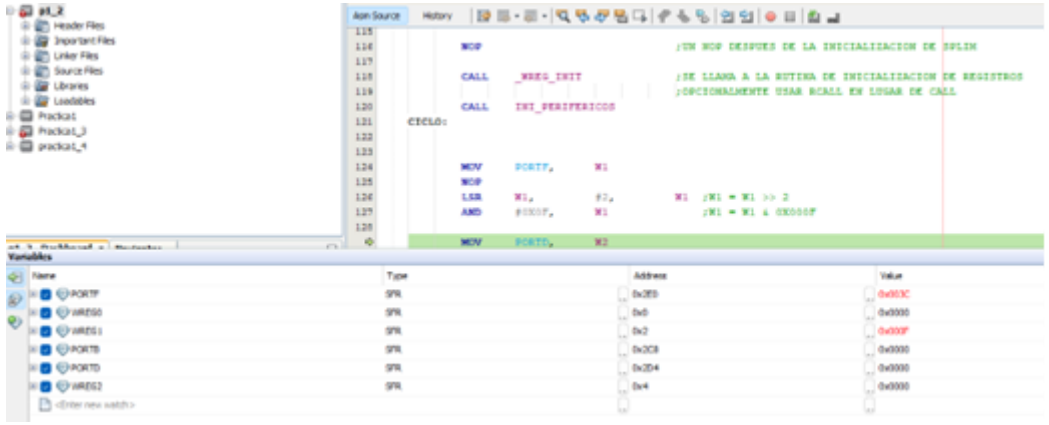
```
       BTSS    W2,#0 ;SI ES 1 SE SALTA
       GOTO CEROS
       GOTO UNOS


       GOTO    CICLO
CEROS:
   BTSS        W2,#1 ;SI ES 1 SE SALTA
   GOTO SUMA
   GOTO MULT
UNOS:
   BTSS        W2,#1 ;SI ES 1 SE SALTA
   GOTO RESTA
   GOTO DIVI
SUMA:
   ADD     W1,     W3,     W4
   MOV     W4,     PORTB
   NOP
   GOTO CICLO
RESTA:
   SUB     W1,     W3,     W4
   MOV     W4,     PORTB
   NOP
   GOTO CICLO
MULT:
   MUL.UU  W1,     W3,     W4
   MOV     W4,     PORTB
   NOP
   GOTO CICLO
DIVI:
   REPEAT #17
   DIV.U   W1,W3
   MOV     W0,     PORTB
   NOP
   GOTO CICLO

;/**@brief ESTA RUTINA INICIALIZA LOS PERIFERICOS DEL DSC
; */
INI_PERIFERICOS:
       MOV     #0X0000,    W0
       NOP
       MOV     W0,         PORTF;PORTF = 0X0000
       NOP
       MOV     W0,         LATF;LATF = 0X0000
       NOP
       SETM    TRISF           ;TRISF = 0XFFFF
       NOP
       MOV     W0,         PORTB;PORTB = 0X0000
       NOP
       MOV     W0,         LATB;LATB = 0X0000
       NOP
       MOV     W0,         TRISB;TRISB = 0X0000
       NOP
       SETM    ADPCFG
       NOP
       MOV     W0,         PORTD;PORTD = 0X0000
       NOP
       MOV     W0,         LATD;LATD = 0X0000
       NOP
       SETM    TRISD           ;TRISD = 0XFFFF

       RETURN

;/**@brief ESTA RUTINA INICIALIZA LOS REGISTROS Wn A 0X0000
; */
_WREG_INIT:
       CLR     W0
       MOV     W0,                         W14
```

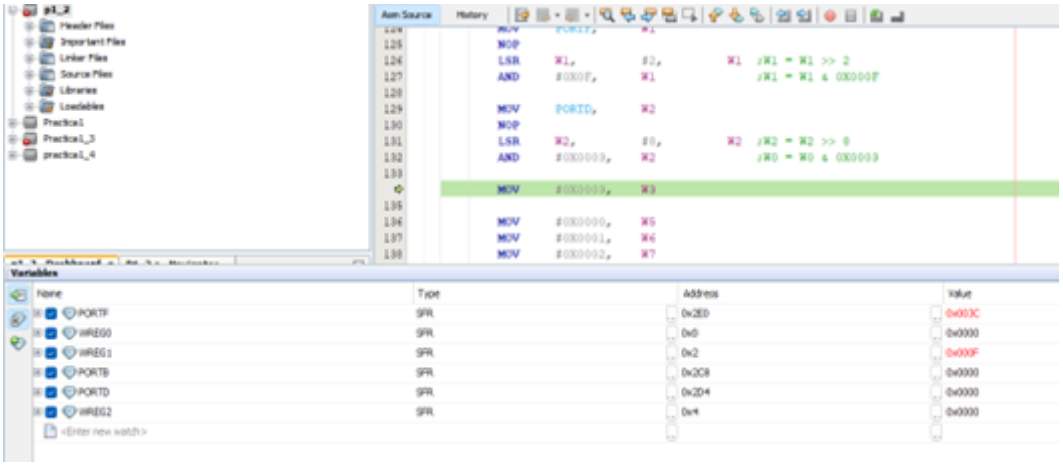
```
        REPEAT  #12
        MOV     W0,                             [++W14]
        CLR     W14
        RETURN

;/**@brief ISR (INTERRUPT SERVICE ROUTINE) DEL TIMER 1
; * SE USA PUSH.S PARA GUARDAR LOS REGISTROS W0, W1, W2, W3,
; * C, Z, N Y DC EN LOS REGISTROS SOMBRA
; */
__T1Interrupt:
        PUSH.S


        BCLR IFS0, #T1IF            ;SE LIMPIA LA BANDERA DE INTERRUPCION DEL TIMER 1

        POP.S

        RETFIE                      ;REGRESO DE LA ISR


.END                                 ;TERMINACION DEL CODIGO DE PROGRAMA EN ESTE ARCHIVO
```

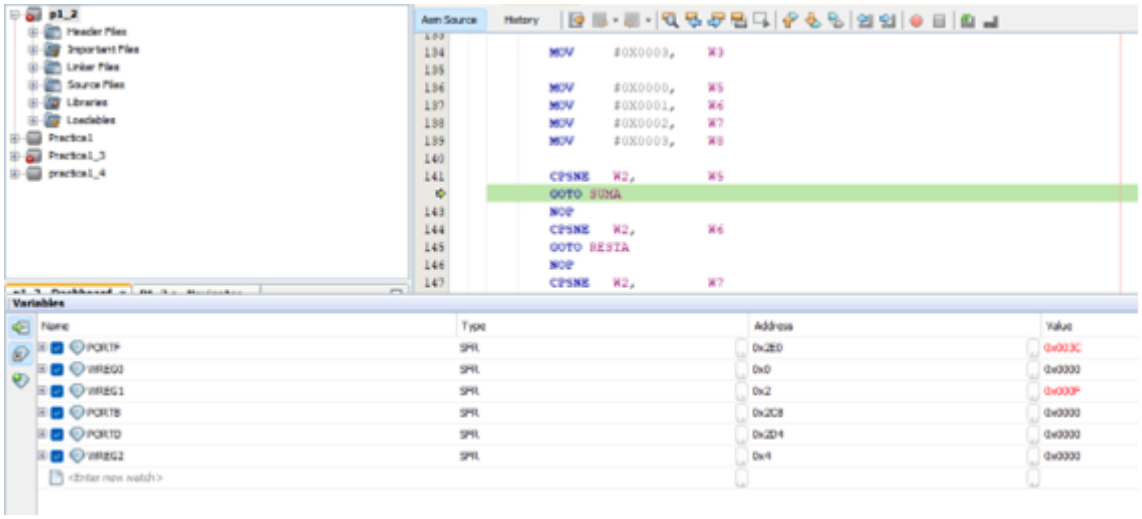
3.2.4. Simulación

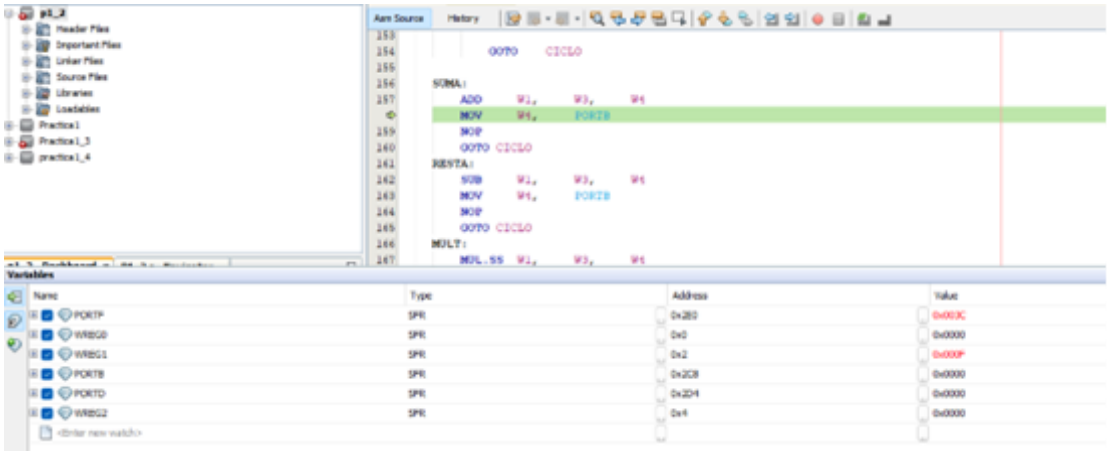
3.2.4.1. Suma



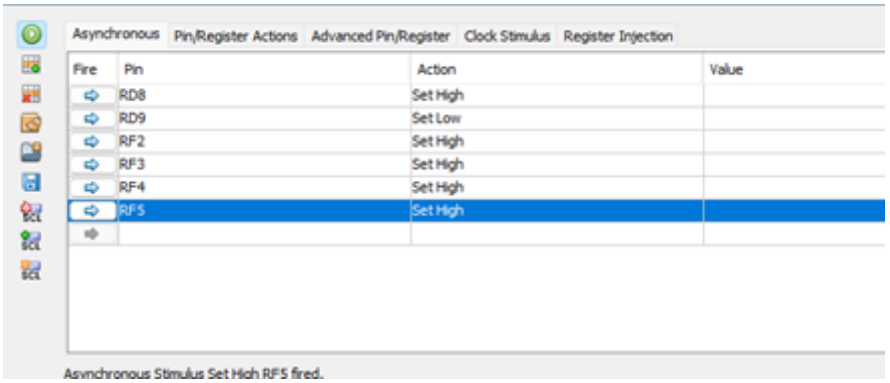


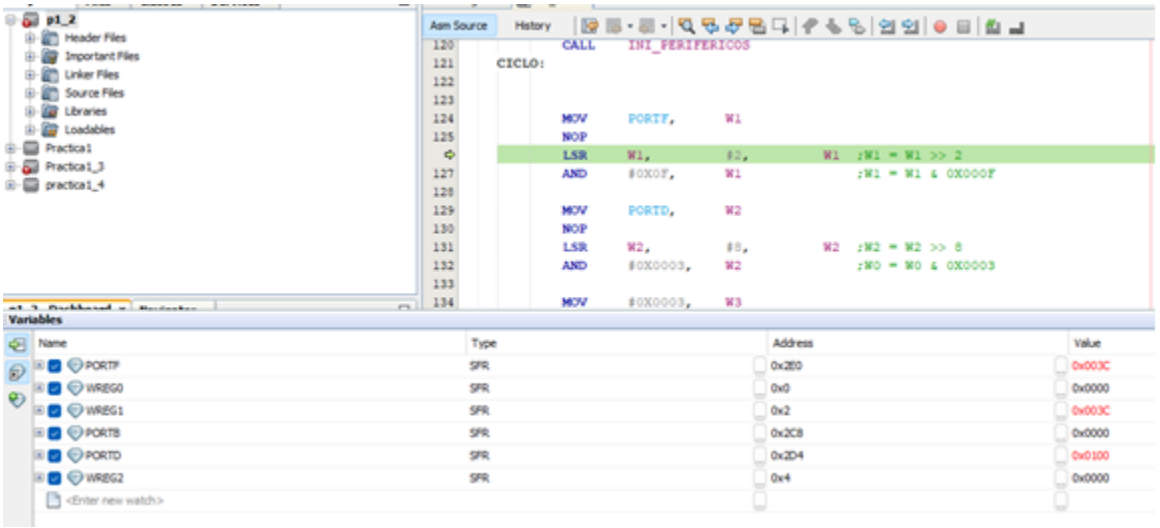


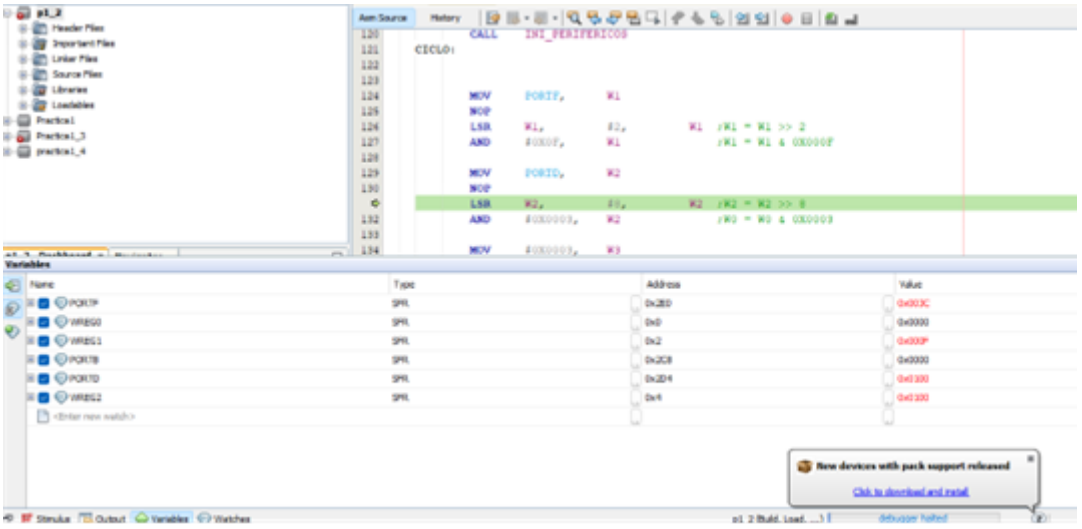


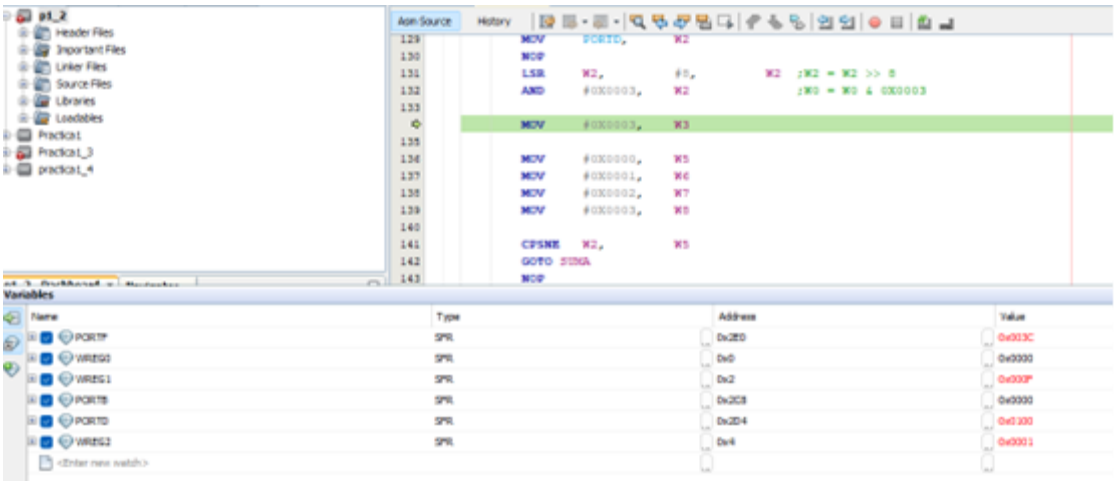


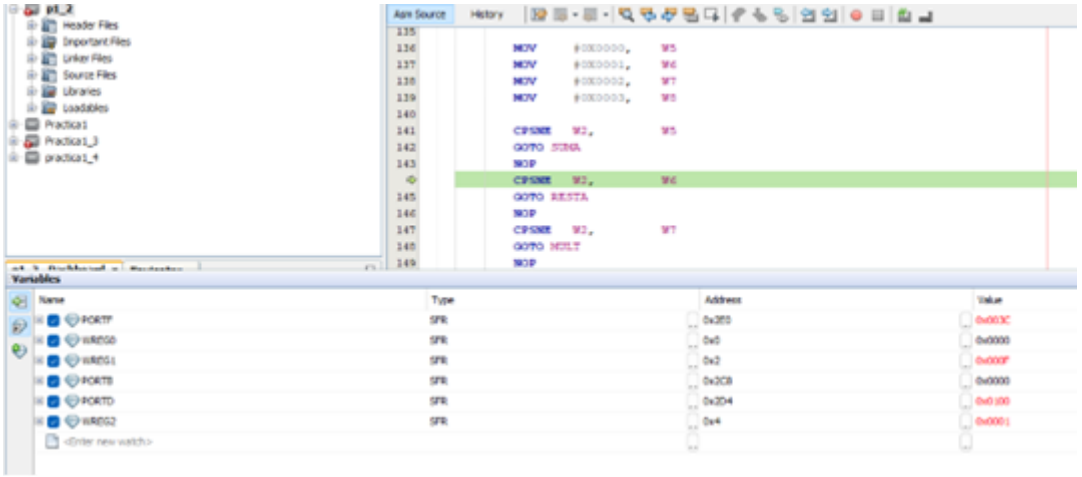
3.2.4.2. Resta

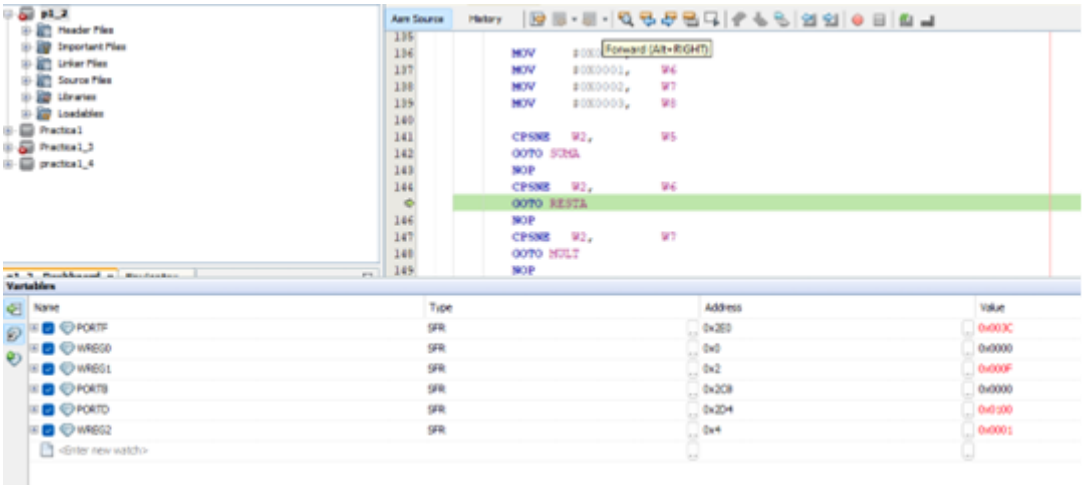

Asynchronous
Pin/Register Actions
Advanced Pin/Register
Clock Stimulus
Register Injection
Fire
Pin
Action
Value
RD8
Set High
RD9
Set Low
RF2
Set High
RF3
Set High
RF4
Set High
RF5
Set High
Asynchronous Stimulus Set High RF5 fired.

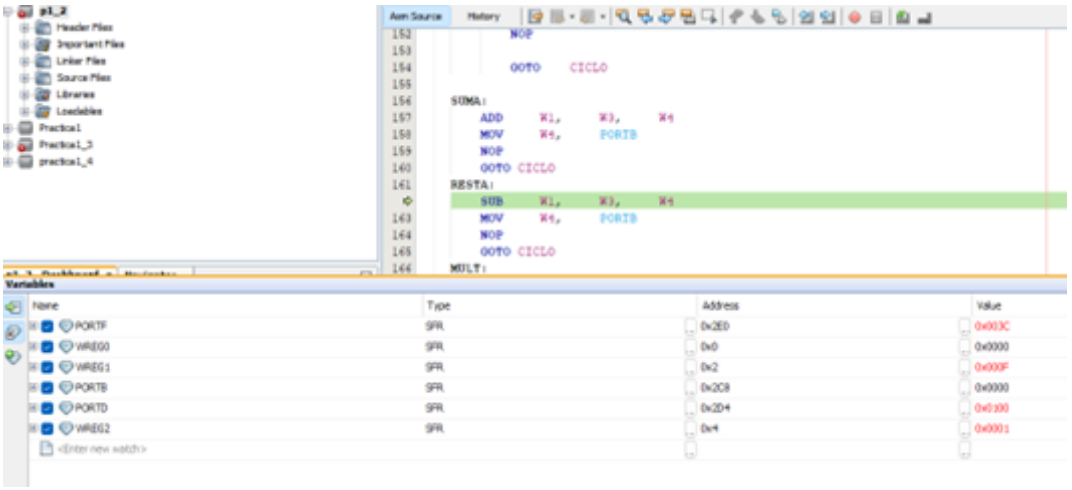




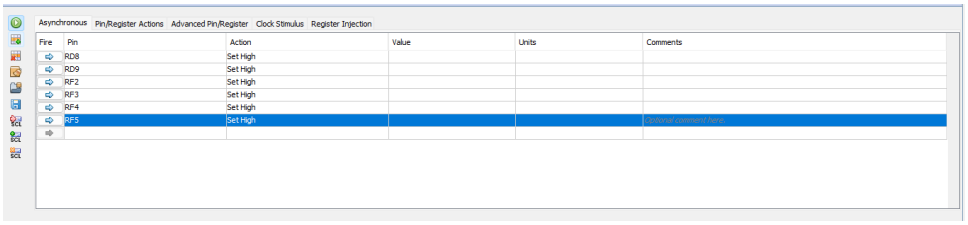


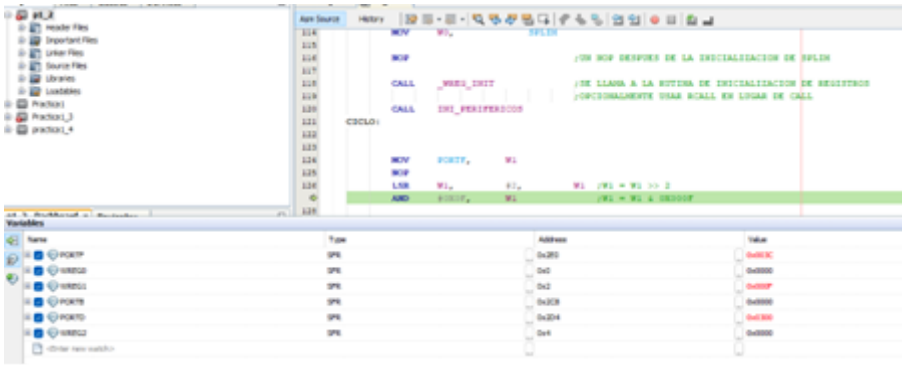


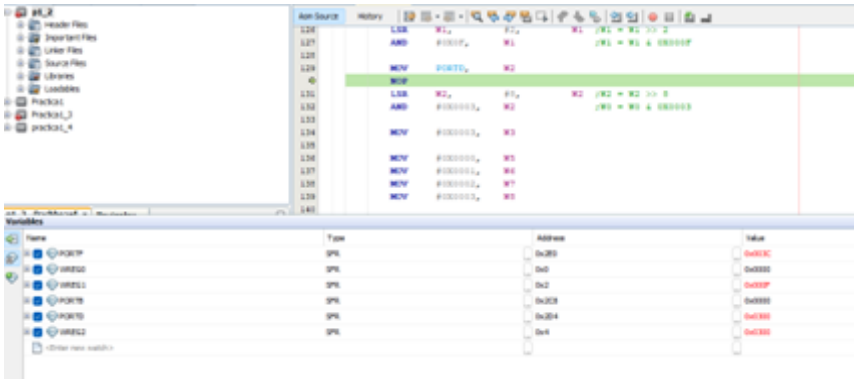


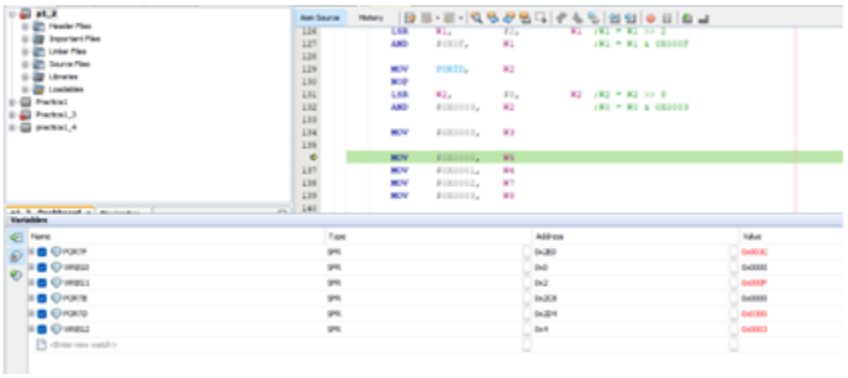


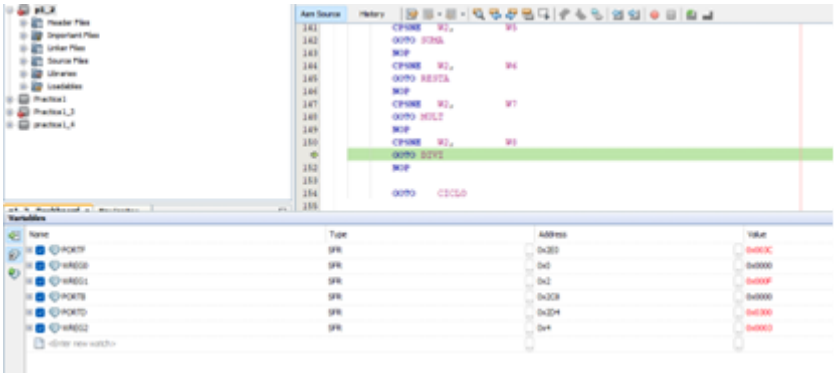
3.2.4.3. División

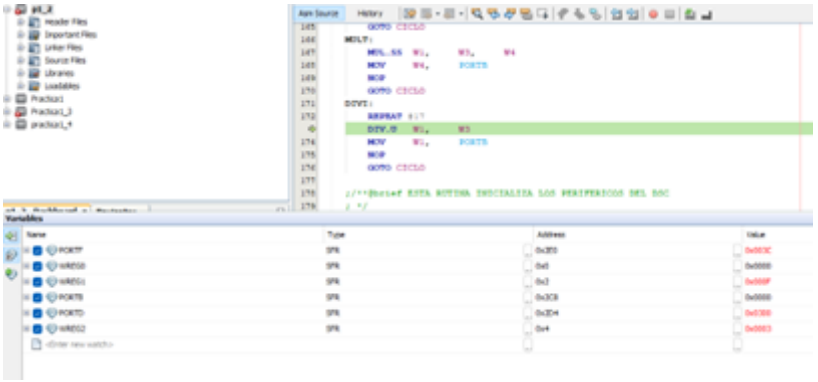




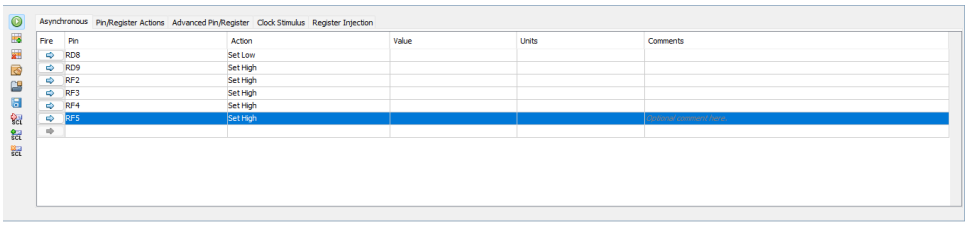


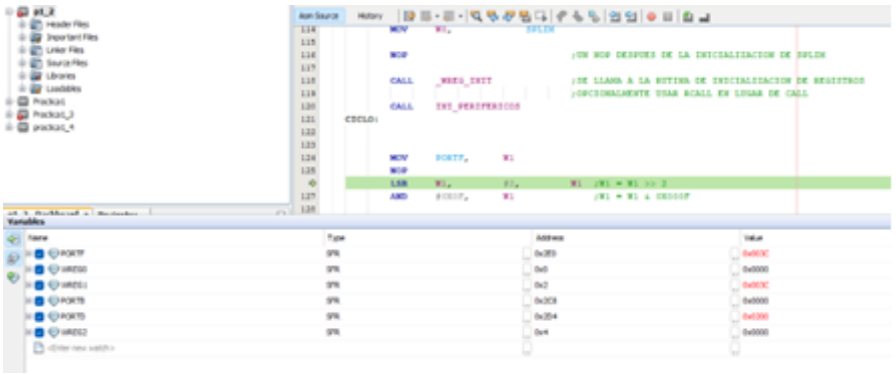


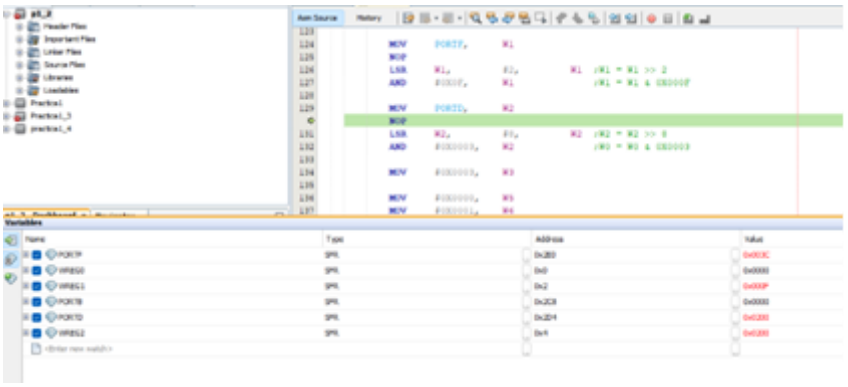


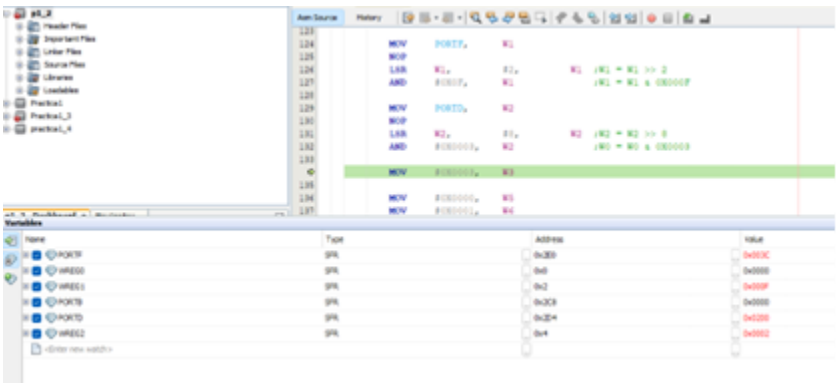


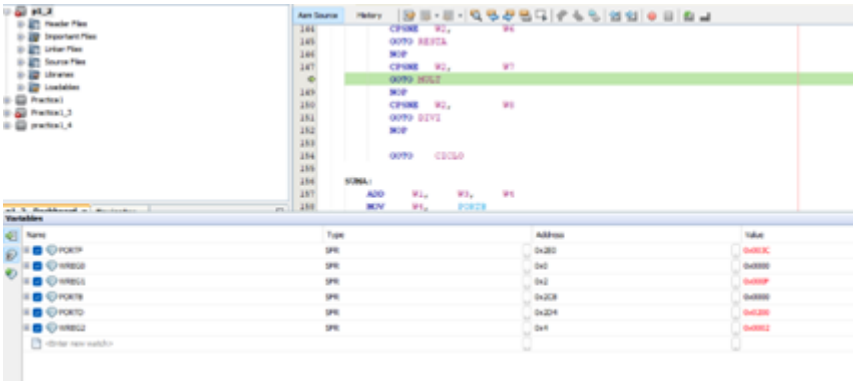
3.2.4.4. Multiplicación

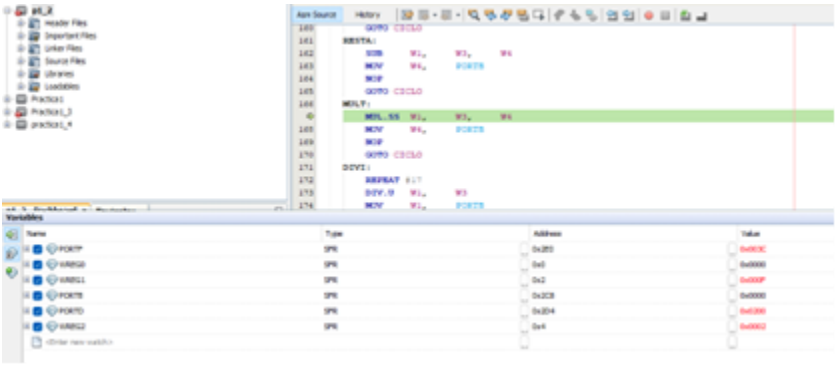








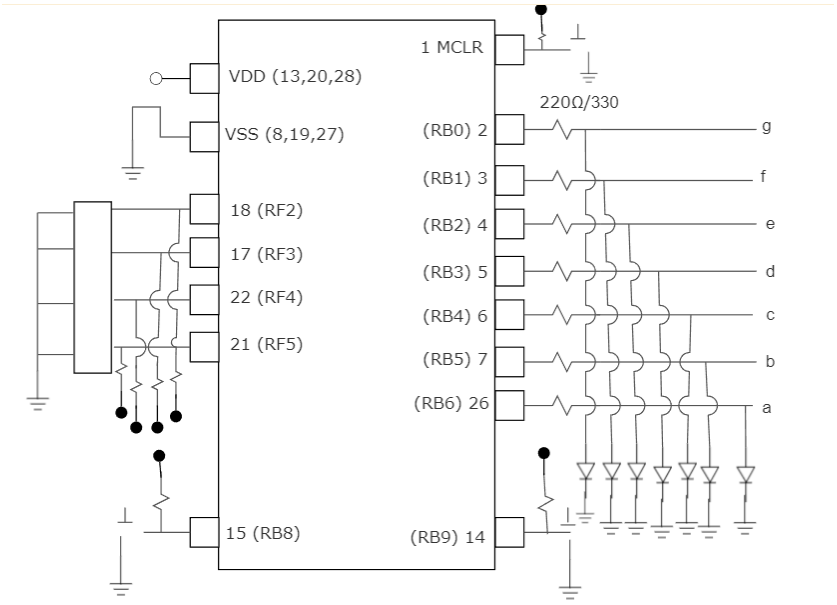




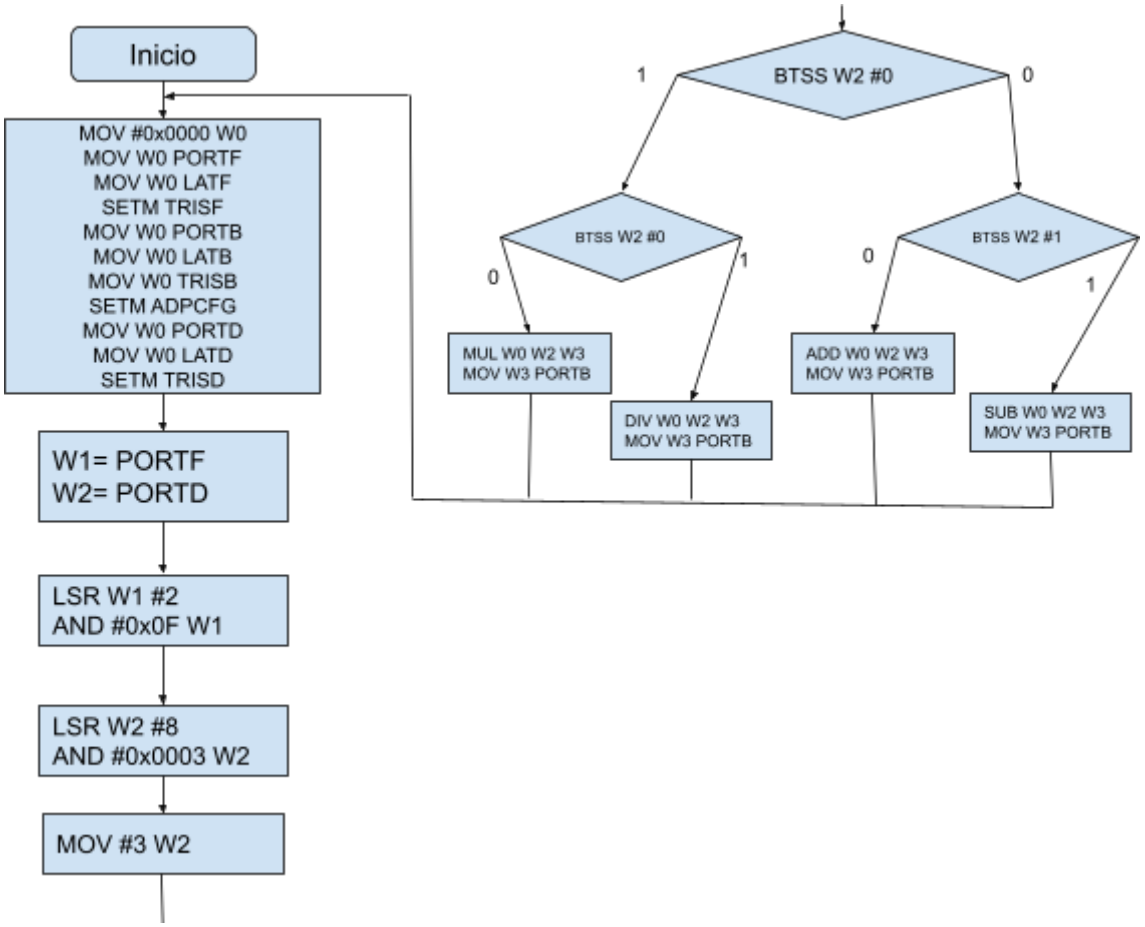
3.3. Segundo ejercicio (versión B)

Siguiendo con la versión A, este segundo ejercicio realiza una calculadora básica *pero*, a diferencia de la primer versión, aquí utilizamos corrimientos de bits para realizar las operaciones directamente.

3.3.1. Diagrama de Bloques (Esquemático)



3.3.2. Diagrama de Flujo (UML)



3.3.3. Código

```
;/**@brief ESTE PROGRAMA MUESTRA LOS BLOQUES QUE FORMAN UN PROGRAMA
; * EN ENSAMBLADOR, LOS BLOQUES SON:
; * BLOQUE 1. OPCIONES DE CONFIGURACION DEL DSC: OSCILADOR, WATCHDOG,
; *     BROWN OUT RESET, POWER ON RESET Y CODIGO DE PROTECCION
; * BLOQUE 2. EQUIVALENCIAS Y DECLARACIONES GLOBALES
; * BLOQUE 3. ESPACIOS DE MEMORIA: PROGRAMA, DATOS X, DATOS Y, DATOS NEAR
; * BLOQUE 4. C�DIGO DE APLICACI�N
; * @device: DSPIC30F4013
; * @oscillator: FRC, 7.3728MHz
; */
      .equ __30F3013, 1
      .include "p30F3013.inc"
;*****************************************************************************
; BITS DE CONFIGURACI�N
;*****************************************************************************
;.............................................................................
;SE DESACTIVA EL CLOCK SWITCHING Y EL FAIL-SAFE CLOCK MONITOR (FSCM) Y SE
;ACTIVA EL OSCILADOR INTERNO DE 7.3728MHZ(FAST RC) PARA TRABAJAR
;FSCM: PERMITE AL DISPOSITIVO CONTINUAR OPERANDO AUN CUANDO OCURRA UNA FALLA
;EN EL OSCILADOR. CUANDO OCURRE UNA FALLA EN EL OSCILADOR SE GENERA UNA TRAMPA
;Y SE CAMBIA EL RELOJ AL OSCILADOR FRC
;.............................................................................
      config __FOSC, CSW_FSCM_OFF & FRC
;.............................................................................
;SE DESACTIVA EL WATCHDOG
;.............................................................................
      config __FWDT, WDT_OFF
;.............................................................................
```

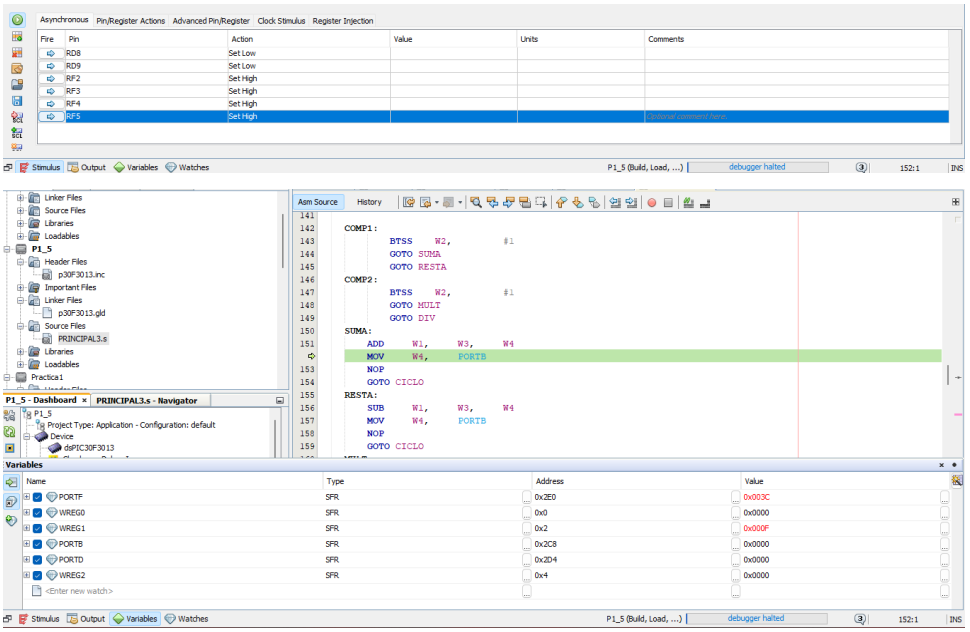
```
;SE ACTIVA EL POWER ON RESET (POR), BROWN OUT RESET (BOR), POWER UP TIMER (PWRT)
;Y EL MASTER CLEAR (MCLR)
;POR: AL MOMENTO DE ALIMENTAR EL DSPIC OCURRE UN RESET CUANDO EL VOLTAJE DE
;ALIMENTACI?N ALCANZA UN VOLTAJE DE UMBRAL (VPOR), EL CUAL ES 1.85V
;BOR: ESTE MODULO GENERA UN RESET CUANDO EL VOLTAJE DE ALIMENTACI?N DECAE
;POR DEBAJO DE UN CIERTO UMBRAL ESTABLECIDO (2.7V)
;PWRT: MANTIENE AL DSPIC EN RESET POR UN CIERTO TIEMPO ESTABLECIDO, ESTO AYUDA
;A ASEGURAR QUE EL VOLTAJE DE ALIMENTACI?N SE HA ESTABILIZADO (16ms)
;.....
       config __FBORPOR, PBOR_ON & BORV27 & PWRT_16 & MCLR_EN
;.....
;SE DESACTIVA EL C?DIGO DE PROTECCI?N
;.....
       config __FGS, CODE_PROT_OFF & GWRP_OFF

;*****************************************************************************
; SECCI?N DE DECLARACI?N DE CONSTANTES CON LA DIRECTIVA .EQU (= DEFINE EN C)
;*****************************************************************************
       .equ MUESTRAS, 64          ;N?MERO DE MUESTRAS

;*****************************************************************************
; DECLARACIONES GLOBALES
;*****************************************************************************
;.....
;PROPORCIONA ALCANCE GLOBAL A LA FUNCI?N _wreg_init, ESTO PERMITE LLAMAR A LA
;FUNCI?N DESDE UN OTRO PROGRAMA EN ENSAMBLADOR O EN C COLOCANDO LA DECLARACI?N
;"EXTERN"
;.....
       .global _wreg_init
;.....
;ETIQUETA DE LA PRIMER LINEA DE C?DIGO
;.....
       .global __reset
;.....
;DECLARACI?N DE LA ISR DEL TIMER 1 COMO GLOBAL
;.....
       .global __T1Interrupt

;*****************************************************************************
;CONSTANTES ALMACENADAS EN EL ESPACIO DE LA MEMORIA DE PROGRAMA
;*****************************************************************************
       .section .myconstbuffer, code
;.....
;ALINEA LA SIGUIENTE PALABRA ALMACENADA EN LA MEMORIA
;DE PROGRAMA A UNA DIRECCION MULTIPLO DE 2
;.....
       .palign 2

ps_coeff:
       .hword   0x0002, 0x0003, 0x0005, 0x000A
;short int ps_coeff[] = {0x0002, 0x0003, 0x0005, 0x000A };

;*****************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO X DE LA MEMORIA DE DATOS
;*****************************************************************************
        .section .xbss, bss, xmemory

x_input: .space 2*MUESTRAS        ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;char x_input[128];
;short int x_input[64];
;int x_input[32];
;*****************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO Y DE LA MEMORIA DE DATOS
;*****************************************************************************

         .section .ybss, bss, ymemory

y_input:  .space 2*MUESTRAS       ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
```

```
;*****************************************************************************
;VARIABLES NO INICIALIZADAS LA MEMORIA DE DATOS CERCANA (NEAR), LOCALIZADA
;EN LOS PRIMEROS 8KB DE RAM
;*****************************************************************************
        .section .nbss, bss, near

var1:     .space 2              ;LA VARIABLE VAR1 RESERVA 1 WORD DE ESPACIO

;*****************************************************************************
;SECCION DE CODIGO EN LA MEMORIA DE PROGRAMA
;*****************************************************************************
.text                                 ;INICIO DE LA SECCION DE CODIGO

__reset:
       MOV     #__SP_init,     W15     ;INICIALIZA EL STACK POINTER

       MOV     #__SPLIM_init,  W0      ;INICIALIZA EL REGISTRO STACK POINTER LIMIT
       MOV     W0,             SPLIM

       NOP                             ;UN NOP DESPUES DE LA INICIALIZACION DE SPLIM

       CALL    _WREG_INIT              ;SE LLAMA A LA RUTINA DE INICIALIZACION DE REGISTROS
                                       ;OPCIONALMENTE USAR RCALL EN LUGAR DE CALL
       CALL    INI_PERIFERICOS
CICLO:

       MOV     PORTF,      W1
       NOP
       LSR     W1,         #2,         W1  ;W1 = W1 >> 2
       AND     #0X0F,      W1              ;W1 = W1 & 0X000F

       MOV     PORTD,      W2
       NOP
       LSR     W2,         #8,         W2  ;W2 = W2 >> 8
       AND     #0X0003,    W2              ;W0 = W0 & 0X0003

       MOV     #3,    W3

       BTSS    W2,#0 ;SI ES 1 SE SALTA
       GOTO CEROS
       GOTO UNOS


       GOTO    CICLO
CEROS:
   BTSS        W2,#1 ;SI ES 1 SE SALTA
   GOTO SUMA
   GOTO MULT
UNOS:
   BTSS        W2,#1 ;SI ES 1 SE SALTA
   GOTO RESTA
   GOTO DIVI
SUMA:
   ADD     W1,     W3,     W4
   MOV     W4,     PORTB
   NOP
   GOTO CICLO
RESTA:
   SUB     W1,     W3,     W4
   MOV     W4,     PORTB
   NOP
   GOTO CICLO
MULT:
   MUL.UU  W1,     W3,     W4
   MOV     W4,     PORTB
   NOP
   GOTO CICLO
DIVI:
```

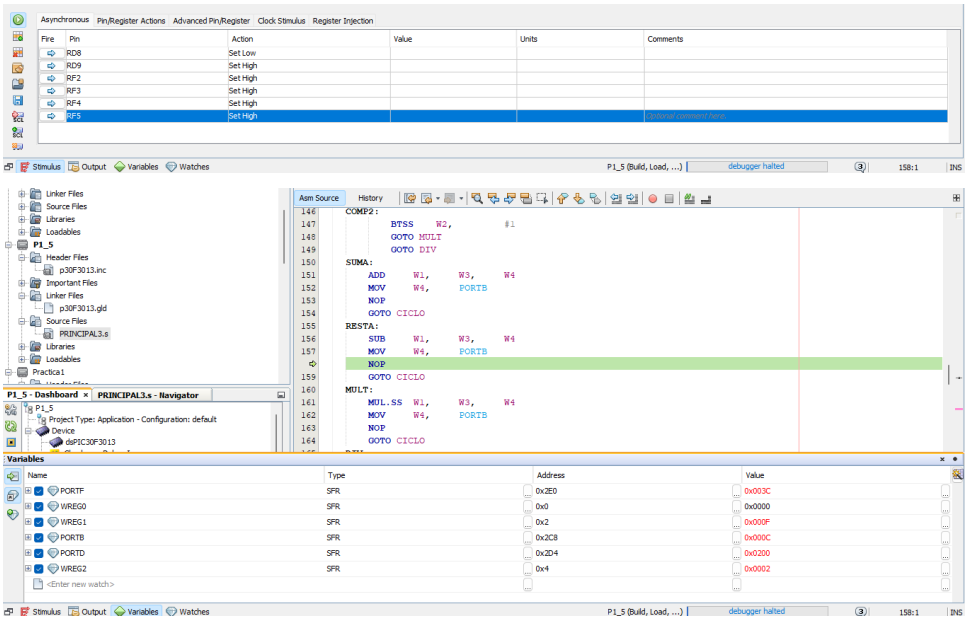
```
   REPEAT #17
   DIV.U   W1,W3
   MOV     W0,     PORTB
   NOP
   GOTO CICLO

;/**@brief ESTA RUTINA INICIALIZA LOS PERIFERICOS DEL DSC
; */
INI_PERIFERICOS:
      MOV     #0X0000,    W0
      NOP
      MOV     W0,         PORTF;PORTF = 0X0000
      NOP
      MOV     W0,         LATF;LATF = 0X0000
      NOP
      SETM    TRISF           ;TRISF = 0XFFFF
      NOP
      MOV     W0,         PORTB;PORTB = 0X0000
      NOP
      MOV     W0,         LATB;LATB = 0X0000
      NOP
      MOV     W0,         TRISB;TRISB = 0X0000
      NOP
      SETM    ADPCFG
      NOP
      MOV     W0,         PORTD;PORTD = 0X0000
      NOP
      MOV     W0,         LATD;LATD = 0X0000
      NOP
      SETM    TRISD           ;TRISD = 0XFFFF

      RETURN

;/**@brief ESTA RUTINA INICIALIZA LOS REGISTROS Wn A 0X0000
; */
_WREG_INIT:
      CLR     W0
      MOV     W0,                         W14
      REPEAT  #12
      MOV     W0,                         [++W14]
      CLR     W14
      RETURN

;/**@brief ISR (INTERRUPT SERVICE ROUTINE) DEL TIMER 1
; * SE USA PUSH.S PARA GUARDAR LOS REGISTROS W0, W1, W2, W3,
; * C, Z, N Y DC EN LOS REGISTROS SOMBRA
; */
__T1Interrupt:
      PUSH.S


      BCLR IFS0, #T1IF          ;SE LIMPIA LA BANDERA DE INTERRUPCION DEL TIMER 1

      POP.S

      RETFIE                    ;REGRESO DE LA ISR


.END                             ;TERMINACION DEL CODIGO DE PROGRAMA EN ESTE ARCHIVO
```

3.3.4. Simulación

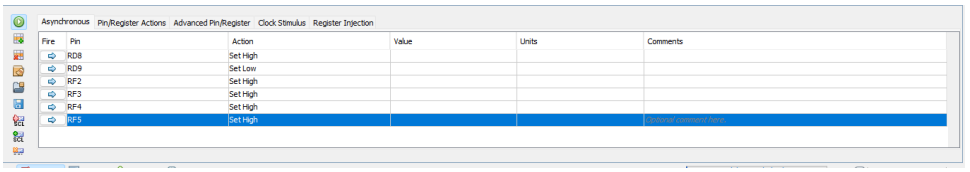
3.3.4.1. Suma

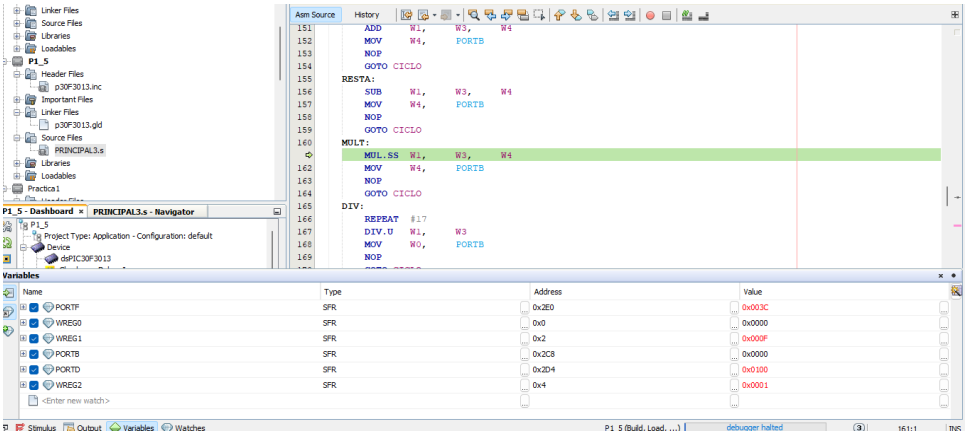


3.3.4.2. Resta

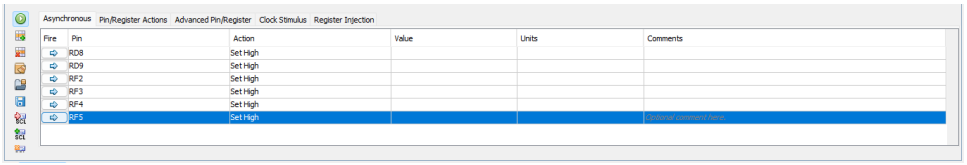


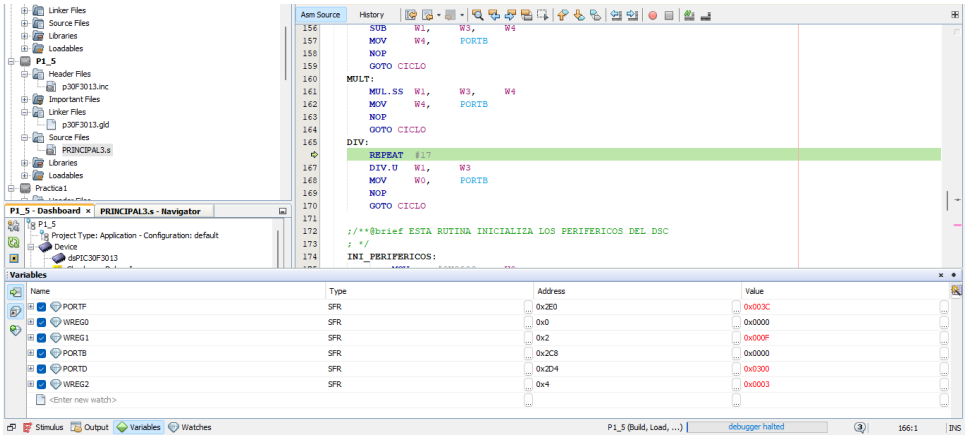
3.3.4.3. Multiplicación





3.3.4.4. División

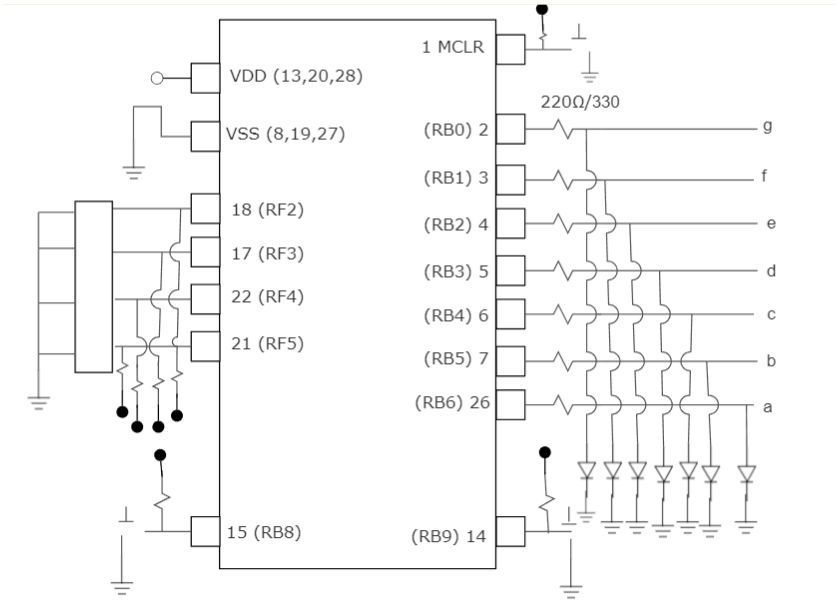




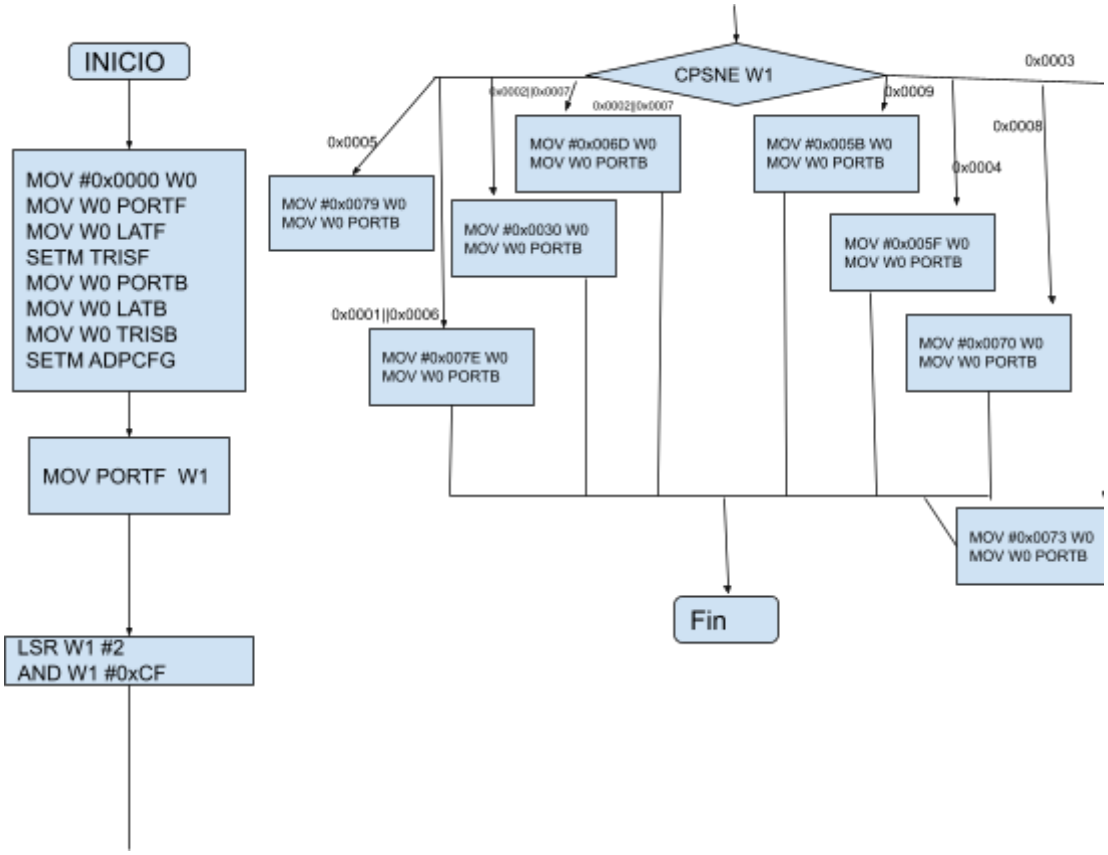
3.4. Tercer ejercicio

El tercer ejercicio consiste en mostrar un número de boleta en un display de siete segmentos de cátodo común, para esto el usuario colocará el ID del número a mostrar (10 dígitos), en un *dip-switch* y se deberá mostrar en el display el dígito del número de boleta correspondiente.

3.4.1. Diagrama de Bloques (Esquemático)



3.4.2. Diagrama de Flujo (UML)



3.4.3. Código

```
;/**@brief ESTE PROGRAMA MUESTRA LOS BLOQUES QUE FORMAN UN PROGRAMA
; * EN ENSAMBLADOR, LOS BLOQUES SON:
; * BLOQUE 1. OPCIONES DE CONFIGURACION DEL DSC: OSCILADOR, WATCHDOG,
; * BROWN OUT RESET, POWER ON RESET Y CODIGO DE PROTECCION
```

```
; * BLOQUE 2. EQUIVALENCIAS Y DECLARACIONES GLOBALES
; * BLOQUE 3. ESPACIOS DE MEMORIA: PROGRAMA, DATOS X, DATOS Y, DATOS NEAR
; * BLOQUE 4. C�DIGO DE APLICACI�N
; * @device: DSPIC30F4013
; * @oscilLator: FRC, 7.3728MHz
; */
       .equ __30F3013, 1
       .include "p30F3013.inc"
;*************************************************************************
; BITS DE CONFIGURACI�N
;*************************************************************************
;.........................................................................
;SE DESACTIVA EL CLOCK SWITCHING Y EL FAIL-SAFE CLOCK MONITOR (FSCM) Y SE
;ACTIVA EL OSCILADOR INTERNO DE 7.3728MHZ(FAST RC) PARA TRABAJAR
;FSCM: PERMITE AL DISPOSITIVO CONTINUAR OPERANDO AUN CUANDO OCURRA UNA FALLA
;EN EL OSCILADOR. CUANDO OCURRE UNA FALLA EN EL OSCILADOR SE GENERA UNA TRAMPA
;Y SE CAMBIA EL RELOJ AL OSCILADOR FRC
;.........................................................................
       config __FOSC, CSW_FSCM_OFF & FRC
;.........................................................................
;SE DESACTIVA EL WATCHDOG
;.........................................................................
       config __FWDT, WDT_OFF
;.........................................................................
;SE ACTIVA EL POWER ON RESET (POR), BROWN OUT RESET (BOR), POWER UP TIMER (PWRT)
;Y EL MASTER CLEAR (MCLR)
;POR: AL MOMENTO DE ALIMENTAR EL DSPIC OCURRE UN RESET CUANDO EL VOLTAJE DE
;ALIMENTACI�N ALCANZA UN VOLTAJE DE UMBRAL (VPOR), EL CUAL ES 1.85V
;BOR: ESTE MODULO GENERA UN RESET CUANDO EL VOLTAJE DE ALIMENTACI�N DECAE
;POR DEBAJO DE UN CIERTO UMBRAL ESTABLECIDO (2.7V)
;PWRT: MANTIENE AL DSPIC EN RESET POR UN CIERTO TIEMPO ESTABLECIDO, ESTO AYUDA
;A ASEGURAR QUE EL VOLTAJE DE ALIMENTACI�N SE HA ESTABILIZADO (16ms)
;.........................................................................
       config __FBORPOR, PBOR_ON & BORV27 & PWRT_16 & MCLR_EN
;.........................................................................
;SE DESACTIVA EL C�DIGO DE PROTECCI�N
;.........................................................................
   config __FGS, CODE_PROT_OFF & GWRP_OFF

;*************************************************************************
; SECCI�N DE DECLARACI�N DE CONSTANTES CON LA DIRECTIVA .EQU (= DEFINE EN C)
;*************************************************************************
       .equ MUESTRAS, 64          ;N�MERO DE MUESTRAS

;*************************************************************************
; DECLARACIONES GLOBALES
;*************************************************************************
;.........................................................................
;PROPORCIONA ALCANCE GLOBAL A LA FUNCI�N _wreg_init, ESTO PERMITE LLAMAR A LA
;FUNCI�N DESDE UN OTRO PROGRAMA EN ENSAMBLADOR O EN C COLOCANDO LA DECLARACI�N
;"EXTERN"
;.........................................................................
       .global _wreg_init
;.........................................................................
;ETIQUETA DE LA PRIMER LINEA DE C�DIGO
;.........................................................................
       .global __reset
;.........................................................................
;DECLARACI�N DE LA ISR DEL TIMER 1 COMO GLOBAL
;.........................................................................
       .global __T1Interrupt

;*************************************************************************
;CONSTANTES ALMACENADAS EN EL ESPACIO DE LA MEMORIA DE PROGRAMA
;*************************************************************************
       .section .myconstbuffer, code
;.........................................................................
;ALINEA LA SIGUIENTE PALABRA ALMACENADA EN LA MEMORIA
```

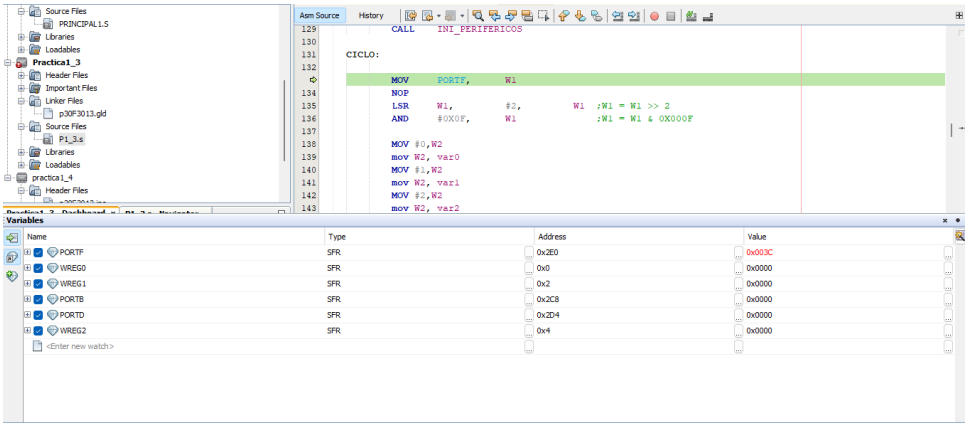
```
;DE PROGRAMA A UNA DIRECCION MULTIPLO DE 2
;.....................................................................................
       .palign 2

ps_coeff:
       .hword   0x0002, 0x0003, 0x0005, 0x000A
;short int ps_coeff[] = {0x0002, 0x0003, 0x0005, 0x000A };

;*************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO X DE LA MEMORIA DE DATOS
;*************************************************************************
        .section .xbss, bss, xmemory

x_input: .space 2*MUESTRAS        ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;char x_input[128];
;short int x_input[64];
;int x_input[32];
;*************************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO Y DE LA MEMORIA DE DATOS
;*************************************************************************

        .section .ybss, bss, ymemory

y_input:  .space 2*MUESTRAS       ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;*************************************************************************
;VARIABLES NO INICIALIZADAS LA MEMORIA DE DATOS CERCANA (NEAR), LOCALIZADA
;EN LOS PRIMEROS 8KB DE RAM
;*************************************************************************
        .section .nbss, bss, near

var1:     .space 2               ;LA VARIABLE VAR1 RESERVA 1 WORD DE ESPACIO
var2:     .space 2
var3:     .space 2
var4:     .space 2
var5:     .space 2
var6:     .space 2
var7:     .space 2
var8:     .space 2
var9:     .space 2
var0:     .space 2

;*************************************************************************
;SECCION DE CODIGO EN LA MEMORIA DE PROGRAMA
;*************************************************************************
.text         ;INICIO DE LA SECCION DE CODIGO

__reset:
       MOV #__SP_init,   W15 ;INICIALIZA EL STACK POINTER

       MOV   #__SPLIM_init,  W0      ;INICIALIZA EL REGISTRO STACK POINTER LIMIT
       MOV   W0,     SPLIM

       NOP                           ;UN NOP DESPUES DE LA INICIALIZACION DE SPLIM

       CALL  _WREG_INIT              ;SE LLAMA A LA RUTINA DE INICIALIZACION DE REGISTROS
                                     ;OPCIONALMENTE USAR RCALL EN LUGAR DE CALL
       CALL    INI_PERIFERICOS
CICLO:
  MOV PORTF,      W1
 NOP
 LSR W1,     #2,   W1  ;W1 = W1 >> 2
 AND #0X0F,      W1        ;W1 = W1 & 0X000F
  MOV #0,W2
 mov W2, var0
 MOV #1,W2
 mov W2, var1
 MOV #2,W2
 mov W2, var2
```

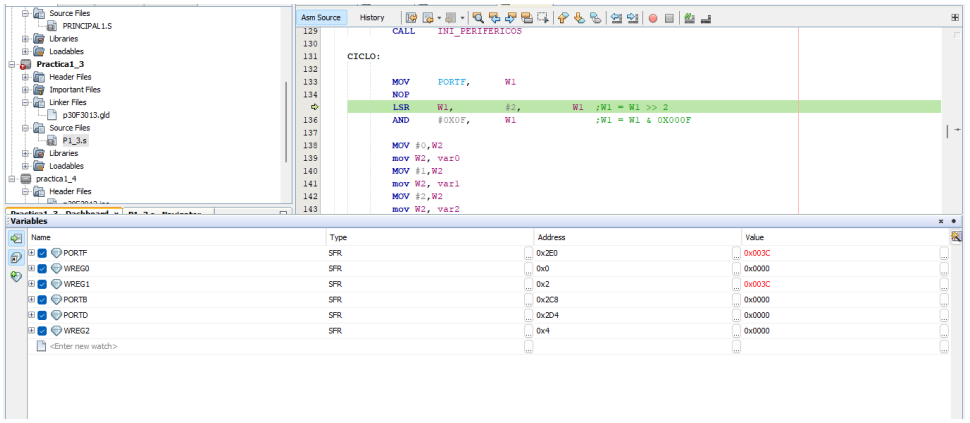
```
MOV #3,W2
mov W2, var3
MOV #4,W2
mov W2, var4
MOV #5,W2
mov W2, var5
MOV #6,W2
mov W2, var6
MOV #7,W2
mov W2, var7
MOV #8,W2
mov W2, var8
MOV #9,W2
mov W2, var9
 MOV var0,W3
CPSNE W1, W3
GOTO DOS
NOP
 MOV var2,W3
CPSNE W1, W3
GOTO UNO
NOP
 MOV var7,W3
CPSNE W1, W3
GOTO CERO
NOP

MOV var1,W3
CPSNE W1, W3
GOTO CERO
NOP
 MOV var3,W3
CPSNE W1, W3
GOTO NUEVE
NOP
 MOV var6,W3
CPSNE W1, W3
GOTO CERO
NOP

MOV var4,W3
CPSNE W1, W3
GOTO TRES
NOP
 MOV var8,W3
CPSNE W1, W3
GOTO OCHO
NOP

MOV var5,W3
CPSNE W1, W3
GOTO CUATRO
NOP

MOV var9,W3
CPSNE W1, W3
GOTO TRES
NOP

MOV var9,W3
CPSGT W1, W3
NOP
GOTO ERROR
NOP

;MOV  #0, W2
;MOV  #1, W3
;MOV  #2, W4
```

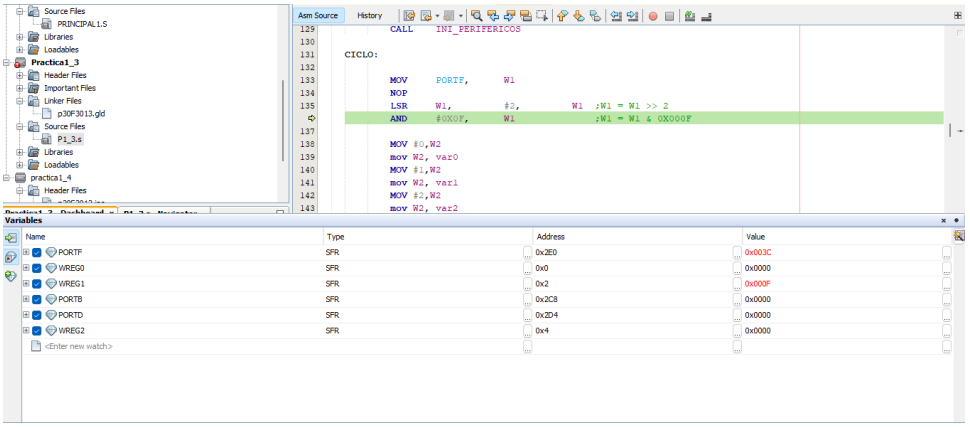
```
 ;MOV  #3, W5
 ;MOV  #4, W6
 ;MOV  #8, W7
 ;MOV  #9, W8
 ;MOV  #0x000A,  W9

 ;CPSNE  W1,     W2
 ;GOTO CERO
 ;NOP
 ;CPSNE  W1,     W3
 ;GOTO UNO
 ;NOP
 ;CPSNE  W1,     W4
 ;GOTO DOS
 ;NOP
 ;CPSNE  W1,     W5
 ;GOTO TRES
 ;NOP
 ;CPSNE  W1,     W6
 ;GOTO CUATRO
 ;NOP
 ;CPSNE  W1,     W7
 ;GOTO OCHO
 ;NOP
 ;CPSNE  W1,     W8
 ;GOTO NUEVE
 ;NOP
 ;CPSGT  W1,     W9
 ;GOTO ERROR
       GOTO    CICLO

CERO:
   MOV     #0X007E,W0
   MOV     W0,     PORTB
   NOP
   GOTO CICLO
UNO:
   MOV     #0X0030,W0
   MOV     W0,     PORTB
   NOP
   GOTO CICLO
DOS:
   MOV     #0X006D,W0
   MOV     W0,     PORTB
   NOP
   GOTO CICLO
TRES:
   MOV     #0X0079,W0
   MOV     W0,     PORTB
   NOP
   GOTO CICLO
CUATRO:
   MOV     #0X0033,W0
   MOV     W0,     PORTB
   NOP
   GOTO CICLO
OCHO:
   MOV     #0X007F,W0
   MOV     W0,     PORTB
   NOP
   GOTO CICLO
NUEVE:
   MOV     #0X0073,W0
   MOV     W0,     PORTB
   NOP
   GOTO CICLO
ERROR:
   MOV     #0X004F,W0
```

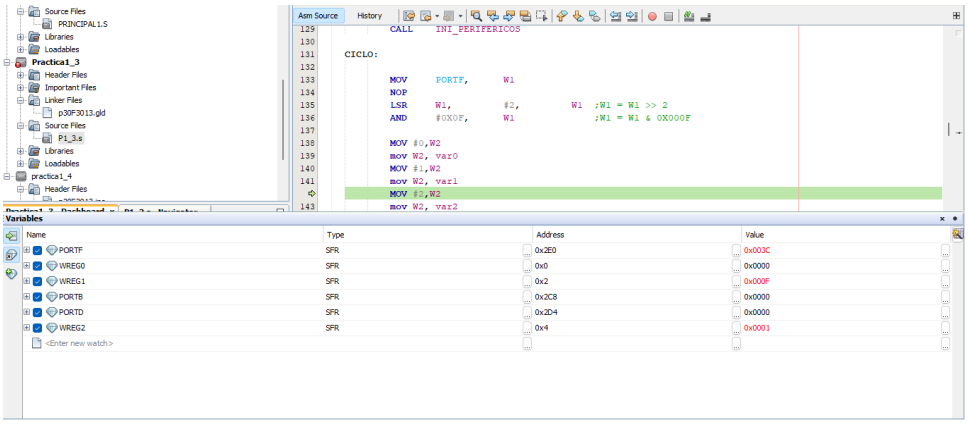
```
   MOV     W0,     PORTB
   NOP
   GOTO CICLO

;/**@brief ESTA RUTINA INICIALIZA LOS PERIFERICOS DEL DSC
; */
INI_PERIFERICOS:
 MOV #0X0000,    W0
 NOP
 MOV W0,     PORTF;PORTF = 0X0000
 NOP
 MOV W0,     LATF;LATF = 0X0000
 NOP
 SETM  TRISF   ;TRISF = 0XFFFF
 NOP
 MOV W0,     PORTB;PORTB = 0X0000
 NOP
 MOV W0,     LATB;LATB = 0X0000
 NOP
 MOV W0,     TRISB;TRISB = 0X0000
 NOP
 SETM  ADPCFG
       RETURN

;/**@brief ESTA RUTINA INICIALIZA LOS REGISTROS Wn A 0X0000
; */
_WREG_INIT:
      CLR   W0
      MOV   W0,         W14
      REPEAT  #12
      MOV   W0,         [++W14]
      CLR   W14
      RETURN

;/**@brief ISR (INTERRUPT SERVICE ROUTINE) DEL TIMER 1
; * SE USA PUSH.S PARA GUARDAR LOS REGISTROS W0, W1, W2, W3,
; * C, Z, N Y DC EN LOS REGISTROS SOMBRA
; */
__T1Interrupt:
      PUSH.S


      BCLR IFS0, #T1IF          ;SE LIMPIA LA BANDERA DE INTERRUPCION DEL TIMER 1

      POP.S

      RETFIE                    ;REGRESO DE LA ISR


.END                             ;TERMINACION DEL CODIGO DE PROGRAMA EN ESTE ARCHIVO
```

3.4.4. Simulación





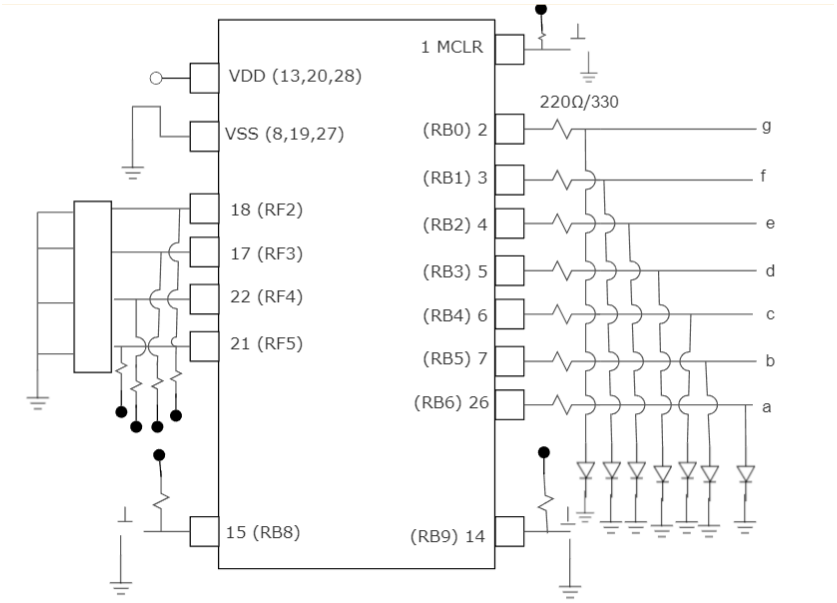




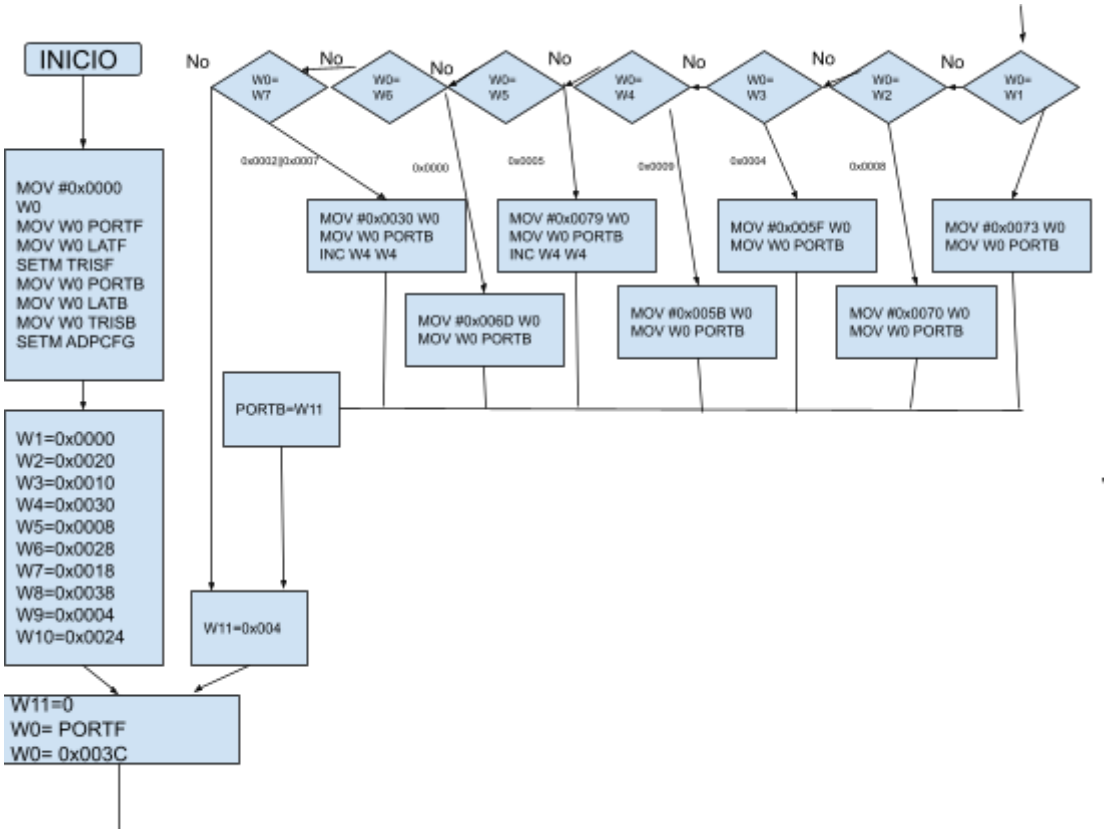
3.5. Cuarto ejercicio

El cuarto ejercicio consiste en retomar el tercer ejercicio dónde se muestra un número de boleta pero, en vez de tomar las entradas a partir de un *dip-switch*, se deberá colocar una interrupción entre cada dígito de un segundo a modo de que se muestre todo el número de boleta en ~10 segundos.

3.5.1. Diagrama de Bloques (Esquemático)



3.5.2. Diagrama de Flujo (UML)



3.5.3. Código

```
;/**@brief ESTE PROGRAMA MUESTRA LOS BLOQUES QUE FORMAN UN PROGRAMA
; * EN ENSAMBLADOR, LOS BLOQUES SON:
; * BLOQUE 1. OPCIONES DE CONFIGURACION DEL DSC: OSCILADOR, WATCHDOG,
; * BROWN OUT RESET, POWER ON RESET Y CODIGO DE PROTECCION
; * BLOQUE 2. EQUIVALENCIAS Y DECLARACIONES GLOBALES
; * BLOQUE 3. ESPACIOS DE MEMORIA: PROGRAMA, DATOS X, DATOS Y, DATOS NEAR
; * BLOQUE 4. C□DIGO DE APLICACI□N
; * @device: DSPIC30F4013
; * @oscilLator: FRC, 7.3728MHz
; */
       .equ __30F3013, 1
       .include "p30F3013.inc"
;*****************************************************************************
; BITS DE CONFIGURACI□N
;*****************************************************************************
;.............................................................................
;SE DESACTIVA EL CLOCK SWITCHING Y EL FAIL-SAFE CLOCK MONITOR (FSCM) Y SE
;ACTIVA EL OSCILADOR INTERNO DE 7.3728MHZ(FAST RC) PARA TRABAJAR
;FSCM: PERMITE AL DISPOSITIVO CONTINUAR OPERANDO AUN CUANDO OCURRA UNA FALLA
;EN EL OSCILADOR. CUANDO OCURRE UNA FALLA EN EL OSCILADOR SE GENERA UNA TRAMPA
;Y SE CAMBIA EL RELOJ AL OSCILADOR FRC
;.............................................................................
       config __FOSC, CSW_FSCM_OFF & FRC
;.............................................................................
;SE DESACTIVA EL WATCHDOG
;.............................................................................
       config __FWDT, WDT_OFF
;.............................................................................
;SE ACTIVA EL POWER ON RESET (POR), BROWN OUT RESET (BOR), POWER UP TIMER (PWRT)
;Y EL MASTER CLEAR (MCLR)
;POR: AL MOMENTO DE ALIMENTAR EL DSPIC OCURRE UN RESET CUANDO EL VOLTAJE DE
;ALIMENTACI□N ALCANZA UN VOLTAJE DE UMBRAL (VPOR), EL CUAL ES 1.85V
```

```
;BOR: ESTE MODULO GENERA UN RESET CUANDO EL VOLTAJE DE ALIMENTACI□N DECAE
;POR DEBAJO DE UN CIERTO UMBRAL ESTABLECIDO (2.7V)
;PWRT: MANTIENE AL DSPIC EN RESET POR UN CIERTO TIEMPO ESTABLECIDO, ESTO AYUDA
;A ASEGURAR QUE EL VOLTAJE DE ALIMENTACI□N SE HA ESTABILIZADO (16ms)
;-----------------------------------------------------------------------
       config __FBORPOR, PBOR_ON & BORV27 & PWRT_16 & MCLR_EN
;-----------------------------------------------------------------------
;SE DESACTIVA EL C□DIGO DE PROTECCI□N
;-----------------------------------------------------------------------
   config __FGS, CODE_PROT_OFF & GWRP_OFF

;***********************************************************************
; SECCI□N DE DECLARACI□N DE CONSTANTES CON LA DIRECTIVA .EQU (= DEFINE EN C)
;***********************************************************************
       .equ MUESTRAS, 64          ;N□MERO DE MUESTRAS

;***********************************************************************
; DECLARACIONES GLOBALES
;***********************************************************************
;-----------------------------------------------------------------------
;PROPORCIONA ALCANCE GLOBAL A LA FUNCI□N _wreg_init, ESTO PERMITE LLAMAR A LA
;FUNCI□N DESDE UN OTRO PROGRAMA EN ENSAMBLADOR O EN C COLOCANDO LA DECLARACI□N
;"EXTERN"
;-----------------------------------------------------------------------
       .global _wreg_init
;-----------------------------------------------------------------------
;ETIQUETA DE LA PRIMER LINEA DE C□DIGO
;-----------------------------------------------------------------------
       .global __reset
;-----------------------------------------------------------------------
;DECLARACI□N DE LA ISR DEL TIMER 1 COMO GLOBAL
;-----------------------------------------------------------------------
       .global __T1Interrupt

;***********************************************************************
;CONSTANTES ALMACENADAS EN EL ESPACIO DE LA MEMORIA DE PROGRAMA
;***********************************************************************
       .section .myconstbuffer, code
;-----------------------------------------------------------------------
;ALINEA LA SIGUIENTE PALABRA ALMACENADA EN LA MEMORIA
;DE PROGRAMA A UNA DIRECCION MULTIPLO DE 2
;-----------------------------------------------------------------------
       .palign 2

ps_coeff:
       .hword   0x0002, 0x0003, 0x0005, 0x000A
;short int ps_coeff[] = {0x0002, 0x0003, 0x0005, 0x000A };

;***********************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO X DE LA MEMORIA DE DATOS
;***********************************************************************
        .section .xbss, bss, xmemory

x_input: .space 2*MUESTRAS        ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;char x_input[128];
;short int x_input[64];
;int x_input[32];
;***********************************************************************
;VARIABLES NO INICIALIZADAS EN EL ESPACIO Y DE LA MEMORIA DE DATOS
;***********************************************************************

         .section .ybss, bss, ymemory

y_input:  .space 2*MUESTRAS       ;RESERVANDO ESPACIO (EN BYTES) A LA VARIABLE
;***********************************************************************
;VARIABLES NO INICIALIZADAS LA MEMORIA DE DATOS CERCANA (NEAR), LOCALIZADA
;EN LOS PRIMEROS 8KB DE RAM
;***********************************************************************
```

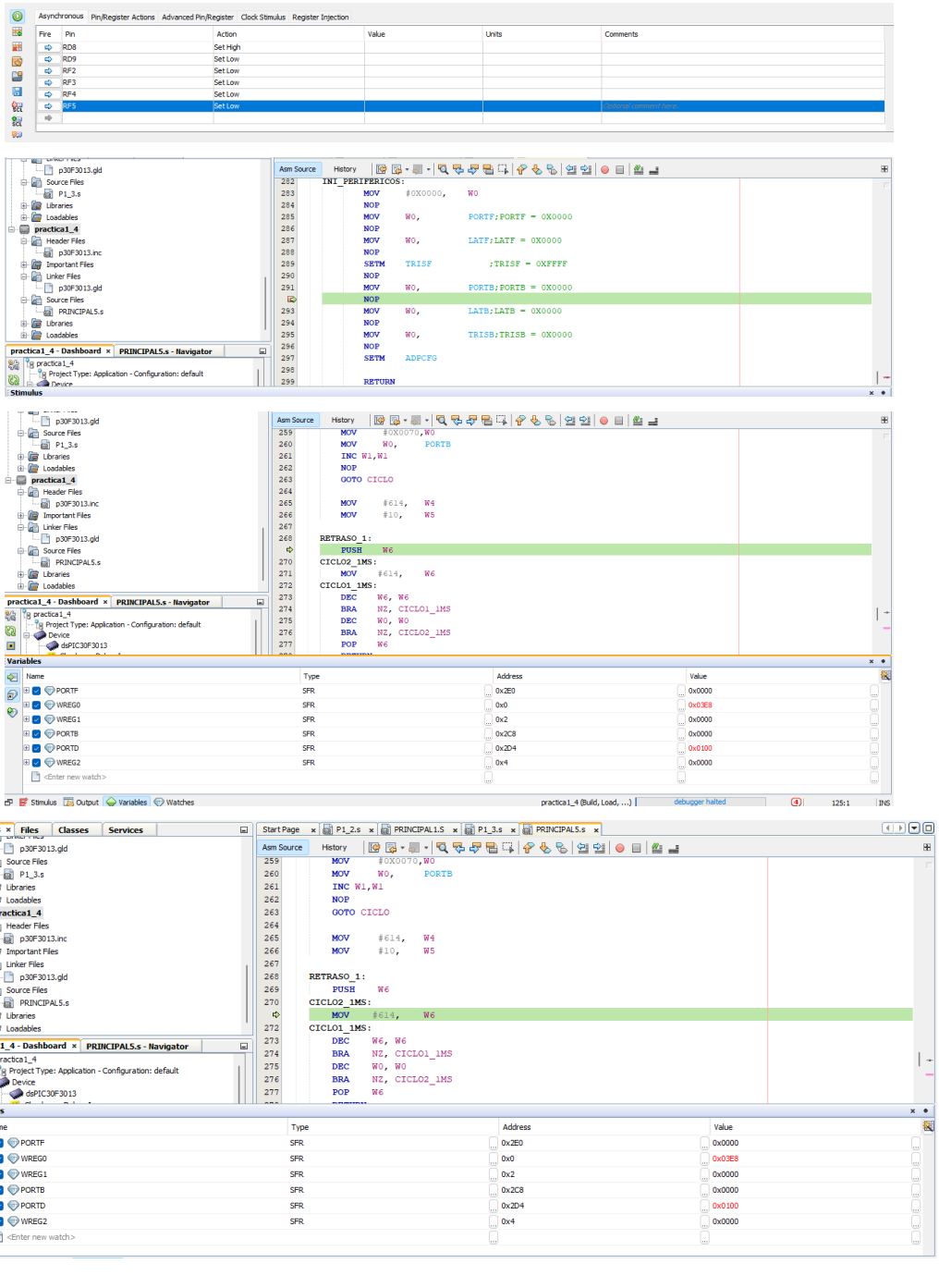
```
        .section .nbss, bss, near

var1:     .space 2              ;LA VARIABLE VAR1 RESERVA 1 WORD DE ESPACIO
var2:     .space 2
var3:     .space 2
var4:     .space 2
var5:     .space 2
var6:     .space 2
var7:     .space 2
var8:     .space 2
var9:     .space 2
var0:     .space 2

;*****************************************************************************
;SECCION DE CODIGO EN LA MEMORIA DE PROGRAMA
;*****************************************************************************
.text         ;INICIO DE LA SECCION DE CODIGO

__reset:
       MOV #__SP_init,   W15 ;INICIALIZA EL STACK POINTER

       MOV   #__SPLIM_init,  W0      ;INICIALIZA EL REGISTRO STACK POINTER LIMIT
       MOV   W0,     SPLIM

       NOP                          ;UN NOP DESPUES DE LA INICIALIZACION DE SPLIM

       CALL  _WREG_INIT             ;SE LLAMA A LA RUTINA DE INICIALIZACION DE REGISTROS
                                    ;OPCIONALMENTE USAR RCALL EN LUGAR DE CALL
       CALL    INI_PERIFERICOS
CICLO:
  PUSH    W0
 MOV    #1000,    W0
 CALL   RETRASOBOLETA
 POP    W0
  MOV #0,W2
 mov W2, var0
 MOV #1,W2
 mov W2, var1
 MOV #2,W2
 mov W2, var2
 MOV #3,W2
 mov W2, var3
 MOV #4,W2
 mov W2, var4
 MOV #5,W2
 mov W2, var5
 MOV #6,W2
 mov W2, var6
 MOV #7,W2
 mov W2, var7
 MOV #8,W2
 mov W2, var8
 MOV #9,W2
 mov W2, var9
  MOV var0,W3
 CPSNE W1, W3
 GOTO DOS
 NOP
  MOV var2,W3
 CPSNE W1, W3
 GOTO UNO
 NOP
  MOV var7,W3
 CPSNE W1, W3
 GOTO CERO
 NOP

 MOV var1,W3
```

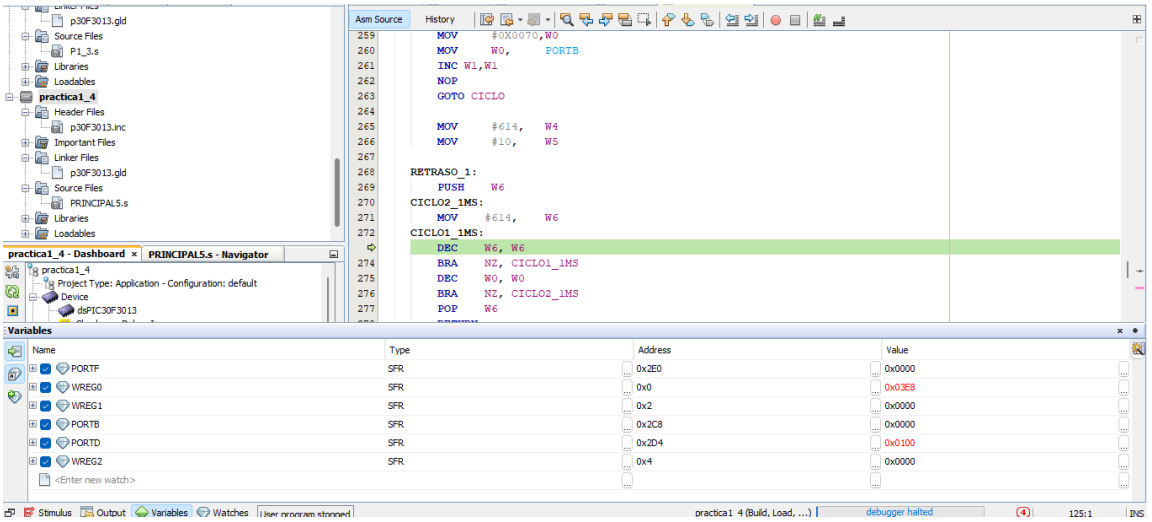
```
 CPSNE W1, W3
 GOTO CERO
 NOP
  MOV var3,W3
 CPSNE W1, W3
 GOTO NUEVE
 NOP
  MOV var6,W3
 CPSNE W1, W3
 GOTO CERO
 NOP

 MOV var4,W3
 CPSNE W1, W3
 GOTO TRES
 NOP
  MOV var8,W3
 CPSNE W1, W3
 GOTO OCHO
 NOP

 MOV var5,W3
 CPSNE W1, W3
 GOTO CUATRO
 NOP

 MOV var9,W3
 CPSNE W1, W3
 GOTO TRES
 NOP

 MOV var9,W3
 CPSGT W1, W3
 NOP
 MOV #0, W4
 NOP


        GOTO    CICLO

CERO:
   MOV     #0X007E,W0
   MOV     W0,     PORTB
   INC W4,W4
   NOP
   GOTO CICLO
UNO:
   MOV     #0X0030,W0
   MOV     W0,     PORTB
   INC W4,W4
   NOP
   GOTO CICLO
DOS:
   MOV     #0X006D,W0
   MOV     W0,     PORTB
   INC W4,W4
   NOP
   GOTO CICLO
TRES:
   MOV     #0X0079,W0
   MOV     W0,     PORTB
   INC W4,W4
   NOP
   GOTO CICLO
CUATRO:
   MOV     #0X0033,W0
   MOV     W0,     PORTB
   INC W4,W4
```

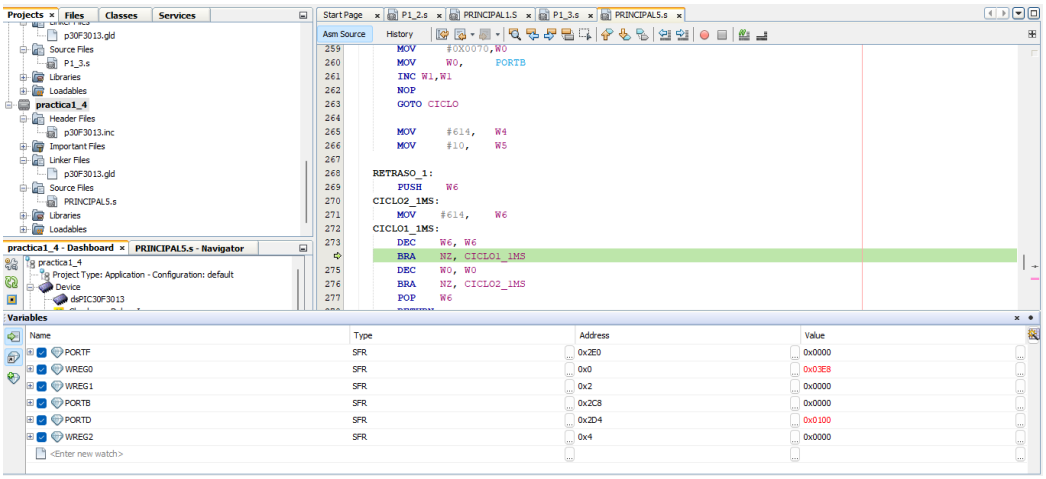
```
   NOP
   GOTO CICLO
OCHO:
   MOV     #0X007F,W0
   MOV     W0,     PORTB
   INC W4,W4
   NOP
   GOTO CICLO
NUEVE:
   MOV     #0X0073,W0
   MOV     W0,     PORTB
   INC W4,W4
   NOP
   GOTO CICLO

   MOV     #614,   W5
   MOV     #10,    W6
RETRASOBOLETA:
   PUSH    W1
CICLO2_1MS:
   MOV     #614,   W1
CICLO1_1MS:
   DEC     W1, W1
   BRA     NZ, CICLO1_1MS
   DEC     W0, W0
   BRA     NZ, CICLO2_1MS
   POP     W1
   RETURN

;/**@brief ESTA RUTINA INICIALIZA LOS PERIFERICOS DEL DSC
; */
INI_PERIFERICOS:
 MOV #0X0000,    W0
 MOV #0X0000,    W4
 NOP
 MOV W0,     PORTF;PORTF = 0X0000
 NOP
 MOV W0,     LATF;LATF = 0X0000
 NOP
 SETM  TRISF   ;TRISF = 0XFFFF
 NOP
 MOV W0,     PORTB;PORTB = 0X0000
 NOP
 MOV W0,     LATB;LATB = 0X0000
 NOP
 MOV W0,     TRISB;TRISB = 0X0000
 NOP
 SETM  ADPCFG
       RETURN

;/**@brief ESTA RUTINA INICIALIZA LOS REGISTROS Wn A 0X0000
; */
_WREG_INIT:
       CLR   W0
       MOV   W0,         W14
       REPEAT  #12
       MOV   W0,         [++W14]
       CLR   W14
       RETURN

;/**@brief ISR (INTERRUPT SERVICE ROUTINE) DEL TIMER 1
; * SE USA PUSH.S PARA GUARDAR LOS REGISTROS W0, W1, W2, W3,
; * C, Z, N Y DC EN LOS REGISTROS SOMBRA
; */
__T1Interrupt:
       PUSH.S
```

```
        BCLR IFS0, #T1IF          ;SE LIMPIA LA BANDERA DE INTERRUPCION DEL TIMER 1

        POP.S

        RETFIE                    ;REGRESO DE LA ISR


.END                              ;TERMINACION DEL CODIGO DE PROGRAMA EN ESTE ARCHIVO
```

3.5.4. Simulación




p30F3013.gld
Source Files
P1_3.s
Libraries
Loadables
practica1_4
Header Files
p30F3013.inc
Important Files
Linker Files
p30F3013.gld
Source Files
PRINCIPAL5.s
Libraries
Loadables
practica1_4 - Dashboard
PRINCIPAL5.s - Navigator
practica1_4
Project Type: Application - Configuration: default
Device
dsPIC30F3013
Asm Source
History
259 MOV #0X0070,W0
260 MOV W0, PORTB
261 INC W1,W1
262 NOP
263 GOTO CICLO
264
265 MOV #614, W4
266 MOV #10, W5
267
268 RETRASO_1:
269 PUSH W6
270 CICLO2_1MS:
271 MOV #614, W6
272 CICLO1_1MS:
273 DEC W6, W6
274 BRA NZ, CICLO1_1MS
275 DEC W0, W0
276 BRA NZ, CICLO2_1MS
277 POP W6
Variables
Name
Type
Address
Value
PORTF SFR 0x2E0 0x0000
WREG0 SFR 0x0 0x03E8
WREG1 SFR 0x2 0x0000
PORTB SFR 0x2C8 0x0000
PORTD SFR 0x2D4 0x0100
WREG2 SFR 0x4 0x0000
<Enter new watch>
Stimulus
Output
Variables
Watches
User program stopped
practica1 4 (Build, Load, ...)
debugger halted
125:1
INS


Projects
Files
Classes
Services
Start Page
P1_2.s
PRINCIPAL1.S
P1_3.s
PRINCIPAL5.s
p30F3013.gld
Source Files
P1_3.s
Libraries
Loadables
practica1_4
Header Files
p30F3013.inc
Important Files
Linker Files
p30F3013.gld
Source Files
PRINCIPAL5.s
Libraries
Loadables
practica1_4 - Dashboard
PRINCIPAL5.s - Navigator
practica1_4
Project Type: Application - Configuration: default
Device
dsPIC30F3013
Asm Source
History
259 MOV #0X0070,W0
260 MOV W0, PORTB
261 INC W1,W1
262 NOP
263 GOTO CICLO
264
265 MOV #614, W4
266 MOV #10, W5
267
268 RETRASO_1:
269 PUSH W6
270 CICLO2_1MS:
271 MOV #614, W6
272 CICLO1_1MS:
273 DEC W6, W6
274 BRA NZ, CICLO1_1MS
275 DEC W0, W0
276 BRA NZ, CICLO2_1MS
277 POP W6
Variables
Name
Type
Address
Value
PORTF SFR 0x2E0 0x0000
WREG0 SFR 0x0 0x03E8
WREG1 SFR 0x2 0x0000
PORTB SFR 0x2C8 0x0000
PORTD SFR 0x2D4 0x0100
WREG2 SFR 0x4 0x0000
<Enter new watch>

4. Hoja de Firmas

Instituto Politécnico Nacional
Escuela Superior de Cómputo

ESCOM

Profesor:	Adbel Anahí Montes Meza
Grupo:	3CM13
Equipo:	10
Integrantes:	Montaño Reyes Jennifer Pardo Alanis Arturo Isaac Ortega Alcocer Humberto Alejandro

Ejercicio	Fecha de entrega	Funcionó	Firma
1. Lectura/escritura puerto	14/Nov/22	✓	Montes
2. Calculadora	18/Nov/22	✓	Montes
3. Boleta en display con switch	16/Nov/22	✓	Montes
4. Boleta en display con contador	23/Nov/22	✓	Montes

5. Conclusiones

5.1. Montaño Reyes Jennifer

Las prácticas ayudaron a comprender las diferentes capacidades que presenta un microcontrolador, averiguamos el uso para su programación y su construcción de manera física. Comprendimos temas con los que no habíamos tenido acercamiento, de modo que fue de gran importancia para las siguientes prácticas que se avecinan.

5.2. Pardo Alanis Arturo Isaac

Con la realización de estos ejercicios de esta primera práctica, sirvió como un buen ejemplo para dar como primer acercamiento al uso de los circuitos de microcontroladores y de MPLAB, ya que estos ejercicios fueron diversos por lo que hacían cada uno de estos ejercicios. También gracias a esta práctica se pudo hacer uso de los conocimientos que hemos visto desde el comienzo de la materia y ha sido de gran utilidad para reafirmar y comprender de mejor manera lo visto en clase.

5.3. Ortega Alcocer Humberto Alejandro

En esta práctica fue donde realmente comencé a ver el uso de un microcontrolador físico. Hasta ahora, y gracias en gran medida a la pandemia, toda mi interacción durante la carrera con circuitos y electrónica en general fue mediante simuladores. El poder trabajar físicamente con componentes, circuitos y sus debidas consideraciones (y correcciones) me ayudó a asentar de mejor forma los conocimientos vistos en clase y a tener muchos más cuidado en el diseño de mis circuitos pues, como pude ver en el desarrollo de esta práctica, requiere de mucho cuidado para no dañar los componentes.