Microcontroladores

Profesor: Kalun José Lau Gan

Semestre 2025-2

Semana 9-10

Coordinaciones previas:

- Con respecto al TP: Las diapositivas tienen que ser basadas en el informe, tener coherencia (si lo que se expone tiene sentido), contexto, etc.
- Lineamientos del TF se publicarán este fin de semana, recordar que el TF es la continuación del TP (el desarrollo del proyecto)
  - Se añadirán los capítulos de Desarrollo, Validación, Resultados, Conclusiones y Recomendaciones.
- Deberán de hacer las correcciones de su TP tanto INF como PPT de acuerdo a la retroalimentación brindada y formará parte de la evaluación de PC2.
- Evaluaciones pendientes: LB3 (sem12), PC2 (sem14), DD (sem15), TF (sem16).
- Cambio de modalidad de trabajo en los Laboratorios: Todos los laboratorios dirigidos en adelante serán evaluados, formarán parte de los laboratorios calificados.

¿Preguntas previas?

- ¿El hecho de usar C lo hace más fácil?
  - El proceso de diseño es el mismo tanto para lenguaje Assembler como en lenguaje C, se tiene que implementar primero el hardware, desarrollar el algoritmo y finalmente aterrizar el algoritmo en un lenguaje de programación.
  - El lenguaje XC va a permitir el uso de librerías y funciones especializadas, además de manejar mejor las operaciones matemáticas y manipulación de datos.
  - El lenguaje XC es mas familiar debido a que es un lenguaje orientado al desarrollador (alto nivel) y no a la máquina (bajo nivel)

Quiero hacer un reloj, pero solo se desarrollar aplicaciones con Arduino

- El reloj debe de ser preciso

Ahora, vemos como hacer un reloj con microcontrolador PIC

- El reloj debe de ser preciso:
  - Usar XTAL 32.768KHz con el Timer1
  - Usar HFINTOSC a 32MHz y subsiguientes divisiones de dicha frecuencia

Preguntas previas:

- En el ejemplo de la semana pasada, el estar haciendo el cambio de velocidad me dí cuenta que demora en actualizar dicho cambio de velocidad aún utilizando interrupciones.
  - Esto se debe a que muy probablemente hayas empleado __delay_ms() para realizar el periodo de parpadeo de la aplicación. El CPU igual va a terminar de ejecutar dicha función aún haya habido una interrupción. Esto se nota mas cuando los retardo son muy prolongados.
  - Se recomienda para estos casos emplear un temporizador (ej Timer0) para hacer la labor de temporizado del tiempo y cada vez que ocurra una interrupción se actualiza el periodo de cuenta del temporizador y reiniciar su cuenta.

Preguntas previas:

- Como se haría para implementar un sistema electrónico basado en PDI para detectar una persona?
  - Utilizar una SBC como un Raspberry Pi para hacer la aplicación
  - La ventaja de usar RPi es que tiene un puerto dedicado para conectar una cámara y/o puertos USB.
  - La RPi funciona con un sistema operativo.
  - La solución es cara.
- Otra alternativa mas barata: ESP32 CAM

Agenda

- Tecnología de periféricos independientes de Microchip
- Lenguajes de alto nivel en el desarrollo con microcontroladores
  - El XC8 de Microchip
- Herramientas de apoyo al desarrollo
  - MPLAB Xpress (va a ser descontinuado en dic 2025)
  - MCC (Microchip Code Configurator)
- Mi primer proyecto en lenguaje XC8 de alto nivel en el MPLAB X IDE
- Uso del LCD alfanumérico 16x2 HD44780
- El conversor ADC del PIC18F57Q43
- El conversor DAC del PIC18F57Q43

Tecnología de Periféricos Independientes (Core-Independent Peripherals – CIP)

- Independizando el funcionamiento de los periféricos en el microcontrolador se logra un mejor rendimiento del CPU
- Los periféricos independientes administran sus tareas sin código ni supervisión del CPU para mantener su funcionamiento

Tecnología de Periféricos Independientes (Core-Independent Peripherals – CIP)

- Dependiendo del modelo y familia el microcontrolador tendrá un conjunto de periféricos CIP
- Los CIPs poseen conexiones adicionales con otros CIP para lograr la independencia del CPU
- Estos CIP se apoyan en el sistema de interconexión de pines (PPS)

El PPS en el PIC18F57Q43

- Referencia: capítulo 21 del datasheet
- Sistema de asignación personalizada de señales de E/S desde o hacia los periféricos.
- Tener en consideración las tablas 21-1 (PPS Inputs) y 21-2 (PPS Outputs) para las asignaciones por defecto y el alcance de la personalización

Panorama de los lenguajes de alto nivel para microcontroladores

Cada dispositivo microcontrolador tendrá sus propias plataformas de lenguaje de programación.

Va a depender del desarrollador del lenguaje de alto nivel para determinado microcontrolador (No todos los desarrolladores soportan todos los dispositivos de un fabricante)

Hay varias compañías que desarrollan lenguajes de alto nivel para un microcontrolador.

Problema de compatibilidad entre distintas plataformas de desarrollo aún empleando un mismo lenguaje.

- Basic
- C
- Java (muy poco utilizado y soportado)
- Python (Rpi Pico RP2040 y en RP2350, y el ESP32)

Panorama de los lenguajes de alto nivel para microcontroladores

- Los compiladores de lenguajes de alto nivel (de terceros) mas empleados en la actualidad.
- Estas empresas ofrecen versiones DEMO para que el usuario pueda probar las bondades de cada uno.
- Cada empresa ofrece un pack de librerías especializadas para crear aplicaciones con periféricos especializados en el menor tiempo.

El compilador XC de Microchip

- Disponible versión "gratis" sin límite.
- La versión PRO no es gratis. Provee optimización de código compilado para que ocupe menos espacio.
- Tres versiones:
  - XC8 – PIC10, PIC12, PIC16, PIC18
  - XC16 – PIC24, dsPIC
  - XC32 – PIC32

El compilador XC de Microchip

- Los XC8, XC16 y XC32 vienen separados…
- Ej. El bootloader HID para el microcontrolador PIC18F4550 requiere que se compile el código en XC8 PRO para que entre en su memoria.

Part Number: SW006021-DGL - MPLAB XC8 Compiler PRO Dongle License

The MPLAB XC8 is a full-featured, highly-optimized ANSI C compiler for all 8-bit AVR® and PIC® MCUs. This compiler integrates into Microchip's MPLAB(R) X IDE, is compatible with all Microchip debuggers and emulators, and runs on Windows, Linux and Mac OS X.

The dongle license is a USB flash drive that contains a single-user encrypted license. It is a perpetual license and unlocks PRO optimizations for all versions of the MPLAB XC8 compilers, **version 1.41 and later**, and does not include High Priority Access (HPA). The Dongle License allows a user to unlock PRO optimizations on any computer it is plugged into. If lost, the Dongle License can be replaced one time for a processing fee of $200 and the dongle must be registered to the user.

More Info »

Standard Pricing:

Order Quantity	USD per Unit
1+	$1,695.00

In Stock: **9**
Delivery and scheduling options available in the cart

Order now, up to **9** can ship on **20-May-2020**

Lead Time For Additional Quantities
Additional quantities can ship by **11-Aug-2020**

Quantity:

Caso:

Optimización en el compilador XC

- **Optimization levels:**
  - Level 0: Fastest compilation time, minimal optimizations
  - Level 1: Optimizations with low debugging impact
  - Level 2: All optimizations that give the best balance between speed and size
  - Level 3: Program execution will be as fast as possible
  - Level s: Code size will be as small as possible
- **Where available use:**
  - Use Whole-program and Link-time setting
  - Procedural abstraction

- En la versión gratuita del XC tenemos optimización hasta el nivel 2
- Por defecto el nivel de optimización esta en 0.
- Para acceder al nivel 3 o nivel s requerimos tener la licencia PRO

Optimización en el compilador XC

- Para acceder a las opciones de optimización ingresamos a las propiedades del proyecto:

Eficiencia de código en Assembler vs C

¿Por qué Assembler y por qué C?

El MPLAB Xpress

- Link: https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xpress
- Entorno de trabajo online
- Permite realizar proyectos en XC sin tener que instalar el MPLAB X en el computador.
- No soporta assembler
- UPDATE: Esta herramienta será retirada en Diciembre 2025

El MPLAB Xpress

- Los proyectos se almacenan en la nube de Microchip (en la actualidad el tiempo de almacenamiento es limitado)
- Soporte solo para algunos modelos de microcontrolador PIC

El Microchip Code Configurator (MCC)

- Entorno de programación/configuración gráfico
- Interface intuitiva para desarrollos rápidos de prototipos
- Configuración automatizada para periféricos y funciones
  - Minimiza el uso de la hoja técnica para configuraciones
  - Reduce el esfuerzo y tiempo dedicado a la configuración
- Como desventaja: Va a ocupar mas espacio en la memoria de programa del micro debido a que incluye procedimientos que no necesariamente va a ejecutar el CPU.

El Microchip Code Configurator (MCC)

- Caso de uso

MPLAB X: Creación de un proyecto en XC8 (lenguaje de alto nivel)

- Link del manual de XC8:
  - http://ww1.microchip.com/downloads/en/devicedoc/50002053g.pdf
- Link de descarga del XC8 v2.50:
  - https://www.microchip.com/mplabxc8windows

Referencia de plantilla:

- La plantilla para los códigos en C se ha basado en la plantilla de programas en Arduino IDE:
  - Uso de función setup() donde se coloca los aspectos iniciales de configuración antes de correr el programa de la aplicación. En XC8 crearemos una función similar. Por ejemplo configuración()
  - Uso de función loop() donde se detalla el programa de la aplicación. En XC8 usaremos la función main() donde dentro llamaremos a la función configuración antes de detallar el programa de usuario.

Plantilla de código en XC8:

```
#include <xc.h>

#define _XTAL_FREQ 48000000UL   //Frecuencia de trabajo 48MHz

void configuracion(void) {
    //Aqui colocas las configuraciones iniciales
}

void main(void) {
    configuracion();
    while (1) {
        //Tu programa de usuario
    }
}
```

Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4

Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4

- Implementación en físico con el Curiosity Nano PIC18F57Q43

- Bits de configuración modificados:

```
#pragma config FEXTOSC = OFF    // External Oscillator Selection (Oscillator not enabled)
#pragma config PWRTS = PWRT_64  // Power-up timer selection bits (PWRT set at 64ms)
#pragma config BOREN = OFF      // Brown-out Reset Enable bits (Brown-out Reset disabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (HV on MCLR/VPP must be used for programming)
#pragma config WDTE = OFF       // WDT operating mode (WDT Disabled; SWDTEN is ignored)
```

Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4

- Código en XC8 (con HFINTOSC a 48MHz)

```c
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include "cabecera.h"

#define _XTAL_FREQ 48000000UL

void configuro(void){
    //configuracion del reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x07;      //HFINTOSC a 48MHz
    OSCEN = 0x40;
    //configuracion de las E/S
    TRISBbits.TRISB0 = 0;   //RB0 como salida
    ANSELBbits.ANSELB0 = 0; //RB0 como digital
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup activado
}
```

```c
void main(void){
    configuro();
    while(1){
        if(PORTBbits.RB4 == 0){
            //parpadeo rapido
            LATBbits.LATB0 = 1;     //enciendo LED
            __delay_ms(100);
            LATBbits.LATB0 = 0;     //apago LED
            __delay_ms(100);
        }
        else{
            //parpadeo lento
            LATBbits.LATB0 = 1;     //enciendo LED
            __delay_ms(200);
            LATBbits.LATB0 = 0;     //apago LED
            __delay_ms(200);
        }
    }
}
```

Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4

- Procedimiento para implementar este ejemplo utilizando el MPLAB Xpress
  - URL: https://www.mplab-xpresside.microchip.com/
  - Debes de registrarte para crear una cuenta dentro de dicha plataforma
  - Seguir las mismas indicaciones para crear un proyecto como en el MPLABX

Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4

- Procedimiento para implementar este ejemplo utilizando el MPLAB Xpress

1º Presionamos el boton "Make and Program Device"

2. Se abrirá una ventana de explorador de archivos
3: Conectar el Curiosity Nano a la PC
4: Aparecerá una unidad de disco "Curiosity"
5: Escoger dicha unidad para grabar el archivo HEX

Ejemplo en XC8: Negador lógico de un bit

```c
#include <xc.h>
#define _XTAL_FREQ 48000000UL

void init_conf(void){
    //Aca colocaremos las configuraciones iniciales de la aplicacion
    //TRISDbits.RD0 = 0;            // RD0 como salida
    asm("bcf TRISD, 0");            // Escribiendo instrucciones en mpasm
}

void main(void) {
    init_conf();
    while(1){
        if (PORTBbits.RB0 == 1) {
            LATDbits.LD0 = 0;
        }
        else{
            LATDbits.LD0 = 1;
        }
    }
}
```

Ejemplo en XC8: Compuerta XOR

```c
void configuracion(void) {
    TRISDbits.RD0 = 0;          //Puerto RD0 como salida
}

void main(void) {
    configuracion();
    while(1) {
        //Bucle infinito
        //Con switch-case
        switch (PORTB & 0x03) {
            case 0x00:
                LATDbits.LD0 = 0;
                break;
            case 0x01:
                LATDbits.LD0 = 1;
                break;
            case 0x02:
                LATDbits.LD0 = 1;
                break;
            case 0x03:
                LATDbits.LD0 = 0;
                break;
        }
    }
}
```

El LCD alfanumérico HD44780

- Basado en el controlador Hitachi HD44780A
- Diferentes tamaños, desde 1x8 hasta 4x40
- Interface paralela de datos (4 ó 8 bits)
- Tiene control de contraste y luz de fondo
- Posee un ROM de caracteres predefinidos

The holy grail of LCD controllers: HD44780A02

El LCD alfanumérico HD44780

- ROM de caracteres A00:
  - Muy similar al código ASCII en 7 bits
  - El símbolo de grado (°) en ASCII es Alt+0167, en el ROM de caracteres del HD44780 es 0xDF
  - El símbolo “ñ” en ASCII es Alt+164, en el ROM de caracteres del HD44780 es 0xEE
  - Capacidad de ocho caracteres personalizados (CGRAM 0x00-0x07)
  - Para enviar un carácter directamente al display se emplea la función ENVIA_CHAR() (librería S_SAL)

El LCD alfanumérico HD44780

- ROM de caracteres A02:
  - Posee caracteres adicionales que son utilizados en occidente.
  - Los HD44780 con ROM code A02 son extremadamente escasos.

El LCD alfanumérico HD44780

- Tabla de caracteres ASCII de 7 bits
- Ref. http://www.asciitable.com

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

El LCD alfanumérico HD44780

- Referencia: Hoja técnica del HD44780
  - http://academy.cba.mit.edu/classes/output_devices/44780.pdf
- Para trabajar con el display se ha creado una librería de comandos (desarrollado por Sergio Salas y Kalun Lau) en la cual posee las siguientes características:
  - Interface de 4 bits
  - Comandos para: Limpiar pantalla, ocultar cursor, pasar de línea, caracteres personalizados, etc.
  - Puerto D empleado (RD0→RS, RD1→RW, RD2→E, RD4→D4, RD5→D5, RD6→D6, RD7→D7)
  - Tener en cuenta FOSC especificado dentro de la librería (4MHz). Esto se encuentra en LCD.h en el #define _XTAL_FREQ
- Video de manipulación de LCD sin microcontrolador:
  - https://www.youtube.com/watch?v=cXpeTxC3_A4

Uso del LCD en XC8 (librería S_SAL)

- **Crear primero el Proyecto en el MPLABX**

descargas

Explorador de archivos de Windows

añades los archivos a tu proyecto en el MPLAB X

desempaquetar

copias a la carpeta del proyecto

```
#include <xc.h>
#include "LCD.h"

#define _XTAL_FREQ 48000000UL   //Frecuencia de trabajo 48MHz
```

llamar a la librería

ANSELD = 0x00;

```
void lcd_init(void) {
    TRISD = 0x00;
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}
```

armar la sgte función

Circuito ejemplo de conexión del PIC18F57Q43 con el LCD

Recomendaciones en la interface del LCD con el microcontrolador

- Los pines deben de estar soldados correctamente!
- Seguir estrictamente el procedimiento para el llamado de las librerías.
- El potenciómetro debe de estar correctamente instalado

Interface del LCD 2x16 HD44780 en I2C

- Se le agrega un módulo adicional al LCD y permite la interface a dicho LCD usando I2C (comunicación serial), dicho módulo está basado en el circuito integrado PCF8574 (expansor de puertos vía I2C)
- Permite minimizar la cantidad de pines entre el LCD y el microcontrolador.
- Se requiere implementar una librería para poder lograr la interface
- Es más fácil, pero tiene un costo mayor

Ejemplo: Visualizar “Hola mundo UPC!” en el LCD

- Hardware

Ejemplo: Visualizar “Hola mundo UPC!” en el LCD

- Hardware

Ejemplo: Visualizar “Hola mundo UPC!” en el LCD

- Se ha agregado un mensaje adicional en la segunda línea del LCD
- Tener en cuenta que se está trabajando a 48MHz como fuente de reloj al CPU
- Revisar si la librería del LCD también se encuentre el _XTAL_FREQ a 48MHz

```c
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

void configuro(void){
    //configuracion del reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x07;      //HFINTOSC a 48MHz
    OSCEN = 0x40;
    //configuracion de las E/S
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void main(void){
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Hola mundo UPC!",15);
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Kalun Lau Gan",13);
    while(1){
        //aqui va el codigo de la aplicacion
    }
}
```

Juntando ambos ejemplos (titileo de LED e impresión de mensajes en el LCD)

```c
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

void configuro(void){
    //configuracion del reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x07;      //HFINTOSC a 48MHz
    OSCEN = 0x40;
    //configuracion de las E/S
    TRISFbits.TRISF3 = 0;   //RF3 como salida
    ANSELFbits.ANSELF3 = 0; //RF3 como digital
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup activado
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}
```

```c
void main(void){
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Kalun Lau Gan",13);
    while(1){
        //aqui va el codigo de la aplicacion
        if(PORTBbits.RB4 == 0){
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Parpadeo rapido",15);
            //parpadeo rapido
            LATFbits.LATF3 = 1;     //enciendo LED
            __delay_ms(100);
            LATFbits.LATF3 = 0;     //apago LED
            __delay_ms(100);
        }
        else{
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Parpadeo lento ",15);
            //parpadeo lento
            LATFbits.LATF3 = 1;     //enciendo LED
            __delay_ms(200);
            LATFbits.LATF3 = 0;     //apago LED
            __delay_ms(200);
        }
    }
}
```

Caracteres personalizados en el LCD 16x2 HD44780

- Podemos incluir caracteres personalizados en nuestras visualizaciones.
- Cada caracter corresponde a una matriz de 8x5 (incluyendo el área del cursor)
- Disponible hasta **ocho** caracteres personalizados.
- Generador online de caracteres personalizados:
  - https://maxpromer.github.io/LCD-Character-Creator/

Procedimiento para caracteres personalizados

1. Diseñar el caracter personalizado y extraer los ocho datos

2. Declarar la constante global con los ocho datos del caracter personalizado

```
const unsigned char corazoncito[]={0x00,0x0A,0x15,0x11,0x11,0x0A,0x04,0x00};
```

```
void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(18);
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    GENERACARACTER(corazoncito, 0);
}
```

3. En la rutina de inicialización del LCD, llamar a la función GENERACARACTER para cargar el caracter personalizado a la memoria CGRAM del LCD

4. Para llamar al caracter personalizado, utilizar la función ENVIA_CHAR()

```
void main(void) {
    configuro();
    LCD_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Hola mundo!", 11);
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Soy UPCINO!", 11);
    POS_CURSOR(2, 13);
    ENVIA_CHAR(0);
```

Caracteres personalizados

Visualización de una figura empleando multiples caracteres personalizados

Conversor analógica a digital (A/D ó ADC)

Revisar Capítulo 40 del datasheet

- Resolución:
- Cantidad de canales analógicos:
- Tiempo de adquisición:
- Rango de voltaje de entrada:
- ¿Cuáles son los valores límites de Vref+ y Vref-?
- Proceso de adquisición de una señal analógica
- ¿Interviene el teorema de muestreo?
- ¿Hay interrupciones?

Conversor A/D

- Resolución: 12bits (ADRESH:ADRESL)
  - Posee un bit ADFM (justificación del resultado)
  - Total 4096 escalones (rango de 0 a 4095)
- Cantidad de canales analógicos: 43
- Se lee un canal analógico a la vez, de manera secuencial
- ¿Interviene el teorema de muestreo? Si.
- Rango máximo de voltaje de entrada por canal: 0-5V?
  (Vref+ = VDD, Vref- = VSS)
- Las señales analógicas no deben bajar de 0V
- Proceso de adquisición de una señal analógica (ver datasheet)
- ¿Hay interrupciones? Si. (ADIE, ADIF)
- Posee módulo computacional

- Computation Features:
  - Averaging and low-pass filter functions
  - Reference comparison
  - 2-level threshold comparison
  - Selectable interrupts

Sensibilidad del A/D del PIC18F57Q43

- Rango de la señal de entrada:
  - Asumiendo que Vref+ = 5V, Vref− = 0V
  - El rango es 0-5V
- Resolución es de 12 bits (4096 escalones) ← 2^{12}
- Altura del escalón: $\frac{5}{4096} = 1.22\,mV$

Ejemplo:

Alternativa de A/D de mayor resolución

Conversor A/D: Diagrama de bloques

Conversor A/D: Justificación del resultado

Figure 40-3. 12-Bit ADC Conversion Result Format

```
unsigned int lectura = 0;    //lectura es una variable de 16 bits
lectura = (ADRESH << 8) + ADRESL;
```

Conversor A/D: Registros

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x03D8	ADCP	7:0	CPON							CPRDY
0x03D9	ADLTH	7:0	LTH[7:0]							
		15:8	LTH[15:8]							
0x03DB	ADUTH	7:0	UTH[7:0]							
		15:8	UTH[15:8]							
0x03DD	ADERR	7:0	ERR[7:0]							
		15:8	ERR[15:8]							
0x03DF	ADSTPT	7:0	STPT[7:0]							
		15:8	STPT[15:8]							
0x03E1	ADFLTR	7:0	FLTR[7:0]							
		15:8	FLTR[15:8]							
0x03E3	ADACC	7:0	ACC[7:0]							
		15:8	ACC[15:8]							
		23:16							ACC[17:16]	
0x03E6	ADCNT	7:0	CNT[7:0]							
0x03E7	ADRPT	7:0	RPT[7:0]							
0x03E8	ADPREV	7:0	PREV[7:0]							
		15:8	PREV[15:8]							
0x03EA	ADRES	7:0	RES[7:0]							
		15:8	RES[15:8]							
0x03EC	ADPCH	7:0			PCH[5:0]					
0x03ED	Reserved									
0x03EE	ADACQ	7:0	ACQ[7:0]							
		15:8				ACQ[12:8]				
0x03F0	ADCAP	7:0				CAP[4:0]				
0x03F1	ADPRE	7:0	PRE[7:0]							
		15:8				PRE[12:8]				
0x03F3	ADCON0	7:0	ON	CONT		CS		FM		GO
0x03F4	ADCON1	7:0	PPOL	IPEN	GPOL					DSEN
0x03F5	ADCON2	7:0	PSIS	CRS[2:0]			ACLR	MD[2:0]		
0x03F6	ADCON3	7:0		CALC[2:0]			SOI	TMD[2:0]		
0x03F7	ADSTAT	7:0	AOV	UTHR	LTHR	MATH		STAT[2:0]		
0x03F8	ADREF	7:0				NREF			PREF[1:0]	
0x03F9	ADACT	7:0			ACT[5:0]					
0x03FA	ADCLK	7:0			CS[5:0]					

Conversor A/D: Registros

Name: ADCON0
Address: 0x3F3

ADC Control Register 0

Bit	7	6	5	4	3	2	1	0
	ON	CONT		CS		FM		GO
Access	R/W	R/W		R/W		R/W		R/W/HC/HS
Reset	0	0		0		0		0

Bit 7 – ON ADC Enable

Value	Description
1	ADC is enabled
0	ADC is disabled

Bit 6 – CONT ADC Continuous Operation Enable

Value	Description
1	GO is retriggered upon completion of each conversion trigger until ADTIF is set (if SOI is set) or until GO is cleared (regardless of the value of SOI)
0	ADC is cleared upon completion of each conversion trigger

Bit 4 – CS ADC Clock Selection

Value	Description
1	Clock supplied from ADCRC dedicated oscillator
0	Clock supplied by F_{OSC}, divided according to ADCLK register

Bit 2 – FM ADC Results Format/Alignment Selection

Value	Description
1	ADRES and ADPREV data are right justified
0	ADRES and ADPREV data are left justified, zero-filled

Bit 0 – GO ADC Conversion Status(1,2)

Value	Description
1	ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the CONT bit
0	ADC conversion completed/not in progress

Conversor A/D: Registros

Name: ADCON1
Address: 0x3F4

ADC Control Register 1

Bit	7	6	5	4	3	2	1	0
	PPOL	IPEN	GPOL					DSEN
Access	R/W	R/W	R/W					R/W
Reset	0	0	0					0

Bit 7 – PPOL Precharge Polarity
Action During 1st Precharge Stage

Value	Condition	Description
x	ADPRE = 0	Bit has no effect
1	ADPRE > 0	External analog I/O pin is connected to V_{DD}. Internal AD sampling capacitor (C_{HOLD}) is connected to V_{SS}.
0	ADPRE > 0	External analog I/O pin is connected to V_{SS}. Internal AD sampling capacitor (C_{HOLD}) is connected to V_{DD}.

Bit 6 – IPEN A/D Inverted Precharge Enable

Value	Condition	Description
x	DSEN = 0	Bit has no effect
1	DSEN = 1	The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle
0	DSEN = 1	Both conversion cycles use the precharge and guards specified by PPOL and GPOL

Bit 5 – GPOL Guard Ring Polarity Selection

Value	Description
1	ADC guard Ring outputs start as digital high during Precharge stage
0	ADC guard Ring outputs start as digital low during Precharge stage

Bit 0 – DSEN Double-Sample Enable

Value	Description
1	Two conversions are processed as a pair. The selected computation is performed after every second conversion.
0	Selected computation is performed after every conversion

Conversor A/D: Registros

Name: ADCON2
Address: 0x3F5

ADC Control Register 2

Bit	7	6	5	4	3	2	1	0
	PSIS	CRS[2:0]			ACLR	MD[2:0]		
Access	R/W	R/W	R/W	R/W	R/W/HC	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – PSIS ADC Previous Sample Input Select

Value	Description
1	ADFLTR is transferred to ADPREV at start-of-conversion
0	ADRES is transferred to ADPREV at start-of-conversion

Bits 6:4 – CRS[2:0] ADC Accumulated Calculation Right Shift Select

Value	Condition	Description
1 to 6	MD = 'b100	Low-pass filter time constant is 2^{CRS}, filter gain is 1:1[2]
1 to 6	MD = 'b011 to 'b001	The accumulated value is right-shifted by CRS (divided by 2^{CRS})[1,2]
x	MD = 'b000	These bits are ignored

Bit 3 – ACLR A/D Accumulator Clear Command[3]

Value	Description
1	The ADACC and ADCNT registers and the AOV bit are cleared
0	Clearing action is complete (or not started)

Bits 2:0 – MD[2:0] ADC Operating Mode Selection[4]

Value	Description
111-101	Reserved
100	Low-Pass Filter mode
011	Burst Average mode
010	Average mode
001	Accumulate mode
000	Basic (Legacy) mode

Notes:

1. To correctly calculate an average, the number of samples (set in ADRPT) must be 2^{CRS}.
2. CRS = 'b111 and 'b000 are reserved.
3. This bit is cleared by hardware when the accumulator operation is complete; depending on oscillator selections, the delay may be many instructions.
4. See the Computation Operation section for full mode descriptions.

Conversor A/D: Registros

Name: ADCON3
Address: 0x3F6

ADC Control Register 3

Bit	7	6	5	4	3	2	1	0
		CALC[2:0]			SOI	TMD[2:0]		
Access		R/W	R/W	R/W	R/W/HC	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 6:4 – CALC[2:0] ADC Error Calculation Mode Select

CALC	ADERR DSEN = 0 Single-Sample Mode	ADERR DSEN = 1 CVD Double-Sample Mode(1)	Application
111	Reserved	Reserved	Reserved
110	Reserved	Reserved	Reserved
101	ADFLTR-ADSTPT	ADFLTR-ADSTPT	Average/filtered value vs. setpoint
100	ADPREV-ADFLTR	ADPREV-ADFLTR	First derivative of filtered value(3) (negative)
011	Reserved	Reserved	Reserved
010	ADRES-ADFLTR	(ADRES-ADPREV)-ADFLTR	Actual result vs. averaged/filtered value
001	ADRES-ADSTPT	(ADRES-ADPREV)-ADSTPT	Actual result vs. setpoint
000	ADRES-ADPREV	ADRES-ADPREV	First derivative of single measurement(2) Actual CVD result(2)

Notes:
1. When DSEN = 1 and PSIS = 0, ADERR is computed only after every second sample.
2. When PSIS = 0.
3. When PSIS = 1.

Bit 3 – SOI ADC Stop-on-Interrupt

Value	Condition	Description
x	CONT = 0	This bit is not used
1	CONT = 1	GO is cleared when the threshold conditions are met, otherwise the conversion is retriggered
0	CONT = 1	GO is not cleared by hardware, must be cleared by software to stop retriggers

Bits 2:0 – TMD[2:0] Threshold Interrupt Mode Select

Value	Description
111	Interrupt regardless of threshold test results
110	Interrupt if ADERR > ADUTH
101	Interrupt if ADERR ≤ ADUTH
100	Interrupt if ADERR < ADLTH or ADERR > ADUTH
011	Interrupt if ADERR > ADLTH and ADERR < ADUTH
010	Interrupt if ADERR ≥ ADLTH
001	Interrupt if ADERR < ADLTH
000	Never interrupt

Conversor A/D: Registros

Name: ADRES
Address: 0x3EA

ADC Result Register

Bit	15	14	13	12	11	10	9	8
	RES[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – RES[15:0] ADC Sample Result

Notes: The individual bytes in this multibyte register can be accessed with the following register names:
- ADRESH: Accesses the high byte ADRES[15: 8]
- ADRESL: Accesses the low byte ADRES[7:0]

Conversor A/D: Registros

Name: ADPCH
Address: 0x3EC

ADC Positive Channel Selection Register

Bit	7	6	5	4	3	2	1	0
			PCH[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:0 – PCH[5:0] ADC Positive Input Channel Selection

PCH	ADC Positive Channel Input
111111	Fixed Voltage Reference (FVR) Buffer 2[1]
111110	Fixed Voltage Reference (FVR) Buffer 1[1]
111101	DAC1 output[2]
111100	Temperature Indicator[3]
111011	V_{SS} (Analog Ground)
111010–110000	Reserved. No channel connected.
101111	RF7/ANF7[5]
101110	RF6/ANF6[5]
101101	RF5/ANF5[5]
101100	RF4/ANF4[5]
101011	RF3/ANF3[5]
101010	RF2/ANF2[5]
101001	RF1/ANF1[5]
101000	RF0/ANF0[5]
100111–100011	Reserved. No channel connected.
100010	RE2/ANE2[4]
100001	RE1/ANE1[4]
100000	RE0/ANE0[4]
011111	RD7/AND7[4]
011110	RD6/AND6[4]
011101	RD5/AND5[4]
011100	RD4/AND4[4]
011011	RD3/AND3[4]
011010	RD2/AND2[4]
011001	RD1/AND1[4]
011000	RD0/AND0[4]
010111	RC7/ANC7
010110	RC6/ANC6
010101	RC5/ANC5
010100	RC4/ANC4
010011	RC3/ANC3
010010	RC2/ANC2
010001	RC1/ANC1
010000	RC0/ANC0
001111	RB7/ANB7

.....continued

PCH	ADC Positive Channel Input
001110	RB6/ANB6
001101	RB5/ANB5
001100	RB4/ANB4
001011	RB3/ANB3
001010	RB2/ANB2
001001	RB1/ANB1
001000	RB0/ANB0
000111	RA7/ANA7
000110	RA6/ANA6
000101	RA5/ANA5
000100	RA4/ANA4
000011	RA3/ANA3
000010	RA2/ANA2
000001	RA1/ANA1
000000	RA0/ANA0

Notes:

1. Refer to the **"Fixed Voltage Reference Module"** chapter for more details.
2. Refer to the **"Digital-to-Analog Converter Module"** chapter for more details.
3. Refer to the **"Temperature Indicator Module"** chapter for more details.
4. 40/44/48-pin devices only.
5. 48-pin devices only.

Conversor A/D: Registros

Name: ADREF
Address: 0x3F8

ADC Reference Selection Register

Bit	7	6	5	4	3	2	1	0
				NREF			PREF[1:0]	
Access				R/W			R/W	R/W
Reset				0			0	0

Bit 4 – NREF ADC Negative Voltage Reference Selection

Value	Description
1	V_{REF}- is connected to external V_{REF}-
0	V_{REF}- is connected to AV_{SS}

Bits 1:0 – PREF[1:0] ADC Positive Voltage Reference Selection

Value	Description
11	V_{REF}+ is connected to internal Fixed Voltage Reference (FVR) module
10	V_{REF}+ is connected to external V_{REF}+
01	Reserved
00	V_{REF}+ is connected to V_{DD}

Conversor A/D: Procedimiento para configurar y adquirir un dato de un canal analógico

Configuración: (Recordar que el rango de voltaje de entrada es de 0V a 5V)

1. Establecer qué puerto será el que reciba la señal analógica
2. Configurar dicho puerto para que sea del tipo entrada y analógico (TRISx.y y ANSELx.y donde “x” es el puerto y “y” el pin)
3. Definir el bit FM: 0 para justificación del resultado a la izquierda, 1 para justificación a la derecha
4. Definir la base de tiempo del ADC con el bit CS
5. Habilitar el funcionamiento del ADC con el bit ADON

Adquisición (toma una muestra):

1. Seleccionar el canal de lectura con el registro ADPCH
2. Iniciar la conversión con el bit GO=1
3. Esperar a que el bit GO baje a cero (término de la conversión)
4. El resultado estará en el par de registros ADRESH:ADRESL

Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal

Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal

- La función *convierte()* se encarga de individualizar los dígitos de una variable *int*. (solo entre 0 y 9999)
- Tener en cuenta que las variables *millar*, *centena*, *decena* y *unidad* deben de estar declaradas como **variables globales**.

```c
void convierte(unsigned int valor){
    millar = valor / 1000;
    centena = (valor % 1000) / 100;
    decena = (valor % 100) / 10;
    unidad = valor % 10;
}
```

Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

unsigned int resultado = 0;     //resultado del ADC en 12bits
unsigned int millar, centena, decena, unidad;

void configuro(void){
    //configuracion del oscilador
    OSCCON1 = 0x60;             //HFINTOSC, POSTS 1:1
    OSCFRQ = 0x07;              //HFINTOSC a 48MHz
    OSCEN = 0x40;               //HFINTOSC enabled
    //configuracion de E/S
    TRISBbits.TRISB4 = 1;       //RB4 como entrada
    ANSELBbits.ANSELB4 = 0;     //RB4 como digital
    WPUBbits.WPUB4 = 1;         //RB4 con pullup activado
    TRISFbits.TRISF3 = 0;       //RF3 como salida
    ANSELFbits.ANSELF3 = 0;     //RF3 como digital
    //configuracion del ADC
    ADCON0bits.FM = 1;          //right justify
    ADCON0bits.CS = 1;          //ADCRC Clock
    ADPCH = 0x00;               //RA0 is Analog channel
    TRISAbits.TRISA0 = 1;       //Set RA0 to input
    ANSELAbits.ANSELA0 = 1;     //Set RA0 to analog
    ADCON0bits.ON = 1;          //Turn ADC On
}

void lcd_init(void){
    TRISD = 0x00;       //RD salidas
    ANSELD = 0x00;      //RD digitales
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}
```

```c
void convierte(unsigned int valor){
    millar = valor / 1000;
    centena = (valor % 1000) / 100;
    decena = (valor % 100) / 10;
    unidad = valor % 10;
}

void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,2);
    ESCRIBE_MENSAJE("Ejerciciomon", 12);
    while(1){
        ADCON0bits.GO = 1; //Start conversion
        while(ADCON0bits.GO); //Wait for conversion done
        resultado = (ADRESH << 8) + ADRESL;
        convierte(resultado);
        POS_CURSOR(2,2);
        ESCRIBE_MENSAJE("ADC CH0:", 8);
        ENVIA_CHAR(millar+0x30);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
    }
}
```

Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal

Asignación:

- Leer dos canales analógicos y mostrarlos en el LCD

Implementación 2024-1

Implementación:

- RA0 entrada analógica y conectado a un potenciómetro en configuración divisor de tensión
- LCD 16x2 HD44780 conectado en RD para usar la librería LCD Salas_Lau

Implementación 2024-1

Prueba inicial:

- Frecuencia de trabajo 48MHz con HFINTOSC
- Titilar el LED integrado ubicado en RF3 con un periodo de 400ms

```c
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>

#define _XTAL_FREQ 48000000UL

void configuro(){
    OSCCON1 = 0x60;          //NOSC=HFINTOSC NDIV1:1
    OSCFRQ = 0x07;           //HFINTOSC 48MHz
    OSCEN = 0x40;            //HFINTOSC enabled
    TRISFbits.TRISF3 = 0;    //RF3 output
    ANSELFbits.ANSELF3 = 0;  //RF3 digital
}

void main(void) {
    configuro();             //calling configuro function
    while(1){                //do it forever
        LATFbits.LATF3 = 0;  //LED on
        __delay_ms(200);     //delay of 200ms
        LATFbits.LATF3 = 1;  //LED off
        __delay_ms(200);     //delay of 200ms
    }
}
```

Implementación 2024-1

Evidencia de titileo de LED en RF3

Implementación 2024-1

Modificación del ejemplo inicial:

- Se agregó el pulsador integrado en RB4 para controlar el titileo del LED en RF3.
- Solo titilará el LED si se mantiene presioando el pulsador

```c
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#define _XTAL_FREQ 48000000UL

void configuro(void){
    OSCCON1 = 0x60;          //NOSC=HFINTOSC NDIV1:1
    OSCFRQ = 0x07;           //HFINTOSC 48MHz
    OSCEN = 0x40;            //HFINTOSC enabled
    TRISFbits.TRISF3 = 0;    //RF3 output
    ANSELFbits.ANSELF3 = 0;  //RF3 digital
    TRISBbits.TRISB4 = 1;    //RB4 input
    ANSELBbits.ANSELB4 = 0;  //RB4 digital
    WPUBbits.WPUB4 = 1;      //RB4 weak pullup enabled
}

void main(void) {
    configuro();             //calling configuro function
    while(1){                //do it forever
        if(PORTBbits.RB4 == 0){
            LATFbits.LATF3 = 0; //LED on
            __delay_ms(100);    //delay of 100ms
            LATFbits.LATF3 = 1; //LED off
            __delay_ms(100);    //delay of 100ms
        }
        else{
            LATFbits.LATF3 = 0; //LED on
            __delay_ms(200);    //delay of 200ms
            LATFbits.LATF3 = 1; //LED off
            __delay_ms(200);    //delay of 200ms
        }
    }
}
```

Implementación 2024-1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 48000000UL

void configuro(void){
    OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x07;       //HFINTOSC 48MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    TRISFbits.TRISF3 = 0;    //RF3 salida
    ANSELFbits.ANSELF3 = 0; //RF3 digital
    TRISBbits.TRISB4 = 1;    //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;      //RB4 pullup enabled
}
```

```c
void main(void) {
    configuro();
    while(1){
        if(PORTBbits.RB4 == 0){
            LATFbits.LATF3 = 0; //LED on
            __delay_ms(100);
            LATFbits.LATF3 = 1; //LEd off
            __delay_ms(100);
        }
        else{
            LATFbits.LATF3 = 0; //LED on
            __delay_ms(300);
            LATFbits.LATF3 = 1; //LEd off
            __delay_ms(300);
        }
    }
}
```

Modificación del ejemplo inicial:

- Se agregó el pulsador integrado en RB4 para cambiar la velocidad del titileo del LED en RF3.
- Al mantener presionado el pulsador aumentará la velocidad del titileo.

Implementación 2024-1

```c
1  #include <xc.h>
2  #include "cabecera.h"
3  #define _XTAL_FREQ 48000000UL
4
5  unsigned char indicador = 0;    //variable global
6
7  void configuro(void){
8      OSCCON1 = 0x60;
9      OSCFRQ = 0x07;
10     OSCEN = 0x40;
11     TRISFbits.TRISF3 = 0;       //RF3 como salida
12     ANSELFbits.ANSELF3 = 0;     //RF3 como digital
13     TRISBbits.TRISB4 = 1;       //RB4 como entrada
14     ANSELBbits.ANSELB4 = 0;     //RB4 como digital
15     WPUBbits.WPUB4 = 1;         //RB4 con weakpullup
16     PIE1bits.INT0IE = 1;        //INT0 habilitada
17     INTCON0bits.GIE = 1;        //global habilitado
18     INTCON0bits.INT0EDG = 0;    //detecccion falling edge
19     INT0PPS = 0x0C;             //ruteando INT0 para RB4
20 }
```

```c
22 void main(void) {
23     configuro();
24     while(1){
25         if(indicador == 0){
26             LATFbits.LATF3 = 0;     //enciendo LED
27             __delay_ms(300);        //retardo de 300ms
28             LATFbits.LATF3 = 1;     //apago el LED
29             __delay_ms(300);        //retardo de 300ms
30         }
31         else if(indicador == 1){
32             LATFbits.LATF3 = 1;     //apago el LED
33         }
34     }
35 }
```

```c
37 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
38     PIR1bits.INT0IF = 0;        //bajamos bandera INT0IF
39     if (indicador == 0){
40         indicador = 1;
41     }
42     else{
43         indicador = 0;
44     }
45 }
46
47 void __interrupt(irq(default)) default_ISR(void){
48
49 }
```

Empleando interrupciones vectorizadas:

- Se esta empleando el pulsador integrado para la interrupción externa INT0, para ello que debe de rutear dicha interrupción hacia RB4 empleando el PPS (línea 19).
- Se estableció una FEM de dos estados en donde la acción de cambió es el evento de INT0, la variable indicador es el que determinará si parapadea el LED o se mantiene apagado.

Implementación 2024-1

```c
1  #include <xc.h>
2  #include "cabecera.h"
3  #define _XTAL_FREQ 48000000UL /*para que el compilador sepa
4                                 la frq que esta trabajando el
5
6  unsigned char estado_LED = 0;
7
8  void configuro(void){
9      OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
10     OSCFRQ = 0x07;       //HFINTOSC 48MHz
11     OSCEN = 0x40;        //HFINTOSC enabled
12     TRISBbits.TRISB4 = 1;    //RB4 entrada
13     ANSELBbits.ANSELB4 = 0; //RB4 digital
14     WPUBbits.WPUB4 = 1;      //RB4 pullup enabled
15     TRISFbits.TRISF3 = 0;    //RF3 salida
16     ANSELFbits.ANSELF3 = 0; //RF3 digital
17     INT0PPS = 0x0C;          //ruteando INT0 a RB4
18     INTCON0bits.INT0EDG = 0;     //INT0 falling edge
19     PIE1bits.INT0IE = 1;     //habilita INT0
20     INTCON0bits.GIE = 1;     //int global enabled
21 }
```

```c
23 void main(void) {
24     configuro();
25     while(1){
26         switch(estado_LED){
27             case 0:
28                 LATFbits.LATF3 = 1; //LED off
29                 break;
30             case 1:
31                 LATFbits.LATF3 = 0; //LED on
32                 __delay_ms(400);    //retardo de 400ms
33                 LATFbits.LATF3 = 1; //LED off
34                 __delay_ms(400);    //retardo de 400ms
35                 break;
36             case 2:
37                 LATFbits.LATF3 = 0; //LED on
38                 __delay_ms(100);    //retardo de 100ms
39                 LATFbits.LATF3 = 1; //LED off
40                 __delay_ms(100);    //retardo de 100ms
41                 break;
42         }
43     }
44 }
```

```c
46 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
47     PIR1bits.INT0IF = 0;         //bajamos bandera INT0IF
48     if(estado_LED == 2){
49         estado_LED = 0;
50     }
51     else{
52         estado_LED++;
53     }
54 }
55
56 void __interrupt(irq(default)) default_ISR(void){
57 }
```

Empleando interrupciones vectorizadas:

- Se esta empleando el pulsador integrado para la interrupción externa INT0, para ello que debe de rutear dicha interrupción hacia RB4 empleando el PPS (línea 19).
- Se estableció una FEM de dos estados en donde la acción de cambió es el evento de INT0, la variable estado_LED es el que determinará la velocidad del titileo del LED.

Implementación 2024-1

```c
1  #include <xc.h>
2  #include "cabecera.h"
3  #define _XTAL_FREQ 48000000UL /*para que el compilador sepa
4                                 la frq que esta trabajando el M
5  #define apagado 0
6  #define parpadeando 1
7
8  unsigned char estado_LED = apagado;
9
10 void configuro(void){
11     OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
12     OSCFRQ = 0x07;       //HFINTOSC 48MHz
13     OSCEN = 0x40;        //HFINTOSC enabled
14     TRISBbits.TRISB4 = 1;    //RB4 entrada
15     ANSELBbits.ANSELB4 = 0; //RB4 digital
16     WPUBbits.WPUB4 = 1;      //RB4 pullup enabled
17     TRISFbits.TRISF3 = 0;    //RF3 salida
18     ANSELFbits.ANSELF3 = 0; //RF3 digital
19     INT0PPS = 0x0C;          //ruteando INT0 a RB4
20     INTCON0bits.INT0EDG = 0;     //INT0 falling edge
21     PIE1bits.INT0IE = 1;     //habilita INT0
22     INTCON0bits.GIE = 1;     //int global enabled
23 }
```

```c
25 void main(void) {
26     configuro();
27     while(1){
28         if(estado_LED == apagado){  //pregunto estado
29             LATFbits.LATF3 = 1; //LED off
30         }
31         else if(estado_LED == parpadeando){
32             LATFbits.LATF3 = 0; //LED on
33             __delay_ms(400);    //retardo de 400ms
34             LATFbits.LATF3 = 1; //LED off
35             __delay_ms(400);    //retardo de 400ms
36         }
37         else{
38         }
39     }
40 }
```

```c
42 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
43     PIR1bits.INT0IF = 0;         //bajamos bandera INT0IF
44     if(estado_LED == apagado){
45         estado_LED = parpadeando;
46     }
47     else{
48         estado_LED = apagado;
49     }
50 }
51
52 void __interrupt(irq(default)) default_ISR(void){
53 }
```

En el lenguaje C se puede emplear etiquetas para que pueda identificarse mejor los parámetros.

- En el presente ejemplo se realizó dos "define": apagado que simboliza 0 y parpadeando que simboliza 1.

Implementación 2024-1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 48000000UL

unsigned char indicador = 0;
unsigned char velocidad = 0;

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x07;
    OSCEN = 0x40;
    TRISFbits.TRISF3 = 0;
    ANSELFbits.ANSELF3 = 0;
    TRISBbits.TRISB4 = 1;
    ANSELBbits.ANSELB4 = 0;
    WPUBbits.WPUB4 = 1;

    TRISBbits.TRISB3 = 1;
    ANSELBbits.ANSELB3 = 0;
    WPUBbits.WPUB3 = 1;

    PIE1bits.INT0IE = 1;
    PIE6bits.INT1IE = 1;
    INTCON0bits.GIE = 1;
    INTCON0bits.INT0EDG = 0;
    INTCON0bits.INT1EDG = 0;
    INT0PPS = 0x0C;
    INT1PPS = 0x0B;
}
```

```c
void main(void) {
    configuro();
    while(1){
        if(indicador == 0){
            if(velocidad == 0){
                LATFbits.LATF3 = 0;     //enciend
                __delay_ms(300);        //retardo
                LATFbits.LATF3 = 1;     //apago e
                __delay_ms(300);        //retardo
            }
            else{
                LATFbits.LATF3 = 0;     //enciend
                __delay_ms(100);        //retardo
                LATFbits.LATF3 = 1;     //apago e
                __delay_ms(100);        //retardo
            }
        }
        else if(indicador == 1){
            LATFbits.LATF3 = 1;     //apago el LE
        }
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;        //bajamos bandera
    if (indicador == 0){
        indicador = 1;
    }
    else{
        indicador = 0;
    }
}
```

```c
void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajo bandera INT1IF
    if (velocidad == 0){
        velocidad = 1;
    }
    else{
        velocidad = 0;
    }
}

void __interrupt(irq(default)) default_ISR(void){

}
```

Empleando interrupciones vectorizadas:

- Ahora se tiene dos fuentes de interrupción: INT0 e INT1
- Tener en cuenta que según el INT1PPS, el puerto de INT1 se ruteó a RB3.
- La función de INT0 permite establecer si va a parpadear el LED o si se mantiene apagado.
- La función de INT1 permite cambiar de velocidad de parpadeo cambiar el periodo de retardo (rápido/lento)

Implementación 2024-1

```c
#include <stdio.h>
#include <stdlib.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

void configuro(void){
    OSCCON1 = 0x60;     //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x07;      //HFINTOSC=48MHz
    OSCEN = 0x40;       //HFINTOSC enabled
    ANSELFbits.ANSELF3 = 0; //RF3 digital
    TRISFbits.TRISF3 = 0;   //RF3 output
}

void main(void) {
    configuro();
    LCD_INIT();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE2("Hola mundo!");
    while(1){
        LATFbits.LATF3 = 0; //LED on
        __delay_ms(300);
        LATFbits.LATF3 = 1; //LED off
        __delay_ms(300);
    }
}
```

Segunda prueba: Visualizar "Hola mundo!" en el LCD:

- Se agregó la librería LCD al proyecto del primer ejemplo.
- Se muestra el mensaje y a la vez el LED integrado se encuentra parpadeando
- Tener en cuenta que la función ESCRIBE_MENSAJE2() esta disponible en la nueva versión de la librería LCD publicada en el repositorio.

Implementación 2024-1

Ejemplo de visualización de mensajes en ambas líneas del LCD:

- Para pasar de una línea a otra se emplea la función POS_CURSOR(x,y) que mueve el cursor a la posición x,y especificada

Implementación 2024-1

En este ejemplo se muestra el estado del pulsador integrado en RB4 en la segunda línea del LCD

```c
#include <stdio.h>
#include <stdlib.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

void configuro(void){
    OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x07;       //HFINTOSC=48MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    ANSELFbits.ANSELF3 = 0; //RF3 digital
    TRISFbits.TRISF3 = 0;   //RF3 output
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    TRISBbits.TRISB4 = 1;   //RB4 input
    WPUBbits.WPUB4 = 1;     //RB4 weak pullup enabled
}

void main(void) {
    configuro();
    LCD_INIT();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE2("Hola mundo!");
    while(1){
        if(PORTBbits.RB4 == 0){
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE2("BTN: pulsado    ");
        }
        else{
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE2("BTN: no pulsado!");
        }
    }
}
```

Implementación de “splash screen”

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;  //HFINTOSC a 32MHz
    OSCEN = 0x40;
    //configuracion del A/D
    TRISAbits.TRISA0 = 1;   //RA0 entrada
    ANSELAbits.ANSELA0 = 1; //RA0 analogica
    ADCON0 = 0x94;  //ADC conv manual, ADCRC, just der, ADC on
}

void LCD_inicia(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(21);
    LCD_CONFIG();
    __delay_ms(22);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    __delay_ms(100);
}
```

```c
void main(void) {
    configuro();
    LCD_inicia();
    POS_CURSOR(1,1);
    ESCRIBE_MENSAJE("Bienvenido UPC",14);
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("INGENIERIA  2024",16);
    __delay_ms(5000);
    BORRAR_LCD();
    while(1){

    }
}
```

EL54 2024_1 Sem10

Logo de UPC empleando seis caracteres personalizados

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

const unsigned char logo_upc_1[]={0x01,0x03,0x06,0x04,0x04,0x0C,0x18,0x18};
const unsigned char logo_upc_2[]={0x11,0x04,0x06,0x0E,0x1C,0x18,0x08,0x06};
const unsigned char logo_upc_3[]={0x10,0x18,0x0C,0x04,0x04,0x06,0x03,0x03};
const unsigned char logo_upc_4[]={0x18,0x1C,0x0C,0x0C,0x07,0x07,0x01,0x00};
const unsigned char logo_upc_5[]={0x03,0x03,0x03,0x06,0x04,0x1F,0x1F,0x0E};
const unsigned char logo_upc_6[]={0x03,0x17,0x06,0x06,0x1C,0x1C,0x10,0x00};

void configuro(void){
    OSCCON1 = 0x60;     //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x06;      //HFINTOSC a 32MHz
    OSCEN = 0x40;       //HFINTOSC enabled
    TRISAbits.TRISA0 = 1;   //RA0 entrada
    ANSELAbits.ANSELA0 = 1; //RA0 analogica
}
```

P	Q	
0x01	0x11	0x10
0x03	0x04	0x18
0x06	0x06	0x0C
0x04	0x0E	0x04
0x04	0x1C	0x04
0x0C	0x18	0x06
0x18	0x08	0x03
0x18	0x06	0x03
0x18	0x03	0x03
0x1C	0x03	0x17
0x0C	0x03	0x06
0x0C	0x06	0x06
0x07	0x04	0x1C
0x07	0x1F	0x1C
0x01	0x1F	0x10
0x00	0x0E	0x00

```c
void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(19);
    LCD_CONFIG();
    __delay_ms(21);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    __delay_ms(100);
    GENERACARACTER(logo_upc_1, 0);
    GENERACARACTER(logo_upc_2, 1);
    GENERACARACTER(logo_upc_3, 2);
    GENERACARACTER(logo_upc_4, 3);
    GENERACARACTER(logo_upc_5, 4);
    GENERACARACTER(logo_upc_6, 5);
}
```

```c
void main(void) {
    configuro();
    LCD_init();
    POS_CURSOR(1,6);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(2);
    POS_CURSOR(2,6);
    ENVIA_CHAR(3);
    ENVIA_CHAR(4);
    ENVIA_CHAR(5);
    __delay_ms(5000);
    BORRAR_LCD();
    while(1){

    }
}
```

Implementación ADC 2024-1

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;      //HFINTOSC a 32MHz
    OSCEN = 0x40;
    //configuracion del ADC
    TRISAbits.TRISA0 = 1;   //RA0 entrada
    ANSELAbits.ANSELA0 = 1; //RA0 analogica
    ADCON0 = 0x94;      //ADC on, just. derecha,
}

void leer_ADC(void){
    ADPCH = 0x00;       //ANA0 seleccionado
    ADCON0bits.GO = 1;  //iniciamos la toma de u
    while(ADCON0bits.GO == 1);
    //el resultado esta en ADRESH:ADRESL
}
```

```c
void main(void) {
    unsigned int valor_leido = 0;
    configuro();
    LCD_INIT();
    __delay_ms(500);
    POS_CURSOR(1,1);
    ESCRIBE_MENSAJE("Ejemplo Sem.10",14);
    POS_CURSOR(2,1);
    ENVIA_CHAR(0xE4);
    ESCRIBE_MENSAJE("C PIC18F57Q43",13);
    __delay_ms(3000);
    BORRAR_LCD();
    while(1){
        leer_ADC();
        valor_leido = (ADRESH<<8) + ADRESL;
        POS_CURSOR(1,0);
        ESCRIBE_MENSAJE("Lectura AN-0:",13);
        POS_CURSOR(2,5);
        ENVIA_CHAR((valor_leido / 10000) + 0x30);   //mostrando dig diez millar
        ENVIA_CHAR(((valor_leido % 10000) / 1000) + 0x30); //muestra dig millar
        ENVIA_CHAR(((valor_leido % 1000) / 100) + 0x30); //muestra dig centena
        ENVIA_CHAR(((valor_leido % 100) / 10) + 0x30);  //muestra dig decena
        ENVIA_CHAR((valor_leido % 10) + 0x30);  //muestra dig unidad
    }
}
```

Primeras pruebas de uso del ADC para obtener una muestra de una señal analógica ingresada por RA0.
La visualización en el LCD corresponde al resultado del ADC en formato de escalones en 12 bits

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

const unsigned char logo_upc_1[]={0x01,0x03,0x06,0x04,0x04,0x0C,0x18,0x18};
const unsigned char logo_upc_2[]={0x11,0x04,0x06,0x0E,0x1C,0x18,0x08,0x06};
const unsigned char logo_upc_3[]={0x10,0x18,0x0C,0x04,0x04,0x06,0x03,0x03};
const unsigned char logo_upc_4[]={0x18,0x1C,0x0C,0x0C,0x07,0x07,0x01,0x00};
const unsigned char logo_upc_5[]={0x03,0x03,0x03,0x06,0x04,0x1F,0x1F,0x0E};
const unsigned char logo_upc_6[]={0x03,0x17,0x06,0x06,0x1C,0x1C,0x10,0x00};

void configuro(void){
    OSCCON1 = 0x60;     //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x06;      //HFINTOSC a 32MHz
    OSCEN = 0x40;       //HFINTOSC enabled
    //configuracion del A/D
    TRISAbits.TRISA0 = 1;   //RA0 entrada
    ANSELAbits.ANSELA0 = 1; //RA0 analogica
    ADCON0 = 0x94;      //ADON, ADCRC, RIGHT JUST, NO
}

unsigned int lectura_ADC_0(void){
    ADPCH = 0x00;       //Canal RA0/ANA0 seleccionado
    ADCON0bits.GO = 1;  //inicio de conversion
    while(ADCON0bits.GO == 1);  //esperar a que termi
    return((ADRESH<<8)+ADRESL);
}
```

```c
void main(void) {
    unsigned int resultado;
    configuro();
    LCD_init();
    POS_CURSOR(1,6);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(2);
    POS_CURSOR(2,6);
    ENVIA_CHAR(3);
    ENVIA_CHAR(4);
    ENVIA_CHAR(5);
    __delay_ms(5000);
    BORRAR_LCD();
    while(1){
        POS_CURSOR(1,0);
        ESCRIBE_MENSAJE("Lectura RA0:",12);
        resultado = lectura_ADC_0();
        POS_CURSOR(2,5);
        ENVIA_CHAR((resultado / 10000) + 0x30);   //vis diez millar
        ENVIA_CHAR(((resultado % 10000) / 1000) + 0x30);  //viz millar
        ENVIA_CHAR(((resultado % 1000) / 100) + 0x30);  //viz centena
        ENVIA_CHAR(((resultado % 100) / 10) + 0x30);    //viz decena
        ENVIA_CHAR((resultado % 10) + 0x30);    //viz unidad
    }
}
```

```c
void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(19);
    LCD_CONFIG();
    __delay_ms(21);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    __delay_ms(100);
    GENERACARACTER(logo_upc_1, 0);
    GENERACARACTER(logo_upc_2, 1);
    GENERACARACTER(logo_upc_3, 2);
    GENERACARACTER(logo_upc_4, 3);
    GENERACARACTER(logo_upc_5, 4);
    GENERACARACTER(logo_upc_6, 5);
}
```

Integración del splash screen con el logo de upc basado en sesis caracteres personalizados y el uso del ADC para tomar una muestra del canal RA0/ANA0, finalmente la visualización en steps en el LCD.

EL54 2024_1
Sem10

Implementación ADC 2024-1

EL54 2024_1 Sem10

Modificación del main() para que se visualice el valor en voltaje

```c
void main(void) {
    unsigned int resultado;
    float calculo;
    unsigned int voltaje;
    configuro();
    LCD_init();
    POS_CURSOR(1,6);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(2);
    POS_CURSOR(2,6);
    ENVIA_CHAR(3);
    ENVIA_CHAR(4);
    ENVIA_CHAR(5);
    __delay_ms(5000);
    BORRAR_LCD();
```

```c
    while(1){
        POS_CURSOR(1,0);
        ESCRIBE_MENSAJE("Lectura RA0:",12);
        resultado = lectura_ADC_0();
        POS_CURSOR(2,0);
        ENVIA_CHAR((resultado / 10000) + 0x30);   //vis diez millar
        ENVIA_CHAR(((resultado % 10000) / 1000) + 0x30);  //viz millar
        ENVIA_CHAR(((resultado % 1000) / 100) + 0x30);  //viz centena
        ENVIA_CHAR(((resultado % 100) / 10) + 0x30);    //viz decena
        ENVIA_CHAR((resultado % 10) + 0x30);    //viz unidad
        calculo = resultado * 0.122;
        voltaje = calculo;          //convierte float a integer
        POS_CURSOR(2,8);
        ENVIA_CHAR(((voltaje % 1000) / 100) + 0x30);  //viz centena
        ENVIA_CHAR('.');
        ENVIA_CHAR(((voltaje % 100) / 10) + 0x30);    //viz decena
        ENVIA_CHAR((voltaje % 10) + 0x30);    //viz unidad
        ENVIA_CHAR('V');
    }
}
```

Implementación 2024-1

Asignación Semana 10 2024-1

- Modificar el ejemplo mostrado en clase para que se visualice CUATRO canales analógicos en el LCD en unidades de voltaje.

- Carpeta compartida: https://bit.ly/3KeDi3L
- Formato de nombre de archivo de video de evidencia: EL256_EL51_[tu primer apellido]_[tu primer nombre]_Sem10.mp4
  - Ejemplo de nombre de archivo de video de evidencia: EL256_EL51_Perez_Juan_Sem10.mp4
- Video grabarlo en calidad 720p como máximo y 30 segundos de duración como máximo.

Asignación Semana 10 2024-1

- Modificar el ejemplo mostrado en clase para que adicionalmente se adquieran tres señales analógicas (RA0, RB4 y RC6) y se visualicen las siguientes pantallas:

```
Channel RA0:
Voltage: 3.87V
```

```
Channel RB4:
Voltage: 1.85V
```

```
Channel RC6:
Voltage: 4.32V
```

- El intercambio de pantallas será al pulsar un botón activo en alto en INT0 para que se visualice la pantalla de RA0,al pulsar el botón activo en alto en INT1 para que se visualice la pantalla de RB4 y al pulsar el botón actvo en alto en INT2 para visualizar la pantalla de RC6
- Carpeta compartida: https://bit.ly/3WRiXsV
- Formato de nombre de archivo de video de evidencia: EL256_EL51_[tu primer apellido]_[tu primer nombre]_Sem10.mp4
  - Ejemplo de nombre de archivo de video de evidencia: EL256_EL51_Perez_Juan_Sem10.mp4
- Video grabarlo en calidad 720p como máximo y 30 segundos de duración como máximo.

Asignación Semana 10 2024-1

- Hardware

Solución

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned char pantalla = 0;

void configuro(void){
    OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config de las I/O
    TRISAbits.TRISA0 = 1;   //RA0 entrada
    ANSELAbits.ANSELA0 = 1; //RA0 analogica
    TRISB = 0xFF;        //RB todas entradas
    ANSELB = 0x18;       //RB4 analogica, RB2 RB1 RB0 digitale
    TRISCbits.TRISC6 = 1;   //RC6 entrada
    ANSELCbits.ANSELC6 = 1; //RC6 analogica
    //configuracion de AD
    ADCON0 = 0x94;
    //config de las ints
    PIE1bits.INT0IE = 1;
    PIE6bits.INT1IE = 1;
    PIE10bits.INT2IE = 1;
    INTCON0bits.GIE = 1;
}
```

```c
unsigned int tomamuestra_ADC(unsigned char canal){
    ADPCH = canal;   //canal RA0
    ADCON0bits.GO = 1;  //inicio conversion
    while(ADCON0bits.GO == 1);  //espero a que termine de co
    return((ADRESH << 8) + ADRESL);
}
```

```c
void main(void) {
    unsigned int lectura;
    float calculo;
    unsigned int voltaje;
    configuro();
    LCD_inicializa();
    splash_screen();
    while(1){
        switch(pantalla){
            case 0:
                POS_CURSOR(1,0);
                ESCRIBE_MENSAJE("Channel RA0:",12);
                POS_CURSOR(2,0);
                ESCRIBE_MENSAJE("Voltage: ", 9);
                lectura = tomamuestra_ADC(0x00);    //toma muestra de RA0
                calculo = lectura * 0.122;
                voltaje = calculo;
                ENVIA_CHAR(((voltaje % 1000) / 100) + 0x30);    //muestra centena
                ENVIA_CHAR('.');
                ENVIA_CHAR(((lectura % 100) / 10) + 0x30);      //muestra decena
                ENVIA_CHAR((lectura % 10) + 0x30);      //muestra unidad
                ENVIA_CHAR('V');
                break;
            case 1:
                POS_CURSOR(1,0);
                ESCRIBE_MENSAJE("Channel RB4:",12);
                POS_CURSOR(2,0);
                ESCRIBE_MENSAJE("Voltage: ", 0x0C);
                lectura = tomamuestra_ADC(0x00);    //toma m
                calculo = lectura * 0.122;
                voltaje = calculo;
                ENVIA_CHAR(((voltaje % 1000) / 100) + 0x30);
                ENVIA_CHAR('.');
                ENVIA_CHAR(((lectura % 100) / 10) + 0x30);
                ENVIA_CHAR((lectura % 10) + 0x30);      //mu
                ENVIA_CHAR('V');
                break;
            case 2:
                POS_CURSOR(1,0);
                ESCRIBE_MENSAJE("Channel RC6:",12);
                POS_CURSOR(2,0);
                ESCRIBE_MENSAJE("Voltage: ", 9);
                lectura = tomamuestra_ADC(0x16);    //toma muestra de RC6
                calculo = lectura * 0.122;
                voltaje = calculo;
                ENVIA_CHAR(((voltaje % 1000) / 100) + 0x30);    //muestra centen
                ENVIA_CHAR('.');
                ENVIA_CHAR(((lectura % 100) / 10) + 0x30);      //muestra decena
                ENVIA_CHAR((lectura % 10) + 0x30);      //muestra unidad
                ENVIA_CHAR('V');
                break;
        }
    }
}
```

```c
void LCD_inicializa(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(31);
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    __delay_ms(100);
}

void splash_screen(void){
    unsigned char splash_1[] = {"HOLA UPCINO!!!"};
    unsigned char var_x;
    POS_CURSOR(1,1);
    for(var_x=0;var_x<14;var_x++){
        ENVIA_CHAR(splash_1[var_x]);
        __delay_ms(100);
    }
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("loading",7);
    for(var_x=0;var_x<7;var_x++){
        ENVIA_CHAR('.');
        __delay_ms(100);
    }
    ESCRIBE_MENSAJE("ok",2);
    __delay_ms(5000);
    BORRAR_LCD();
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;
    pantalla = 0;
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;
    pantalla = 1;
}

void __interrupt(irq(IRQ_INT2)) INT2_ISR(void){
    PIR10bits.INT2IF = 0;
    pantalla = 2;
}
```

Implementación 2024-1

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"

#define _XTAL_FREQ 48000000UL /*para que el compilador sepa
                                la frq que esta trabajando el MCU*/

unsigned char estado_LED = 0;
unsigned char corazon[]={0x00,0x0A,0x15,0x11,0x11,0x0A,0x04,0x00};

void configuro(void){
    OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x07;       //HFINTOSC 48MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    TRISBbits.TRISB4 = 1;   //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enabled
    TRISFbits.TRISF3 = 0;   //RF3 salida
    ANSELFbits.ANSELF3 = 0; //RF3 digital
    INT0PPS = 0x0C;         //ruteando INT0 a RB4
    INTCON0bits.INT0EDG = 0;    //INT0 falling edge
    PIE1bits.INT0IE = 1;    //habilita INT0
    INTCON0bits.GIE = 1;    //int global enabled
}
```

```c
void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(21);
    LCD_CONFIG();
    __delay_ms(23);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    GENERACARACTER(corazon,0);
}

void main(void) {
    configuro();
    LCD_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Hola mundo!", 11);
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Soy UPCINO ", 11);
    ENVIA_CHAR(0);

    while(1){

    }
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;        //bajamos bandera INT0IF
}

void __interrupt(irq(default)) default_ISR(void){
}
```

Implementación 2024-1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 48000000UL

#define LED_integrado LATFbits.LATF3
#define LED_apagado 0
#define LED_parpadeo 1

unsigned char LED_estado = LED_apagado;

void configuro(void){
    OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1
    OSCFRQ = 0x07;       //HFINTOSC 48MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    TRISFbits.TRISF3 = 0;   //RF3 salida
    ANSELFbits.ANSELF3 = 0; //RF3 digital
    TRISBbits.TRISB4 = 1;   //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enabl
    INT0PPS = 0x0C;         //redirigir señal
    INTCON0bits.INT0EDG = 0;    //INT0 detecci
    PIE1bits.INT0IE = 1;        //INT0 habilitada
    INTCON0bits.GIE = 1;        //switch global de ints h
}
```

```c
void main(void) {
    configuro();
    while(1){
        if(LED_estado == LED_parpadeo){     //pregunto si presione BTN
            LED_integrado = 0; //LED on
            __delay_ms(400);    //retardo de 400ms
            LED_integrado = 1; //LED off
            __delay_ms(400);    //retardo de 400ms
        }
        else if(LED_estado == LED_apagado){
            LED_integrado = 1; //LED off
        }
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;        //bajamos la bandera INT0IF
    if(LED_estado == LED_apagado){
        LED_estado = LED_parpadeo;
    }
    else{
        LED_estado = LED_apagado;
    }
}
```

```c
void __interrupt(irq(default)) default_ISR(void){

}
```

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

#define LED_integrado LATFbits.LATF3
#define LED_apagado 0
#define LED_parpadeo 1

unsigned char LED_estado = LED_apagado;

void configuro(void){
    OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1
    OSCFRQ = 0x07;       //HFINTOSC 48MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    TRISFbits.TRISF3 = 0;   //RF3 salida
    ANSELFbits.ANSELF3 = 0; //RF3 digital
    TRISBbits.TRISB4 = 1;   //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enable
    INT0PPS = 0x0C;         //redirigir señal
    INTCON0bits.INT0EDG = 0;    //INT0 detecci
    PIE1bits.INT0IE = 1;        //INT0 habilita
    INTCON0bits.GIE = 1;        //switch global
}
```

```c
void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(18);
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void main(void) {
    configuro();
    LCD_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Hola mundo!", 11);
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Soy UPCINO!", 11);
    while(1){
        if(LED_estado == LED_parpadeo){     //pregunto si presione BTN
            LED_integrado = 0; //LED on
            __delay_ms(400);    //retardo de 400ms
            LED_integrado = 1; //LED off
            __delay_ms(400);    //retardo de 400ms
        }
        else if(LED_estado == LED_apagado){
            LED_integrado = 1; //LED off
        }
    }
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;        //bajamos la bandera INT0IF
    if(LED_estado == LED_apagado){
        LED_estado = LED_parpadeo;
    }
    else{
        LED_estado = LED_apagado;
    }
}

void __interrupt(irq(default)) default_ISR(void){

}
```

Implementación 2024-1

El módulo DAC del PIC18F57Q43

- Referencia: capítulo 41 del datasheet
- Resolución: 8bits
- Tipo arreglo R-2R
- Se puede usar el FVR como voltaje de referencia positivo (bits PSS)
- Fórmula:

Equation 41-1. DAC Output Equation

$$DACx_output = \left((V_{REF+} - V_{REF-}) \times \frac{DACR}{2^n}\right) + V_{REF-}$$

- Registros:

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7D	DAC1DATL	7:0	DAC1R[7:0]							
0x7E	Reserved									
0x7F	DAC1CON	7:0	EN		OE[1:0]		PSS[1:0]			NSS

El módulo DAC del PIC18F57Q43

Name: DACxCON
Address: 0x7F

Digital-to-Analog Converter Control Register

Bit	7	6	5	4	3	2	1	0
	EN		OE[1:0]		PSS[1:0]			NSS
Access	R/W		R/W	R/W	R/W	R/W		R/W
Reset	0		0	0	0	0		0

Bit 7 – EN DAC Enable

Value	Description
1	DAC is enabled
0	DAC is disabled

Bits 5:4 – OE[1:0] DAC Output Enable

OE	DAC Outputs
11	DACxOUT is disabled
10	DACxOUT is enabled on pin RA2 only
01	DACxOUT is enabled on pin RB7 only
00	DACxOUT is disabled

Bits 3:2 – PSS[1:0] DAC Positive Reference Selection

PSS	DAC Positive Reference
11	Reserved, do not use
10	FVR Buffer 2
01	V_{REF+}
00	V_{DD}

Bit 0 – NSS DAC Negative Reference Selection

NSS	DAC Negative Reference
1	V_{REF-}
0	V_{SS}

El módulo DAC del PIC18F57Q43

- Procedimiento:
  - Establecer el puerto de salida de la señal (RA2 ó RB7) con DAC1CON bits 5-4 (OE)
  - Seleccionar referencia positiva (PSS) y referencia negativa (NSS) del DAC
  - Establecer puerto seleccionado anteriormente como salida analógica (TRISx y ANSELx):
    - Por Ejemplo la salida en RA2:
    - TRISAbits.TRISA2 = 0; //RA2 como salida
    - ANSELAbits.ANSELA2 = 1; //RA2 como analógica
  - Habilitar el módulo con DAC1CON bit 7 (EN)
  - Establecer el valor de 8 bits en DAC1DATL

Ejemplo ADC-DAC 2024-2

- Se tendrá una entrada analógica en RA0 y una salida analógica en RA2.
- El ADC leerá el canal RA0 y estará en configuración de justificación del resultado a la izquierda para que los 8 bits más significativos del resultado esté alojado en el registro ADRESH.
- El DAC estará configurado con Vref+ en Vdd (5V) y Vref- en Vss (0V)
- Se hará el trasladado del ADRESH hacia DAC1DATL (registro del DAC para establecer un valor en su salida).
- Se visualizará ADRESH y de DAC1DATL en el LCD

Ejemplo ADC-DAC 2024-2

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    //conf de la fuente de reloj HFINTOSC a 32MHz
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;
    OSCEN = 0x40;
    //conf de las E/S
    TRISAbits.TRISA0 = 1;
    ANSELAbits.ANSELA0 = 1;
    TRISAbits.TRISA2 = 0;
    ANSELAbits.ANSELA2 = 1;
    //conf del ADC
    TRISAbits.TRISA0 = 1;   //RA0 como entrada
    ANSELAbits.ANSELA0 = 1; //RA0 como analogico
    ADCON0bits.FM = 0;      //resultado just izq (8bits en ADRESH)
    ADCON0bits.CS = 1;      //fuente de reloj del ADC en ADCRC
    ADCON0bits.ADON = 1;    //encendemos el ADC
    //conf del DAC
    DAC1CON = 0xA0;     //DAC enabled, RA2 output, Vref+=Vdd, Vref-=Vss
    DAC1DATL = 0x00;    //DAC salida = 0V
    //conf inicial del LCD
    LCD_INIT();
}
```

```c
unsigned char tomamuestra_ADC(void){
    ADPCH = 0;              //canal seleccionado RA0
    ADCON0bits.GO = 1;      //toma una muestra
    while(ADCON0bits.GO == 1);  //espero a que termine la toma de muestra
    return ADRESH;
}

void main(void) {
    configuro();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE2("Prueba ADC-DAC");
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE2("con PIC18F57Q43");
    __delay_ms(3000);
    BORRAR_LCD();
    while(1){
        DAC1DATL = tomamuestra_ADC();
        POS_CURSOR(1,0);
        ESCRIBE_MENSAJE2("ADRESH:");
        LCD_ESCRIBE_VAR_CHAR(ADRESH, 3);
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE2("DAC1DATL:");
        LCD_ESCRIBE_VAR_CHAR(DAC1DATL, 3);
    }
}
```

Ejemplo 2024-2

- Parpadeo de un LED en RB0 con control de velocidad de parpadeo con RB4

```c
#include <xc.h>
#include "cabecera.h"

#define _XTAL_FREQ 32000000UL

void configuro(void){
    //config fuente de reloj
    OSCCON1 = 0x60;     //HFINTOSC POSTS 1:1
    OSCFRQ = 0x06;      //HFINTOSC 32MHz
    OSCEN = 0x40;       //HFINTOSC enabled
    //config de las E/S
    TRISBbits.TRISB4 = 1;   //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enabled
    TRISBbits.TRISB0 = 0;   //RB0 salida
    ANSELBbits.ANSELB0 = 0; //RB0 digital
}
```

```c
void main(void) {
    configuro();
    while(1){
        if(PORTBbits.RB4 == 0){
            LATBbits.LATB0 = 1; //enciende el LED
            __delay_ms(100);
            LATBbits.LATB0 = 0; //apaga el LED
            __delay_ms(100);
        }
        else{
            LATBbits.LATB0 = 1; //enciende el LED
            __delay_ms(300);
            LATBbits.LATB0 = 0; //apaga el LED
            __delay_ms(300);
        }
    }
}
```

Ejemplo 2024-2

Parpadeo de un LED en RB0 con control de velocidad de parpadeo con interrupción INT0:

- Para trabajar con la interrupción externa INT0 hay que revisar hoja técnica:
  - Capítulo 11 – VIC
  - Capítulo 21 – PPS
- Para la INT0, el INT0IE se encuentra en PIE1 bit0, el INT0IF se encuentra en PIR1 bit0. El GIE (habilitador global de interrupciones) se encuentra en el registro INTCON0 bit7.
- El INTEDG0 que se encuentra en INTCON0 bit0 por defecto esta en 1 (detección de la INT0 activo en alto), por lo que se deberá de cambiar a detección activo en bajo.
- La INT0 por defecto esta en RB0, para cambiarlo a RB4 debo de configurar el PPS: reg. INT0PPS = 0x0C;

Ejemplo 2024-2

```c
#include <xc.h>
#include "cabecera.h"

#define _XTAL_FREQ 32000000UL
#define LENTO 0
#define RAPIDO 1

unsigned char estado = LENTO;   //variable que conti

void configuro(void){
    //config fuente de reloj
    OSCCON1 = 0x60;      //HFINTOSC POSTS 1:1
    OSCFRQ = 0x06;       //HFINTOSC 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config de las E/S
    TRISBbits.TRISB4 = 1;   //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enabled
    TRISBbits.TRISB0 = 0;   //RB0 salida
    ANSELBbits.ANSELB0 = 0; //RB0 digital
    //config de la INT0
    INT0PPS = 0x0C;         //la INT0 entra por RB4
    INTCON0bits.INT0EDG = 0;    //deteccion activo e
    PIE1bits.INT0IE = 1;    //habilitamos INT0
    INTCON0bits.GIE = 1;    //encendemos interrupcio
}
```

```c
//Sistema del RB4 en modo latch con interrupciones
void main(void) {
    configuro();
    while(1){
        if(estado == RAPIDO){
            LATBbits.LATB0 = 1; //enciende el LED
            __delay_ms(100);
            LATBbits.LATB0 = 0; //apaga el LED
            __delay_ms(100);
        }
        else{
            LATBbits.LATB0 = 1; //enciende el LED
            __delay_ms(300);
            LATBbits.LATB0 = 0; //apaga el LED
            __delay_ms(300);
        }
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos bandera de INT0
    if(estado == LENTO){
        estado = RAPIDO;
    }
    else{
        estado = LENTO;
    }
}
```

Ejemplo 2024-2

Ejemplo 2024-2

- Si queremos utilizar el pulsador integrado en el Curiosity Nano PIC18F57Q43, debemos tener en cuenta que dicho pulsador esta conectado al puerto RB4 en configuración activo en bajo y sin resistencia de pull-up externa.
- La interrupción externa INT1 por defecto esta conectado al RB1, para poder utilizar la INT1 con el pulsador integrado, debemos de reasignar utilizar el PPS: INT1PPS = 0x0C;
- La INT1 se puede configurar si detectas rising_edge o falling_edge, se configura en el bit INT1EDG del reg INTCON0 (1 para rising_edge y 0 para falling_edge)
- Para activar la INT1: bit INT1IE=1 (reg PIE6), la bandera INT1IF esta en el reg PIR6.
- No olvidar de activar el GIE=1 del reg INTCON0

Ejemplo 2024-2

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 32000000UL
#define E_LENTO 0
#define E_RAPIDO 1

unsigned char estado=E_LENTO;

void configuro(void){
    //config de la fuente de reloj
    OSCCON1 = 0x60;      //HFINTOSC POSTS 1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config de las E/S
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 con pullup
    TRISBbits.TRISB0 = 0;   //RB0 como salida
    ANSELBbits.ANSELB0 = 0; //RB0 como digital
    //config de la INT1
    INT1PPS = 0x0C;      //RB4 sea la INT1
    INTCON0bits.INT1EDG = 0;    //INT1 en falling-edge
    PIE6bits.INT1IE = 1;    //INT1 enabled
    INTCON0bits.GIE = 1;    //GIE enabled
}
```

```c
void condicion(void){
        if(estado == E_LENTO){
            __delay_ms(500);
        }
        else{
            __delay_ms(100);
        }
}

void main(void) {
    configuro();
    while(1){
        LATBbits.LATB0 = 1;
        condicion();
        LATBbits.LATB0 = 0;
        condicion();
    }
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajamos la bandera
    if(estado == E_LENTO){
        estado = E_RAPIDO;
    }
    else{
        estado = E_LENTO;
    }
}
```

```c
void __interrupt(irq(default)) DEFAULT_ISR(void){

}
```

Ejemplo 2024-2

Ejemplo 2024-2

- El pulsador en RB4 va a determinar la velocidad del parpadeo, si no esta presionado el parpadeo será lento, si lo mantienes presionado el parpadeo será rápido.
- El botón en RB4 es activo en bajo y no tiene pullup externa.

- ¿Y cómo hago para que la función de cambio de velocidad de parpadeo se mantenga enclavado?
  - Cambiar la función del pulsador a LATCH
  - Vamos a utilizar una interrupción externa INT1 para dicha acción.
  - INT1 por defecto tiene la conexión a RB1, debemos de acudir al PPS para cambiar la asignación de la INT1 hacia el puerto RB4.
    - INT1PPS = 0x0C; //cambio de RB1 a RB4 para la INT1
  - INT1 por defecto detecta flanco positivo, nuestro botón en RB4 es de flanco negativo, debemos de modificar dicha opción con lo siguiente:
    - INTCON0bits.INT1EDG = 0; //falling_edge, dicho se encuentra en el reg INTCON0
  - Para activar la INT1: En el bit INT1IE=1 (reg PIE6), El bit GIE=1 (reg INTCON0)

Ejemplo 2024-2

```c
8   #include <xc.h>
9   #include "cabecera.h"
10  #define _XTAL_FREQ 32000000UL
11  #define RAPIDISIMO 1
12  #define LENTISIMO 0
13
14  unsigned char ESTADO = LENTISIMO;
15
16  void configuro(void){
17      //config de la fuente de reloj
18      OSCCON1 = 0x60;      //NOSC=HFINTOSC POSTS 1:1
19      OSCFRQ = 0x06;       //HFINTOSC a 32MHz
20      OSCEN = 0x40;        //HFINTOSC enabled
21      //config las E/S
        TRISBbits.TRISB4 = 1;    //RB4 como entrada
        ANSELBbits.ANSELB4 = 0; //RB4 como digital
        WPUBbits.WPUB4 = 1;      //RB4 pullup enabled
        TRISBbits.TRISB0 = 0;    //RB0 como salida
        ANSELBbits.ANSELB0 = 0; //RB0 como digital
27      //config de la INT1
28      INT1PPS = 0x0C;      //Asignando RB4 para INT1
        INTCON0bits.INT1EDG = 0;     //INT1 en flanco negativo
        PIE6bits.INT1IE = 1;     //habilitando INT1
        INTCON0bits.GIE = 1;     //habilitador GIE en uno
        PIR6bits.INT1IF = 0;     //bajamos la bandera inicialmente
33  }
```

```c
35  void main(void){
36      configuro();
37      while(1){
            LATBbits.LATB0 = 1;
39          if(ESTADO == LENTISIMO){
40              __delay_ms(500);
41          }
42          else if(ESTADO == RAPIDISIMO){
43              __delay_ms(100);
44          }
            LATBbits.LATB0 = 0;
46          if(ESTADO == LENTISIMO){
47              __delay_ms(500);
48          }
49          else if(ESTADO == RAPIDISIMO){
50              __delay_ms(100);
51          }
52      }
53  }
54
55  void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
        PIR6bits.INT1IF = 0;     //bajamos la bandera de INT1
57      if(ESTADO == RAPIDISIMO){
58          ESTADO = LENTISIMO;
59      }
60      else{
61          ESTADO = RAPIDISIMO;
62      }
63  }
```

Ejercicio 2024-2

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 32000000UL
#define RAPIDO 1
#define LENTO 0

unsigned char ESTADO = LENTO;

void configuro(void){
    //config la fuente de reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC POSTS=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config de las E/S
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enabled
    TRISBbits.TRISB0 = 0;   //RB0 como salida
    ANSELBbits.ANSELB0 = 0; //RB0 como digital
    //config la INT1
    INT1PPS = 0x0C;      //asignar INT1 hacia RB4
    INTCON0bits.INT1EDG = 0;     //INT1 en falling_edge
    PIE6bits.INT1IE = 1;         //INT1 enabled
    INTCON0bits.GIE = 1;         //interrupciones habilitadas
    PIR6bits.INT1IF = 0;         //bajamos bandera de INT1
}
```

```c
void pregunta(void){
    if(ESTADO == LENTO){
        __delay_ms(500);
    }
    else if(ESTADO == RAPIDO){
        __delay_ms(100);
    }
}

void main(void) {
    configuro();
    while(1){
        LATBbits.LATB0 = 1; //enciende el LED
        pregunta();
        LATBbits.LATB0 = 0; //apago el LED
        pregunta();
    }
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;     //bajamos bandera INT1
    if(ESTADO == LENTO){
        ESTADO = RAPIDO;
    }
    else if(ESTADO == RAPIDO){
        ESTADO = LENTO;
    }
}
```

Ejercicio 2024-2 Mejora aplicando LATCH

- Para que parpadee rapido el LED se tiene que mantener pulsador el boton en RB4, es un inconveniente.
- Se puede mejorar el desarrollo aplicando LATCH, por lo tanto la acción del pulsador va a cambiar entre dos estados: parpadeo rápido y parpadeo lento.
- Significa que debes de considerar dos estados de trabajo.
- Utilizando interrupciones se puede lograr hacer esta mejora, tenemos tres: INT0, INT1, INT2, en esta oportunidad usaremos INT2.
- La INT2 es una interrupción externa que por defecto detecta el flanco positivo (rising_edge) y por defecto estra conectado en RB2.
- Vamos a utilizar el PPS para cambiar la asignación de la INT2 hacia RB4:
  - INT2PPS = 0x0C;
- Vamos a cambiar la detección de flanco positivo a flanco negativo en la INT2:
  - INTCON0bits.INT2EDG = 0; //bit INT2EDG ubicado en el reg INTCON0

Ejercicio mejorado 2024-2 EL56-1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 32000000UL
#define RAPIDO 1
#define LENTITO 0

unsigned char ESTADO = LENTITO;

void configuro(void){
    //conf la fuente de reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC POSTS=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config de las E/S
    TRISBbits.TRISB4 = 1;   //RB4 es entrada
    ANSELBbits.ANSELB4 = 0; //RB4 es digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enabled
    TRISBbits.TRISB0 = 0;   //RB0 es salida
    ANSELBbits.ANSELB0 = 0; //RB0 es digital
    //config de la INT2
    INT2PPS = 0x0C;         //asignando RB4 para INT2
    INTCON0bits.INT2EDG = 0;    //falling_edge para INT2
    PIE10bits.INT2IE = 1;   //activo INT2
    INTCON0bits.GIE = 1;    //habilitador global de ints ON
    PIR10bits.INT2IF = 0;   //bajamos bandera INT2
}
```

```c
void pregunta(void){
    if(ESTADO == RAPIDO){
        __delay_ms(100);
    }
    else if(ESTADO == LENTITO){
        __delay_ms(500);
    }
}

void main(void){
    configuro();
    while(1){
        LATBbits.LATB0 = 1; //enciendo el LED
        pregunta();
        LATBbits.LATB0 = 0; //apagado el LED
        pregunta();
    }
}

void __interrupt(irq(IRQ_INT2)) INT2_ISR(void){
    PIR10bits.INT2IF = 0;   //bajamos bandera de INT2
    if(ESTADO == LENTITO){
        ESTADO = RAPIDO;
    }
    else if(ESTADO == RAPIDO){
        ESTADO = LENTITO;
    }
}
```

Ejercicio mejorado 2024-2 EL56-2

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 32000000UL
#define LENTASO 0
#define RAPIDASO 1

unsigned char ESTADO = LENTASO;

void configuro(void){
    //config la fuente reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config las E/S
    TRISBbits.TRISB0 = 0;   //RB0 sea salida
    ANSELBbits.ANSELB0 = 0; //RB0 sea digital
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB$ pull-up enabled
    //config de la INT2
    INT2PPS = 0x0C;         //INT2 tenga asignado el RB4
    INTCON0bits.INT2EDG = 0;    //INT2 en flanco negativo
    PIE10bits.INT2IE = 1;   //INT2 habilitada
    INTCON0bits.GIE = 1;    //habilitador global encendido
    PIR10bits.INT2IF = 0;   //bajamos la bandera de INT2
}
```

```c
void __interrupt(irq(default)) default_ISR(void){

}
```

```c
void pregunto_delay(void){
    if(ESTADO == RAPIDASO){
        __delay_ms(100);
    }
    if(ESTADO == LENTASO){
        __delay_ms(500);
    }
}

void main(void) {
    configuro();
    while(1){
        LATBbits.LATB0 = 1; //enciendo LED
        pregunto_delay();
        LATBbits.LATB0 = 0; //apagar LED
        pregunto_delay();
    }
}

void __interrupt(irq(IRQ_INT2)) INT2_ISR(void){
    PIR10bits.INT2IF = 0;   //bajamos la bandera de INT2
    if(ESTADO == LENTASO){
        ESTADO = RAPIDASO;
    }
    else if(ESTADO == RAPIDASO){
        ESTADO = LENTASO;
    }
}
```

Ejercicio 2024-2

- Circuito implementado

Ejercicio 2024-2

- Funcionamiento:

Ejercicio 2024-2

- Ejemplo:

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    //config de la fuente de reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;  //HFINTOSC a 32MHz
    OSCEN = 0x40;
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(17);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,2);
    ESCRIBE_MENSAJE("Hola upcino",11);
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Biomecaele 20242",16);
    while(1){
    }
}
```

Ej. 2024-2 Agregando la función de un pulsador

Ej. 2024-2 Agregando la función de un pulsador

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    //config de la fuente de reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;  //HFINTOSC a 32MHz
    OSCEN = 0x40;
    //config de las E/S
    TRISBbits.TRISB4 = 1;       //RB4 input
    ANSELBbits.ANSELB4 = 0;     //RB4 digital
    WPUBbits.WPUB4 = 1;         //pullup enabled
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(17);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}
```

```c
void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,1);
    ESCRIBE_MENSAJE("Estado de RB4",13);
    while(1){
        if(PORTBbits.RB4 == 0){
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("PRESIONADO   ", 13);
        }
        else{
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("SIN PRESIONAR", 13);
        }
    }
}
```

Ejercicio 2024-2

- Conversión ADC: Canal RA0/ANA0

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned int resultado_ADC;
unsigned char millar, centena, decena, unidad;

void configuro(void){
    //config fuente de reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;      //HFINTOSC a 32MHz
    OSCEN = 0x40;
    //config de las E/S
    TRISBbits.TRISB4 = 1;   //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup activado
    TRISAbits.TRISA0 = 1;   //RA0 entrada
    ANSELAbits.ANSELA0 = 1; //RA0 analogica
    //config del ADC
    ADCON0bits.FM = 1;  //just a la derecha
    ADCON0bits.CS = 1;  //fuente de reloj ADCRC
    ADPCH = 0x00;       //canal RA0/ANA0 seleccionado
    ADREF = 0x00;
    ADCON0bits.ADON = 1;  //encendemos el ADC
}

void LCD_INIT(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(16);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

unsigned int leer_ADC(void){
    ADCON0bits.GO = 1;      //inicia la toma de una muestra
    while(ADCON0bits.GO);   //espera a que termine de convertir
    //ya tenemos el resultado de la conversion ADRESH:ADRESL
    return (ADRESH << 8) + ADRESL;
}

void conversion(unsigned int dato){
    millar = dato / 1000;
    centena = (dato % 1000) / 100;
    decena = (dato % 100) / 10;
    unidad = dato % 10;
}

void main(void) {
    configuro();
    LCD_INIT();
    //me voy a la primera linea del LCD
    POS_CURSOR(1,2);
    ESCRIBE_MENSAJE("Upcino 2024-2", 13);
    while(1){
        resultado_ADC = leer_ADC();
        conversion(resultado_ADC);
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("Canal ANA0:", 11);
        ENVIA_CHAR(millar+0x30);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
    }
}
```

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned char millar, centena, decena, unidad;

void configuro(void){
    //config de la fuente de reloj
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;  //HFINTOSC a 32MHz
    OSCEN = 0x40;
    //config de las E/S
    TRISBbits.TRISB4 = 1;       //RB4 input
    ANSELBbits.ANSELB4 = 0;     //RB4 digital
    WPUBbits.WPUB4 = 1;         //pullup enabled
    //config del ADC
    TRISAbits.TRISA0 = 1;       //RA0 input
    ANSELAbits.ANSELA0 = 1;     //RA0 analog
    ADCON0bits.FM = 1;          //just a la derecha
    ADCON0bits.CS = 1;          //fuente de reloj ADCRC
    ADCON0bits.ADON = 1;        //ADC encendido
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(17);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

unsigned int toma_muestra(void){
    ADPCH = 0x00;
    ADCON0bits.GO = 1;      //toma una muestra
    while(ADCON0bits.GO == 1);  //va a esperar a que termine de convertir
    return ((ADRESH << 8) + ADRESL);
}

void convierte(unsigned int dato){
    millar = dato / 1000;
    centena = (dato % 1000) / 100;
    decena = (dato % 100) / 10;
    unidad = dato % 10;
}

void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,2);
    ESCRIBE_MENSAJE("Lectura ADC",11);
    while(1){
        convierte(toma_muestra());
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("CH0: ",5);
        ENVIA_CHAR(millar+0x30);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
        __delay_ms(50);
    }
}
```

- Conversión ADC: Canal RA0/ANA0

- Conversión ADC: Canal RA0/ANA0

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned char millar,centena,decena,unidad;

void configuro(void){
    //config fuente de reloj
    OSCCON1 = 0x60; //NOSC=HFINTOSC NDIV=1:1
    OSCFRQ = 0x06;  //HFINTOSC a 32MHz
    OSCEN = 0x40;   //HFINTOSC enabled
    //config del RB4
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 con pullup
    //config el ADC
    TRISAbits.TRISA0 = 1;   //RA0 es entrada
    ANSELAbits.ANSELA0 = 1; //RA0 es analogica
    ADCON0bits.FM = 1;      //resultado just derecha
    ADCON0bits.CS = 1;      //fuente de reloj del ADC en ADCRC
    ADCON0bits.ADON = 1;    //enciendo el ADC
}

unsigned int tomamuestra_ADC(void){
    ADPCH = 0;      //selecciono el canal RA0/ANA0
    ADCON0bits.GO = 1;  //procede a tomar una muestra
    while(ADCON0bits.GO == 1);  //espera a que termine
    return ((ADRESH << 8) + ADRESL);
}

void convierte(unsigned int dato){
    millar = dato / 1000;
    centena = (dato % 1000) / 100;
    decena = (dato % 100) / 10;
    unidad = dato % 10;
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(21);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Tercer ejemplo:",15);
    while(1){
        convierte(tomamuestra_ADC());
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("CH0:",4);
        ENVIA_CHAR(millar+0x30);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
    }
}
```

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    //conf la fuente de reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(23);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    __delay_ms(50);
}
```

```c
void main(void){
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Hola mundo UPC!",15);
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Microcontrolador",16);
    while(1){
    }
}
```

- Visualización de mensajes en el LCD con la librería LCD

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned char millar, centena, decena, unidad;

void configuro(void){
    //conf la fuente de reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //conf el ADC
    TRISAbits.TRISA0 = 1;   //RA0 como entrada
    ANSELAbits.ANSELA0 = 1; //RA0 como analogica
    ADCON0bits.FM = 1;      //resultado just a la derecha
    ADCON0bits.CS = 1;      //fuente de reloj del ADC = ADCRC
    ADCON0bits.ADON = 1;    //encendemos el adc
}

unsigned int tomamuestra_ADC(void){
    ADPCH = 0;  //seleccionamos RA0
    ADCON0bits.GO = 1;      //que toma una muestra
    while(ADCON0bits.GO == 1);  //espero a que termine de tomar la muestra
    return ((ADRESH << 8) + ADRESL);
}

void conviertemon(unsigned int dato){
    millar = dato / 1000;
    centena = (dato % 1000) / 100;
    decena = (dato % 100) / 10;
    unidad = dato % 10;
}
```

```c
void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(23);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    __delay_ms(50);
}

void main(void){
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Ejemplo #3",10);
    while(1){
        conviertemon(tomamuestra_ADC());
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("RA0: ",5);
        ENVIA_CHAR(millar+0x30);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
    }
}
```

- Conversión ADC: Canal RA0/ANA0

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    //config fuente de reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config del boton
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 con pullup activo
}

void lcd_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(26);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

/*void retardado(unsigned int valor){
    unsigned int temporal;
    for(temporal=0;temporal<valor;temporal++){
        __delay_us(1);
    }
}*/
```

```c
void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Ejemplo #2",10);
    while(1){
        if(PORTBbits.RB4 == 0){
            //aqui cuando es verdad
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Machucado   ",12);
        }
        else{
            //aqui cuando es falso
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Sin machucar",12);
        }
    }
}
```

- Visualización del estado del pulsador en RB4

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned char millar, centena, decena, unidad;

void configuro(void){
    //config fuente de reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config del boton
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 con pullup activo
    //config ADC
    TRISAbits.TRISA0 = 1;   //RA0 como entrada
    ANSELAbits.ANSELA0 = 1; //RA0 como analogico
    ADCON0bits.FM = 1;      //resultado just derecha
    ADCON0bits.CS = 1;      //fuente de reloj del ADC en ADCRC
    ADCON0bits.ADON = 1;    //encendemos el ADC
}

unsigned int tomamuestra_ADC(void){
    ADPCH = 0;              //canal seleccionado RA0
    ADCON0bits.GO = 1;      //toma una muestra
    while(ADCON0bits.GO == 1);  //espero a que termine la toma de muestra
    return ((ADRESH << 8) + ADRESL);
}
```

```c
conviertemon(unsigned int dato){
    millar = dato / 1000;
    centena = (dato % 1000) / 100;
    decena = (dato % 100) / 10;
    unidad = dato % 10;
}

void lcd_init(void){...9 lines }

void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Ejemplo #3",10);
    while(1){
        conviertemon(tomamuestra_ADC());
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("Canal RA0:",10);
        ENVIA_CHAR(millar+0x30);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
    }
}
```

- Conversión ADC: Canal RA0/ANA0

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 32000000UL

unsigned char millar, centena, decena, unidad;

void configuro(void){
    //config de la fuente de reloj
    OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
    OSCFRQ = 0x06;       //HFINTOSC a 32MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //config del boton en RB4
    TRISBbits.TRISB4 = 1;   //RB4 entrada
    ANSELBbits.ANSELB4 = 0; //RB4 digital
    WPUBbits.WPUB4 = 1;     //RB4 pullup enabled
    //config el ADC
    TRISAbits.TRISA0 = 1;   //RA0 entrada
    ANSELAbits.ANSELA0 = 1; //RA0 analogica
    ADCON0bits.FM = 1;      //res just a la derecha
    ADCON0bits.CS = 1;      //clk source a ADCRC
    ADCON0bits.ADON = 1;    //encendemos el ADC
}

unsigned int tomamuestra_ADC(unsigned char canal){
    ADPCH = canal;              //seleccionamos el canal
    ADCON0bits.GO = 1;          //iniciamos la toma
    while(ADCON0bits.GO == 1);  //esperamos a que termine
    return ((ADRESH << 8) + ADRESL);
}
```

```c
void convierte(unsigned int dato){
    millar = dato / 1000;
    centena = (dato % 1000) / 100;
    decena = (dato % 100) / 10;
    unidad = dato % 10;
}

void lcd_init(void){...10 lines }

void main(void) {
    configuro();
    lcd_init();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Ejercicio No.3",14);
    while(1){
        convierte(tomamuestra_ADC(0));
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE("CH0: ",5);
        ENVIA_CHAR(millar+0x30);
        ENVIA_CHAR(centena+0x30);
        ENVIA_CHAR(decena+0x30);
        ENVIA_CHAR(unidad+0x30);
    }
}
```

- Conversión ADC: Canal RA0/ANA0

Asignación Semana 10:

- Crear una “splash screen” (pantalla inicial de la aplicación), incluirle efectos (negativo, desplazamiento, etc), caracteres personalizados, logotipos o imágenes, etc.
- El microcontrolador deberá de leer dos canales analógicos provenientes de potenciómetros en configuración divisor de tensión, la visualización debe de mostrarse el valor en voltios.

Asignación (Minilaboratorio) Semana 10

- Desarrollar un "Splash Screen" de mínimo 5 segundos y máximo 15 segundos, dicha visualización puede contener caracteres personalizados, animaciones, desplazamientos, textos en negativo, logotipos.
- Luego del splash screen proceder con la aplicación de leer dos canales analógicos con el ADC (fuente de entrada de señal analógica potenciómetros de 10KΩ en configuración divisor de tensión), los valores deberán de visualizarse en el LCD en valor resistivo (0 a 10KΩ).

Asignación (Minilaboratorio) Semana 10

- Desarrollar un "Splash Screen" de mínimo 5 segundos y máximo 15 segundos, dicha visualización puede contener caracteres personalizados, animaciones, desplazamientos, textos en negativo, logotipos.
- Luego del splash screen proceder con la aplicación de leer dos canales analógicos con el ADC (fuente de entrada de señal analógica potenciómetros de 10KΩ en configuración divisor de tensión), los valores deberán de visualizarse en el LCD en valor porcentual (0% a 100%)

Ejemplo de aplicación

- Construir un hardware de prueba consistente en un display de siete segmentos del tipo cátodo común y dos pulsadores activos en bajo.

Ejemplo de aplicación 2025-1

- Construir un hardware de prueba consistente en un display de siete segmentos del tipo cátodo común y dos pulsadores activos en bajo.

Ejemplo de aplicación 2025-1

- Prender y apagar todos los segmentos del display de siete segmentos usando el pulsador en RB0.
  - Vamos a emplear la interrupción externa INT0 para la acción de encender y apagar todos los segmentos del display
  - Tenemos que configurar: RD(6:0) como salidas digitales, INT0 con PPS hacia RB0 (configurado como entrada digital) y detección de flanco negativo.

Ejemplo de aplicación 2025-1

- Prender y apagar todos los segmentos del display de siete segmentos usando el pulsador en RB0.
  - Configuracion del PPS

Name: xxxPPS → INT0PPS = 0x08

Peripheral Input Selection Register

Bit	7	6	5	4	3	2	1	0
	0	0	0	PORT[2:0]	1	0	PIN[2:0]	0
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			m	m	m	m	m	m

Bits 5:3 – PORT[2:0] Peripheral Input PORT Selection[1]
See the PPS Input Selection Table for the list of available Ports and default pin locations.

PORT	Selection
101	PORTF
100	PORTE
011	PORTD
010	PORTC
001	PORTB
000	PORTA

Reset States: POR = mmm
All other Resets = uuu

Bits 2:0 – PIN[2:0] Peripheral Input PORT Pin Selection[2]
Reset States: POR = mmm
All other Resets = uuu

Value	Description
111	Peripheral input is from PORTx Pin 7 (Rx7)
110	Peripheral input is from PORTx Pin 6 (Rx6)
101	Peripheral input is from PORTx Pin 5 (Rx5)
100	Peripheral input is from PORTx Pin 4 (Rx4)
011	Peripheral input is from PORTx Pin 3 (Rx3)
010	Peripheral input is from PORTx Pin 2 (Rx2)
001	Peripheral input is from PORTx Pin 1 (Rx1)
000	Peripheral input is from PORTx Pin 0 (Rx0)

Ejemplo de aplicación 2025-1

- Prender y apagar todos los segmentos del display de siete segmentos usando el pulsador en RB0.
  - Código ejemplo
  - Se visualiza lo siguiente:
    - El _XTAL_FREQ (línea 16) sirve para indicarle al compilador la frecuencia de trabajo del CPU a efectos de hacer los cálculos adecuados en las instrucciones del XC8 de alto nivel que tengan que ver con el tiempo. Ejemplo __delay_ms(200)
    - Configuración de PPS para la INT0 (línea 30).
    - La función para la atención de interrupciones vectorizadas (línea 45).
    - Ejecución de una instrucción de ensamblador dentro del código en C (línea 47).

```
13  #include <xc.h>
14  #include "cabecera.h"
15
16  #define _XTAL_FREQ 4000000UL
17
18  void configuro(void){
19      //configuracion del modulo de oscilador
20      OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV 1:1
21      OSCFRQ = 0x02;       //HFINTOSC a 4MHz
22      OSCEN = 0x40;        //HFINTOSC enabled
23      //configuracion de las E/S
24      TRISD = 0x80;        //RD6 al RD0 como salidas
25      ANSELD = 0x80;       //RD6 al RD0 como digitales
        TRISBbits.TRISB0 = 1;   //RB0 como entrada
        ANSELBbits.ANSELB0 = 0; //RB0 como digital
        WPUBbits.WPUB0 = 1;     //RB0 con pullup
29      //configuracion de la INT0
30      INT0PPS = 0x08;          //INT0 proviene de RB0
        PIE1bits.INT0IE = 1;     //habilitando la INT0
        PIR1bits.INT0IF = 0;     //bajamos la bandera INT0IF
        INTCON0bits.INT0EDG = 0;    //INT0 en flanco negativo
        INTCON0bits.GIE = 1;        //Habilitador global de interrups ON
35      //condiciones iniciales de operacion
36      LATD = 0x00;             //segmentos apagados del display
37  }
38
39  void main(void) {
40      configuro();
41      while(1);
42  }
43
44  //funcion de la interrupcion INT0 (modo vectorizado MVECEN ON)
45  void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
        PIR1bits.INT0IF = 0;     //bajamos la bandera INT0IF
47      asm("comf LATD, 1");            //aplicas complemento a RD
48  }
```

Ejemplo de aplicación 2025-1

- Efecto de visualización de un gusanito moviéndose en forma de un ocho en el display de siete segmentos.
- Se observa lo siguiente:
  - Uso de un array para construir una LUT en la línea 18 (función de decodificación).
  - Uso de función de retardo en la línea 36

```
13  #include <xc.h>
14  #include "cabecera.h"
15
16  #define _XTAL_FREQ 4000000UL
17
18  unsigned char gusanito[]={0x01, 0x02, 0x40, 0x10, 0x08, 0x04, 0x40, 0x20}
19  unsigned char cuenta = 0;
20
21  void configuro(void){
22      //configuracion del modulo de oscilador
23      OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV 1:1
24      OSCFRQ = 0x02;       //HFINTOSC a 4MHz
25      OSCEN = 0x40;        //HFINTOSC enabled
26      //configuracion de las E/S
27      TRISD = 0x80;        //RD6 al RD0 como salidas
28      ANSELD = 0x80;       //RD6 al RD0 como digitales
29  }
30
31  void main(void) {
32      configuro();
33      while(1){
34          for(cuenta=0;cuenta<8;cuenta++){
35              LATD = gusanito[cuenta];
36              __delay_ms(300);
37          }
38      }
39  }
```

Ejempo de aplicación 2025-1

- Efecto de visualización de un gusanito moviéndose en forma de un ocho en el display de siete segmentos con opción de sentido usando la INT0 en RB0

```c
#include <xc.h>
#include "cabecera.h"

#define _XTAL_FREQ 4000000UL

unsigned char gusanito[]={0x01, 0x02, 0x40, 0x10, 0x08, 0x04, 0x40, 0x20};
unsigned char cuenta = 0;
unsigned char sentido = 0;      //cero para izquierda, uno para derecha

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;     //NOSC=HFINTOSC, NDIV 1:1
    OSCFRQ = 0x02;      //HFINTOSC a 4MHz
    OSCEN = 0x40;       //HFINTOSC enabled
    //configuracion de las E/S
    TRISD = 0x80;       //RD6 al RD0 como salidas
    ANSELD = 0x80;      //RD6 al RD0 como digitales
    TRISBbits.TRISB0 = 1;   //RB0 como entrada
    ANSELBbits.ANSELB0 = 0; //RB0 como digital
    WPUBbits.WPUB0 = 1;     //RB0 con pullup
    //configuracion de la INT0
    INT0PPS = 0x08;         //INT0 proviene de RB0
    PIE1bits.INT0IE = 1;    //habilitando la INT0
    PIR1bits.INT0IF = 0;    //bajamos la bandera INT0IF
    INTCON0bits.INT0EDG = 0;    //INT0 en flanco negativo
    INTCON0bits.GIE = 1;        //Habilitador global de interrups ON
}
```

```c
void main(void) {
    configuro();
    while(1){
        if(sentido == 0){
            for(cuenta=0;cuenta<8;cuenta++){
                LATD = gusanito[cuenta];
                __delay_ms(200);
            }
        }
        if(sentido == 1){
            for(cuenta=8;cuenta>=1;cuenta--){
                LATD = gusanito[cuenta-1];
                __delay_ms(200);
            }
        }
    }
}

//funcion de la interrupcion INT0 (modo vectorizado MVECEN ON)
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos la bandera INT0IF
    switch(sentido){
        case 0:
            sentido = 1;
            break;
        case 1:
            sentido = 0;
            break;
        default:
            sentido = 0;
    }
}
```

Ejemplo de aplicación 2025-1

- Animación de gusanito v2 en el display de siete segmentos y control velocidad con INT0

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL


unsigned char animacion[]={ 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x01, 0x02, 0x40,
                            0x20, 0x01, 0x02, 0x04, 0x08, 0x10, 0x40, 0x04, 0x08,
                            0x10, 0x20};

unsigned char x_var = 0;
unsigned char estado = 0;   //'0' retardo lento, '1' retardo rapido

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //configuracion de las E/S
    TRISD = 0x80;       //RD(6:0) como salidas
    ANSELD = 0x80;      //RD(6:0) como digitales
    TRISBbits.TRISB0 = 1;   //RB0 como entrada
    TRISBbits.TRISB1 = 1;   //RB1 como entrada
    ANSELBbits.ANSELB0 = 0; //RB0 como digital
    ANSELBbits.ANSELB1 = 0; //RB1 como digital
    WPUBbits.WPUB0 = 1;     //RB0 pullup enabled
    WPUBbits.WPUB1 = 1;     //RB1 pullup enabled
    //configuracion de la INT0
    INT0PPS = 0x08;         //asignando la INT0 para que entre por RB0
    PIE1bits.INT0IE = 1;        //INT0 enabled
    PIR1bits.INT0IF = 0;        //bajamos inicialmente la bandera
    INTCON0bits.INT0EDG = 0;    //INT0 en flanco negativo
    INTCON0bits.GIE = 1;        //interruptor global enabled
}
```

```c
void main(void) {
    configuro();
    while(1){
        for(x_var=0;x_var<20;x_var++){
        LATD = animacion[x_var];
        switch(estado){
            case 0:
                __delay_ms(400);
                break;
            case 1:
                __delay_ms(200);
                break;
            case 2:
                __delay_ms(100);
                break;
            case 3:
                __delay_ms(50);
                break;
            }
        }
    }
}
```

```c
//funcion de la INT0
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;        //bajamos la bandera de INT0
    switch(estado){
        case 0:
            estado = 1;
            break;
        case 1:
            estado = 2;
            break;
        case 2:
            estado = 3;
            break;
        case 3:
            estado = 0;
            break;
        default:
            estado = 0;
    }
}
```

Ejemplo de aplicación 2025-1

- Prender y apagar una letra "A" en el display de siete segmentos con un retardo de 500ms

Ejemplo de aplicación 2025-1

- Prender y apagar una letra "A" en el display de siete segmentos con un retardo de 500ms

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;      //NOSC HFINTOSC, NDIV 1:1
    OSCFRQ = 0x02;       //HFINTOSC a 4MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //configuracion de las E/S
    TRISD = 0x80;        //RD(6:0) como salidas
    ANSELD = 0x80;       //RD(6:0) como digitales
    TRISBbits.TRISB0 = 1;   //RB0 como entrada
    ANSELBbits.ANSELB0 = 0; //RB0 como digital
    WPUBbits.WPUB0 = 1;     //RB0 con pullup habilitado
    TRISBbits.TRISB1 = 1;   //RB1 como entrada
    ANSELBbits.ANSELB1 = 0; //RB1 como digital
    WPUBbits.WPUB1 = 1;     //RB1 con pullup habilitado
}
```

```c
void main(void) {
    configuro();
//    LATDbits.LATD3 = 0;
    while(1){
        LATD = 0x77;
        __delay_ms(500);
        LATD = 0x00;
        __delay_ms(500);
//        LATDbits.LATD0 = 1;
//        LATDbits.LATD1 = 1;
//        LATDbits.LATD2 = 1;
//        LATDbits.LATD4 = 1;
//        LATDbits.LATD5 = 1;
//        LATDbits.LATD6 = 1;
//        __delay_ms(500);
//        LATDbits.LATD0 = 0;
//        LATDbits.LATD1 = 0;
//        LATDbits.LATD2 = 0;
//        LATDbits.LATD4 = 0;
//        LATDbits.LATD5 = 0;
//        LATDbits.LATD6 = 0;
//        __delay_ms(500);
    }
}
```

Ejemplo de aplicación 2025-1

- Controlar el encendido del display con las interrupciones externas INT0 e INT1

Ejemplo de aplicación 2025-1

- Controlar el encendido del display con las interrupciones externas INT0 e INT1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;      //NOSC HFINTOSC, NDIV 1:1
    OSCFRQ = 0x02;       //HFINTOSC a 4MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //configuracion de las E/S
    TRISD = 0x80;        //RD(6:0) como salidas
    ANSELD = 0x80;       //RD(6:0) como digitales
    TRISBbits.TRISB0 = 1;   //RB0 como entrada
    ANSELBbits.ANSELB0 = 0; //RB0 como digital
    WPUBbits.WPUB0 = 1;     //RB0 con pullup habilitado
    TRISBbits.TRISB1 = 1;   //RB1 como entrada
    ANSELBbits.ANSELB1 = 0; //RB1 como digital
    WPUBbits.WPUB1 = 1;     //RB1 con pullup habilitado
    //configuracion de las interrupciones INT0 e INT1
    INT0PPS = 0x08;          //INT0 asignado a RB0
    INT1PPS = 0x09;          //INT1 asignado a RB1
    PIE1bits.INT0IE = 1;     //INT0 enabled
    PIE6bits.INT1IE = 1;     //INT1 enabled
    INTCON0bits.INT0EDG = 0;    //INT0 flanco negativo
    INTCON0bits.INT1EDG = 0;    //INT1 flanco negativo
    INTCON0bits.GIE = 1;        //GIE habilitado
}
```

```c
void main(void) {
    configuro();
    while(1);
}

//zona de las funciones de interrupcion
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;         //bajamos la bandera INT0IF
    LATD = 0x77;
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;         //bajamos la bandera INT1IF
    LATD = 0x00;
}
```

Ejemplo de aplicación 2025-1

- Visualiza FAIL y bica con opcion de cambio de mensaje con INT0 y cambio de velocidad con INT1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //configuracion de las E/S
    TRISD = 0x80;
    ANSELD = 0x80;
    TRISBbits.TRISB0 = 1;
    TRISBbits.TRISB1 = 1;
    ANSELBbits.ANSELB0 = 0;
    ANSELBbits.ANSELB1 = 0;
    WPUBbits.WPUB0 = 1;
    WPUBbits.WPUB1 = 1;
    //configuracion de las interrupciones INT0 e INT1
    INT0PPS = 0x08;     //asignar INT0 en RB0
    INT1PPS = 0x09;     //asignar INT1 en RB1
    PIE1bits.INT0IE = 1;    //INT0 activada
    PIE6bits.INT1IE = 1;    //INT1  activada
    INTCON0bits.INT0EDG = 0;    //INT0 deteccion flanco negativo
    INTCON0bits.INT1EDG = 0;    //INT1 deteccion flanco descende
    INTCON0bits.GIE = 1;        //habilitador global activado
}

unsigned char mensaje1[]={0x71, 0x77, 0x30, 0x38, 0x00};  //FAIL
unsigned char mensaje2[]={0x7C, 0x10, 0x58, 0x5F, 0x00};  //bica
unsigned char visual = 0;   //'0' para mensaje1 y '1' para mensaje2
unsigned char velocidad = 0;    //'0' para lento, '1' para rapido
```

```c
void main(void) {
    configuro();
    while(1){
        if(visual == 0){
            unsigned char x_var;
            for(x_var=0;x_var<5;x_var++){
                LATD = mensaje1[x_var];
                if(velocidad == 0){
                    __delay_ms(500);
                }
                else{
                    __delay_ms(200);
                }
            }
        }
        else{
            unsigned char x_var;
            for(x_var=0;x_var<5;x_var++){
                LATD = mensaje2[x_var];
                if(velocidad == 0){
                    __delay_ms(500);
                }
                else{
                    __delay_ms(200);
                }
            }
        }
    }
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //se baja la bandera INT0IF
    if(visual == 0){
        visual = 1;
    }
    else{
        visual = 0;
    }
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //se baja la bandera INT0IF
    if(velocidad == 0){
        velocidad = 1;
    }
    else{
        velocidad = 0;
    }
}
```

Ejemplo de aplicación 2025-1

- Contador 0-9 donde cuenta inicial es 0, INT0 es para decrementar cuenta e INT1 para incrementar cuenta, cuando cuenta llega a los extremos no se desborda

Ejemplo de aplicación 2025-1

- Contador 0-9 donde cuenta inicial es 0, INT0 es para decrementar cuenta e INT1 para incrementar cuenta, cuando cuenta llega a los extremos no se desborda

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //configuracion de las E/S
    TRISD = 0x80;        //RD6 al RD0 como salidas
    ANSELD = 0x80;       //RD6 al RD0 como digitales
    TRISBbits.TRISB0 = 1;
    TRISBbits.TRISB1 = 1;
    ANSELBbits.ANSELB0 = 0;
    ANSELBbits.ANSELB1 = 0;
    WPUBbits.WPUB0 = 1;
    WPUBbits.WPUB1 = 1;
    //configuracion de las INT0 e INT1
    INT0PPS = 0x08;          //INT0 asignado a RB0
    INT1PPS = 0x09;          //INT1 asignado a RB1
    PIE1bits.INT0IE = 1;     //INT0 enabled
    PIE6bits.INT1IE = 1;     //INT1 enabled
    PIR1bits.INT0IF = 0;     //INT0 flag down
    PIR6bits.INT1IF = 1;     //INT1 flag down
    INTCON0bits.INT0EDG = 0;     //INT0 falling edge detect
    INTCON0bits.INT1EDG = 0;     //INT1 falling edge detect
    INTCON0bits.GIE = 1;         //Global interrupt enabled
}
```

```c
const unsigned char tableta[]={0x3F, 0x06, 0x5B, 0x4F, 0x66,
                               0x6D, 0x7D, 0x07, 0x7F, 0x67};
unsigned char numero = 0;

void main(void) {
    configuro();
    while(1){
        LATD = tableta[numero];
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;
    if(numero == 0){
        numero = 0;
    }
    else{
        numero--;
    }
    __delay_ms(50);
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;
    if(numero == 9){
        numero = 9;
    }
    else{
        numero++;
    }
    __delay_ms(50);
}
```

Ejemplo de aplicación 2025-1

- Visualizador de mensaje "pregunta luego actua" en un display de siete segmentos del tipo catodo común, INT0 determinará la velocidad de visualización entre 250ms y 500ms

Ejemplo de aplicación 2025-1

- Visualizador de mensaje "pregunta luego actua" en un display de siete segmentos del tipo catodo común. INT0 determinará la velocidad de visualización entre 250ms y 500ms

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL
#define LENTO 0
#define RAPIDO 1

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //configuracion de las E/S
    TRISD = 0x80;
    ANSELD = 0x80;
    TRISB = 0xFF;        //RB0 y RB1 como entradas
    ANSELB = 0xFC;       //RB0 y RB1 como digitales
    WPUB = 0x03;         //RB0 y RB1 con weak pullup
    //configuracion de las interrupciones
    INT0PPS = 0x08;      //INT0 asignado a RB0}
    PIE1bits.INT0IE = 1;     //INT0 enabled
    PIR1bits.INT0IF = 0;     //bajar bandera de INT0
    INTCON0bits.INT0EDG = 0;     //INT0 con deteccion en flanco descendente
    INTCON0bits.GIE = 1;     //global interrupt enabled
}

const unsigned char frase[]={0x73, 0x50, 0x7B, 0x6F, 0x1C, 0x54, 0x78,
                             0x5F, 0x00, 0x38, 0x1C, 0x7B, 0x6F, 0x5C,
                             0x00, 0x5F, 0x58, 0x78, 0x1C, 0x5F, 0x00};
unsigned char veloc = LENTO;
```

```c
void main(void) {
    configuro();
    while(1){
        unsigned char numero;
        for(numero=0;numero<21;numero++){
            LATD = frase[numero];
            switch(veloc){
                case LENTO:
                    __delay_ms(500);
                    break;
                case RAPIDO:
                    __delay_ms(250);
                    break;
            }
        }
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;     //bajar bandera de INT0
    if(veloc == LENTO){
        veloc = RAPIDO;
    }
    else{
        veloc = LENTO;
    }
}
```

Ejemplo de aplicación 2025-1

- Añadiendo al ejercicio anterior la opcion de poner el mensaje de cabeza en el display con INT1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL
#define LENTO 0
#define RAPIDO 1
#define AL_DERECHO 2
#define AL_REVES 3

void configuro(void){
    //configuracion del modulo de oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //configuracion de las E/S
    TRISD = 0x80;
    ANSELD = 0x80;
    TRISB = 0xFF;        //RB0 y RB1 como entradas
    ANSELB = 0xFC;       //RB0 y RB1 como digitales
    WPUB = 0x03;         //RB0 y RB1 con weak pullup
    //configuracion de las interrupciones
    INT0PPS = 0x08;      //INT0 asignado a RB0
    INT1PPS = 0x09;      //INT0 asignado a RB1
    PIE1bits.INT0IE = 1;     //INT0 enabled
    PIE6bits.INT1IE = 1;     //INT1 enabled
    PIR1bits.INT0IF = 0;     //bajar bandera de INT0
    PIR6bits.INT1IF = 0;     //bajar bandera de INT1
    INTCON0bits.INT0EDG = 0;     //INT0 flanco descendente
    INTCON0bits.INT1EDG = 0;     //INT1 flanco descendente
    INTCON0bits.GIE = 1;     //global interrupt enabled
}
```

```c
const unsigned char frase[]={0x73, 0x50, 0x7B, 0x6F, 0x1C,
                             0x54, 0x78, 0x5F, 0x00, 0x38,
                             0x1C, 0x7B, 0x6F, 0x5C, 0x00,
                             0x5F, 0x58, 0x78, 0x1C, 0x5F,
                             0x00};
unsigned char veloc = LENTO;
unsigned char sentido = AL_DERECHO;
unsigned char temporal;
```

```c
void main(void) {
    configuro();
    while(1){
        unsigned char numero;
        for(numero=0;numero<21;numero++){
            switch(sentido){
                case AL_DERECHO:
                    LATD = frase[numero];
                    break;
                case AL_REVES:
                    LATD = ((frase[numero] << 3) & 0x38) +
                           ((frase[numero] >> 3) & 0x07) +
                           (frase[numero] & 0x40);
                    break;
            }
            switch(veloc){
                case LENTO:
                    __delay_ms(500);
                    break;
                case RAPIDO:
                    __delay_ms(250);
                    break;
            }
        }
    }
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;     //bajar bandera de INT0
    if(veloc == LENTO){
        veloc = RAPIDO;
    }
    else{
        veloc = LENTO;
    }
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;     //bajar bandera de INT1
    if(sentido == AL_DERECHO){
        sentido = AL_REVES;
    }
    else{
        sentido = AL_DERECHO;
    }
}
```

Asignación minilab semana 9 NRC9349 2025-1

- Presentación **individual**, tiempo 60 minutos.
- Subir evidencia en la actividad del AV (Unidad 3, semana 9): video de máximo 2 minutos. Debes de presentarte, luego mostrar el código y finalmente hacer las pruebas de la asignación en el circuito implementado.
- Cada minilab representa 2 puntos cada uno que forman parte de la nota de los laboratorios calificados LB3 y DD.
- Con el mismo hardware desarrollar un contador autincremental según lo siguiente:
  - INT0: Intercambia entre cuenta ascendente de números pares y impares
  - INT1: Cambia el tiempo de retardo de la cuenta autoincremental entre 500ms y 100ms

Asignación minilab semana 9 NRC9350 2025-1

- Presentación **individual**, tiempo 60 minutos.
- Cada minilab representa 2 puntos cada uno y forman parte de la nota de los laboratorios calificados LB3 y DD.
- Subir evidencia en la actividad del AV (Unidad 3, semana 9): video de máximo 2 minutos. Debes de presentarte, luego mostrar el código y finalmente hacer las pruebas de la asignación en el circuito implementado.
- Con el mismo hardware modificar el último ejemplo del gusanito para que tenga las siguientes funcionalidades y teniendo un retardo fijo de 100ms para el desplazamiento del gusanito
  - INT0: El gusanito cambia de tamaño de entre un segmento y dos segmentos de largo
  - INT1: Cambia el sentido del movimiento del gusanito entre horario y anti-horario

Asignación minilab semana 9 NRC9353 2025-1

- Presentación **individual**, tiempo 60 minutos.
- Cada minilab representa 2 puntos cada uno y forman parte de la nota de los laboratorios calificados LB3 y DD.
- Subir evidencia en la actividad del AV (Unidad 3, semana 9): video de máximo 2 minutos. Debes de presentarte, luego mostrar el código y finalmente hacer las pruebas de la asignación en el circuito implementado.
- Agregar al circuito dos LEDs como aparecen en la imagen y desarrollar un contador autoincremental 0-9 donde INT0 hará que la cuenta sea ascendente e INT1 cambiará a cuenta descendente, LED en RA0 se encenderá solo cuando la cuenta sea 4 y LED en RA1 se encenderá solo cuando sea 8 la cuenta.

Asignación minilab semana 9 NRC9354 2025-1

- Presentación **individual**, tiempo 60 minutos.
- Cada minilab representa 2 puntos cada uno y forman parte de la nota de los laboratorios calificados LB3 y DD.
- Subir evidencia en la actividad del AV (Unidad 3, semana 9): video de máximo 2 minutos. Debes de presentarte, luego mostrar el código y finalmente hacer las pruebas de la asignación en el circuito implementado.
- Agregar al circuito dos LEDs como aparecen en la imagen y desarrollar un visualizador de la palabra "robotica", se tendrá cinco velocidades de visualizacion y configuradas segun: INT0 aumenta la velocidad, INT1 reduce la velocidad. De llegarse a los límites deberá de mantenerse o en el valor mínimo o en el valor máximo. LED en RA0 será para cuando esté en velocidad máxima y RA1 será para cuando esté en velocidad mínima.

Asignación minilab semana 9 NRC9357 2025-1

- Presentación **individual**, tiempo 60 minutos.
- Cada minilab representa 2 puntos cada uno y forman parte de la nota de los laboratorios calificados LB3 y DD.
- Subir evidencia en la actividad del AV (Unidad 3, semana 9): video de máximo 2 minutos. Debes de presentarte, luego mostrar el código y finalmente hacer las pruebas de la asignación en el circuito implementado.
- Agregar al circuito dos LEDs como aparecen en la imagen y desarrollar un visualizador de la palabra "simulacro de sismo" a razón de una letra a la vez con periodo de cambio inicial de 200ms, se tendrá cinco velocidades de visualizacion (50ms, 100ms, 200ms, 500ms, 1000ms) y configuradas segun: INT0 aumenta la velocidad, INT1 reduce la velocidad. De llegarse a los límites deberá de mantenerse o en el valor mínimo o en el valor máximo. LED en RA0 será para cuando esté en velocidad máxima y RA1 será para cuando esté en velocidad mínima.

Asignación minilab semana 9 NRC9360 2025-1

- Presentación **individual**, tiempo 60 minutos.
- Cada minilab representa 2 puntos cada uno y forman parte de la nota de los laboratorios calificados LB3 y DD.
- Subir evidencia en la actividad del AV (Unidad 3, semana 9): video de máximo 2 minutos. Debes de presentarte, luego mostrar el código y finalmente hacer las pruebas de la asignación en el circuito implementado.
- Agregar al circuito dos LEDs como aparecen en la imagen y desarrollar un visualizador de la palabra "ANTIRETIRO", se tendrá siete velocidades de visualizacion y configuradas segun: INT0 aumenta la velocidad, INT1 reduce la velocidad. De llegarse a los límites deberá de mantenerse o en el valor mínimo o en el valor máximo. LED en RA0 será para cuando esté en velocidad máxima y RA1 será para cuando esté en velocidad mínima.

Ejemplo 2025-2 NRC8437

- Del diagrama de bloques presentado, implementar el circuito electrónico, la elección de los puertos de E/S corre por parte del estudiante

Ejemplo 2025-2 NRC8437

Conexiones:

- Display de siete segmentos conectado a RD(6:0)
- Boton1 a RB1, botón2 a RB2
- LED en RE0

Ejemplo 2025-2 NRC8437

Plantilla inicial:

Como se puede apreciar, es muy similar a lo que se hizo en Assembler de XC8

```c
#include <xc.h>
#include "cabecera.h"

void configuro(void){
    //config del modulo de oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //config de las E/S
    TRISD = 0x80;        //RD(6:0) salidas
    ANSELD = 0x80;       //RD(6:0) digitales
    TRISEbits.TRISE0 = 0;   //RE0 como salida
    ANSELEbits.ANSELE0 = 0; //RE0 como digital
    //por defecto los puertos son entrada analógica
    ANSELBbits.ANSELB1 = 0; //RB1 como digital
    ANSELBbits.ANSELB2 = 0; //RB2 como digital
    WPUBbits.WPUB1 = 1;     //RB1 pullup activo
    WPUBbits.WPUB2 = 1;     //RB@ pullup activo
}

void main(void) {
    return;
}
```

Ejemplo 2025-2 NRC8437

- Primer ejemplo: Prender y apagar el LED con un periodo de 500ms

```c
7   #include <xc.h>
8   #include "cabecera.h"
9   #define _XTAL_FREQ 4000000UL
10
11  void configuro(void){
12      //config del modulo de oscilador HFINTOSC a 4MHz
13      OSCCON1 = 0x60;
14      OSCFRQ = 0x02;
15      OSCEN = 0x40;
16      //config de las E/S
17      TRISD = 0x80;        //RD(6:0) salidas
18      ANSELD = 0x80;       //RD(6:0) digitales
        TRISEbits.TRISE0 = 0;   //RE0 como salida
        ANSELEbits.ANSELE0 = 0; //RE0 como digital
21      //por defecto los puertos son entrada analógica
        ANSELBbits.ANSELB1 = 0; //RB1 como digital
        ANSELBbits.ANSELB2 = 0; //RB2 como digital
        WPUBbits.WPUB1 = 1;     //RB1 pullup activo
        WPUBbits.WPUB2 = 1;     //RB@ pullup activo
26      LATD = 0x00;
27  }
28
29  void main(void) {
30      configuro();
31      while(1){
            LATEbits.LATE0 = 1;     //RE0 = 1
33          __delay_ms(500);        //retardo de 500ms
            LATEbits.LATE0 = 0;     //RE0 = 0
35          __delay_ms(500);        //retardo de 500ms
36      }
37  }
```

Ejemplo 2025-2 NRC8437

- Segundo ejemplo: Añadir funcionalidad
  - BTN1 (RB2) control de acción – parpadeo y apagado
  - BTN2 (RB1) control de velocidad de parpadeo – lento o rápido
- Dichas funcionalidades deben de estar contempladas como interrupciones externas
  - INT0 -> RB1
  - INT1 -> RB2

Ejemplo 2025-2 NRC8437

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL
#define PARPADEANDO 0
#define APAGADO 1
#define LENTO 2
#define RAPIDO 3

unsigned char velocidad = LENTO;
unsigned char estado = APAGADO;

void configuro(void){
    //config del modulo de oscilador HFINTOSC a 4MHz
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //config de las E/S
    TRISD = 0x80;       //RD(6:0) salidas
    ANSELD = 0x80;      //RD(6:0) digitales
    TRISEbits.TRISE0 = 0;   //RE0 como salida
    ANSELEbits.ANSELE0 = 0; //RE0 como digital
    //por defecto los puertos son entrada analógica
    ANSELBbits.ANSELB1 = 0; //RB1 como digital
    ANSELBbits.ANSELB2 = 0; //RB2 como digital
    WPUBbits.WPUB1 = 1;     //RB1 pullup activo
    WPUBbits.WPUB2 = 1;     //RB@ pullup activo
    //config las interrupciones
    INT0PPS = 0x09;     //INT0 esta conectado a RB1
    INT1PPS = 0x0A;     //INT1 esta conectado a RB2
    PIE1bits.INT0IE = 1;    //INT0IE = 1
    PIE6bits.INT1IE = 1;    //INT1IE = 1
    PIR1bits.INT0IF = 0;    //bandera INT0 bajada
    PIR6bits.INT1IF = 0;    //bandera INT1 bajada
    INTCON0bits.INT0EDG = 0;    //deteccion de INT0 en flan
    INTCON0bits.INT1EDG = 1;    //deteccion de INT1 en flan
    INTCON0bits.GIE = 1;    //GIE = 1
    //condiciones iniciales
    LATD = 0x00;
}
```

```c
void main(void) {
    configuro();
    while(1){
        switch(estado){
            case APAGADO:
                LATEbits.LATE0 = 0;
                break;
            case PARPADEANDO:
                LATEbits.LATE0 = 1;
                if(velocidad == LENTO){
                    __delay_ms(500);
                }
                else if(velocidad == RAPIDO){
                    __delay_ms(200);
                }
                LATEbits.LATE0 = 0;
                if(velocidad == LENTO){
                    __delay_ms(500);
                }
                else if(velocidad == RAPIDO){
                    __delay_ms(200);
                }
                break;
        }
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos la bandera INT0IF
    if(velocidad == LENTO){
        velocidad = RAPIDO;
    }
    else if(velocidad == RAPIDO){
        velocidad = LENTO;
    }
}
```

```c
void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajamos la bandera INT1IF
    if(estado == APAGADO){
        estado = PARPADEANDO;
    }
    else if(estado == PARPADEANDO){
        estado = APAGADO;
    }
}
```

Asignación minilab 2025-2 Turno Lunes 13:00

- Modificar el ejemplo desarrollado para que tenga 5 niveles de velocidad y añadir un tercer estado de encendido todo el tiempo.
- Carga de video en la actividad del AV Unidad 3 Semana 9:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 16:45

Ejemplo 2025-2 NRC8438

- Del diagrama de bloques presentado, implementar el circuito electrónico, la elección de los puertos de E/S corre por parte del estudiante

Ejemplo 2025-2 NRC8438

Conexiones:

- Display de siete segmentos conectado a RD(6:0)
- Boton1 a RB2, botón2 a RB3
- LED en RF1

Ejemplo 2025-2 NRC8438

Segundo ejemplo:

Contador 0-9 autoincremental (que va a contar solo con un periodo de cambio de 500ms)

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

const unsigned char tbl_nums_7seg[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,
                                       0x6D, 0x7D, 0x07, 0x7F, 0x67};

void configuro(void){
    //conf el modulo de oscilador
    OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV 1:1
    OSCFRQ = 0x02;       //HFINTOSC a 4MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //conf las E/S
    TRISD = 0x80;        //RD6 al RD0 son salidas
    ANSELD = 0x80;       //RD6 al RD0 como digitales
    TRISFbits.TRISF1 = 0;   //RF1 como salida
    ANSELFbits.ANSELF1 = 0; //RF1 como digital
    //por defecto los puertos son entradas analógicas
    ANSELBbits.ANSELB2 = 0; //RB2 como digital
    ANSELBbits.ANSELB3 = 0; //RB3 como digital
    WPUBbits.WPUB2 = 1;     //RB2 con pullup
    WPUBbits.WPUB3 = 1;     //RB3 con pullup
    //condicionales de operacion
    LATFbits.LATF1 = 0;
}

void main(void) {
    configuro();
    while(1){
        unsigned char x_var;
        for(x_var=0;x_var<10;x_var++){
            LATD = tbl_nums_7seg[x_var];
            __delay_ms(500);
        }
    }
}
```

Ejemplo 2025-2 NRC8438

Modificando el ejemplo, el primer pulsador que cambie de incremento a decremento (y viceversa), el segundo botón que cambie la velocidad (lento-rápido)

Ejemplo 2025-2 NRC8438

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

#define PA_ARRIBA 0
#define PA_ABAJO 1
#define RAPIDO 2
#define LENTO 3

unsigned char sentido = PA_ARRIBA;
unsigned char velocidad = LENTO;
unsigned char cuenta=0;

const unsigned char tbl_nums_7seg[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,
                                       0x6D, 0x7D, 0x07, 0x7F, 0x67};

void configuro(void){
    //conf el modulo de oscilador
    OSCCON1 = 0x60;     //NOSC=HFINTOSC, NDIV 1:1
    OSCFRQ = 0x02;      //HFINTOSC a 4MHz
    OSCEN = 0x40;       //HFINTOSC enabled
    //conf las E/S
    TRISD = 0x80;       //RD6 al RD0 son salidas
    ANSELD = 0x80;      //RD6 al RD0 como digitales
    TRISFbits.TRISF1 = 0;   //RF1 como salida
    ANSELFbits.ANSELF1 = 0; //RF1 como digital
    //por defecto los puertos son entradas analógicas
    ANSELBbits.ANSELB2 = 0; //RB2 como digital
    ANSELBbits.ANSELB3 = 0; //RB3 como digital
    WPUBbits.WPUB2 = 1;     //RB2 con pullup
    WPUBbits.WPUB3 = 1;     //RB3 con pullup
    //conf las interrupciones
    INT0PPS = 0x0A;         //INT0 conectado a RB2
    INT1PPS = 0x0B;         //INT1 conectado a RB3
    PIE1bits.INT0IE = 1;    //INT0 enabled
    PIE6bits.INT1IE = 1;    //INT1 enabled
    INTCON0bits.INT0EDG = 0;    //INT0 flanco negativo
    INTCON0bits.INT1EDG = 0;    //INT1 flanco negativo
    INTCON0bits.GIE = 1;    //GIE=1
    //condicionales de operacion
    LATFbits.LATF1 = 0;
}
```

```c
void main(void) {
    configuro();
    while(1){
        if(sentido == PA_ARRIBA){
            if(cuenta == 9){
                cuenta = 0;
            }
            else{
                cuenta++;
            }
        }
        else if(sentido == PA_ABAJO){
            if(cuenta == 0){
                cuenta = 9;
            }
            else{
                cuenta--;
            }
        }
        LATD = tbl_nums_7seg[cuenta];
        if(velocidad == LENTO){
            __delay_ms(500);
        }
        else if(velocidad == RAPIDO){
            __delay_ms(200);
        }
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos bandera INT0IF
    if(sentido == PA_ARRIBA){
        sentido = PA_ABAJO;
    }
    else if(sentido == PA_ABAJO){
        sentido = PA_ARRIBA;
    }
}
```

```c
void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajamos bandera INT1IF
    if(velocidad == LENTO){
        velocidad = RAPIDO;
    }
    else if(velocidad == RAPIDO){
        velocidad = LENTO;
    }
}

void __interrupt(irq(default)) DEFAULT_ISR(void){

}
```

Asignación minilab 2025-2 Turno Lunes 17:00

- Modificar el ejemplo desarrollado para que tenga 5 niveles de velocidad.
- Carga de video en la actividad del AV Unidad 3 Semana 9:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 20:45

Ejemplo 2025-2 NRC 8445

- La aplicación será la de un contador binario con visualización en el display de siete segmentos del tipo cátodo común (0 al F) donde BTN1 hará que incremente la cuenta de uno en uno y el BTN2 hará que decremente la cuenta de uno en uno.
- Tener en cuenta usar interrupciones!

Ejemplo 2025-2 NRC 8445

- Prueba inicial: decodificación binaria hacia el display de siete segmentos, se visualizará el dígito uno.
- Tener en cuenta que RE0 debe de ser uno para que se encienda el display.

```c
#include <xc.h>
#include "cabecera.h"

const unsigned char deco_bin[] = {0x3F, 0x06, 0x5B, 0x4F,
                                  0x66, 0x6D, 0x7D, 0x67,
                                  0x7F, 0x6F, 0x5F, 0x7C,
                                  0x58, 0x5E, 0x7B, 0x71};

void configuro(void){
    //conf modulo de oscilador
    OSCCON1 = 0x60;      //NOSC HFINTOSC NDIV 1:1
    OSCFRQ = 0x02;       //HFINTOSC a 4MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //conf de las E/S
    TRISD = 0x80;        //RD(6:0) como salidas
    ANSELD = 0x80;       //RD(6:0) como digitales
    TRISFbits.TRISF0 = 0;   //RF0 como salida
    ANSELFbits.ANSELF0 = 0; //RF0 como digital
    TRISEbits.TRISE0 = 0;   //RE0 como salida
    ANSELEbits.ANSELE0 = 0; //RE0 como digital
    ANSELB = 0xF3;       //RB2 y RB3 como digitales
    WPUB = 0x0C;         //RB2 y RB3 con pullup
    //condiciones iniciales
    LATEbits.LATE0 = 1; //encendemos el display de 7seg
}

void main(void) {
    configuro();
    while(1){
        LATD = deco_bin[1];
    }
}
```

Ejemplo 2025-2 NRC 8446

- Si se usa interrupciones me doy cuenta que:

Table 21-1. PPS Input Selection Table

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	Available Input Port												
				28-Pin Devices			40-Pin Devices					48-Pin Devices				
Interrupt 0	INT0PPS	RB0	'b001 000	A	B	—	A	B	—	—	—	A	B	—	—	—	—
Interrupt 1	INT1PPS	RB1	'b001 001	A	B	—	A	B	—	—	—	—	B	—	D	—	—
Interrupt 2	INT2PPS	RB2	'b001 010	A	B	—	A	B	—	—	—	—	B	—	—	—	F

- Tengo que cambiar la asignacion de puertos de las interrupciones externas para cumplir con el circuito anterior:
  - BTN1 esta en RB2
  - BTN2 esta en RB3

Ejemplo 2025-2 NRC 8446

- A continuacion los registros PIE donde se encuentran los habilitadores de las fuentes de interrupción:

0x049E	PIE0	7:0	IOCIE		CLC1IE		CSWIE	OSFIE	HLVDIE	SWIE
0x049F	PIE1	7:0	SMT1PWAIE	SMT1PRAIE	SMT1IE	CM1IE	ACTIE	ADIE	ZCDIE	INT0IE
0x04A0	PIE2	7:0	DMA1AIE	DMA1ORIE	DMA1DCNTIE	DMA1SCNTIE				ADTIE
0x04A1	PIE3	7:0	TMR0IE	CCP1IE	TMR1GIE	TMR1IE	TMR2IE	SPI1IE	SPI1TXIE	SPI1RXIE
0x04A2	PIE4	7:0	PWM1IE	PWM1PIE			U1IE	U1EIE	U1TXIE	U1RXIE
0x04A3	PIE5	7:0	PWM2IE	PWM2PIE	TMR3GIE	TMR3IE		SPI2IE	SPI2TXIE	SPI2RXIE
0x04A4	PIE6	7:0	DMA2AIE	DMA2ORIE	DMA2DCNTIE	DMA2SCNTIE	NCO1IE	CWG1IE	CLC2IE	INT1IE
0x04A5	PIE7	7:0	PWM3IE	PWM3PIE	CLC3IE		I2C1EIE	I2C1IE	I2C1TXIE	I2C1RXIE
0x04A6	PIE8	7:0	SCANIE	CCP2IE	TMR5GIE	TMR5IE	U2IE	U2EIE	U2TXIE	U2RXIE
0x04A7	PIE9	7:0			CLC4IE		U3IE	U3EIE	U3TXIE	U3RXIE
0x04A8	PIE10	7:0	DMA3AIE	DMA3ORIE	DMA3DCNTIE	DMA3SCNTIE	NCO2IE	CWG2IE	CLC5IE	INT2IE
0x04A9	PIE11	7:0	DMA4AIE	DMA4ORIE	DMA4DCNTIE	DMA4SCNTIE	TMR4IE	CWG3IE	CLC6IE	CCP3IE
0x04AA	PIE12	7:0	DMA5AIE	DMA5ORIE	DMA5DCNTIE	DMA5SCNTIE	U4IE	U4EIE	U4TXIE	U4RXIE
0x04AB	PIE13	7:0	DMA6AIE	DMA6ORIE	DMA6DCNTIE	DMA6SCNTIE	U5IE	U5EIE	U5TXIE	U5RXIE
0x04AC	PIE14	7:0					NCO3IE	CM2IE	CLC7IE	
0x04AD	PIE15	7:0					TMR6IE	CRCIE	CLC8IE	NVMIE

0x04D6	INTCON0	7:0	GIE/GIEH	GIEL	IPEN			INT2EDG	INT1EDG	INT0EDG
0x04D7	INTCON1	7:0	STAT[1:0]							

Ejemplo 2025-2 NRC 8445

- Contador 0-F donde los pulsadores BTN1 y BTN2 funcionan con interrupciones externas.

```c
#include <xc.h>
#include "cabecera.h"

const unsigned char deco_bin[] = {0x3F, 0x06, 0x5B, 0x4F,
                                  0x66, 0x6D, 0x7D, 0x67,
                                  0x7F, 0x6F, 0x5F, 0x7C,
                                  0x58, 0x5E, 0x7B, 0x71};

unsigned char cuenta_actual = 0;

void configuro(void){
    //conf modulo de oscilador
    OSCCON1 = 0x60;     //NOSC HFINTOSC NDIV 1:1
    OSCFRQ = 0x02;      //HFINTOSC a 4MHz
    OSCEN = 0x40;       //HFINTOSC enabled
    //conf de las E/S
    TRISD = 0x80;       //RD(6:0) como salidas
    ANSELD = 0x80;      //RD(6:0) como digitales
    TRISFbits.TRISF0 = 0;   //RF0 como salida
    ANSELFbits.ANSELF0 = 0; //RF0 como digital
    TRISEbits.TRISE0 = 0;   //RE0 como salida
    ANSELEbits.ANSELE0 = 0; //RE0 como digital
    ANSELB = 0xF3;      //RB2 y RB3 como digitales
    WPUB = 0x0C;        //RB2 y RB3 con pullup
    //conf de las interrupciones
    INT0PPS = 0x0A;     //la INT0 entra por RB2
    INT1PPS = 0x0B;     //la INT1 entra por RB3
    PIE1bits.INT0IE = 1;    //INT0 enabled
    PIE6bits.INT1IE = 1;    //INT1 enabled
    INTCON0bits.GIE = 1;    //GIE enabled
    INTCON0bits.INT0EDG = 0;    //INT0 falling edge
    INTCON0bits.INT1EDG = 0;    //INT1 falling edge
    //condiciones iniciales
    LATEbits.LATE0 = 1; //encendemos el display de 7seg
}
```

```c
void main(void) {
    configuro();
    while(1){
        LATD = deco_bin[cuenta_actual];
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos la bandera INT0IF
    if(cuenta_actual == 15){
        cuenta_actual = 0;
    }
    else{
        cuenta_actual++;
    }
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajamos la bandera INT0IF
    if(cuenta_actual == 0){
        cuenta_actual = 15;
    }
    else{
        cuenta_actual--;
    }
}

void __interrupt(irq(default)) DEFAULT_ISR(void){
}
```

Ejemplo 2025-2 NRC 8445

```c
#include ...2 lines
#define _XTAL_FREQ 4000000UL

const unsigned char deco_bin[] = {...4 lines };
unsigned char cuenta_actual = 0;

void configuro(void){
    //conf modulo de oscilador
    OSCCON1 = 0x60;      //NOSC HFINTOSC NDIV 1:1
    OSCFRQ = 0x02;       //HFINTOSC a 4MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    //conf de las E/S
    TRISD = 0x80;        //RD(6:0) como salidas
    ANSELD = 0x80;       //RD(6:0) como digitales
    TRISFbits.TRISF0 = 0;   //RF0 como salida
    ANSELFbits.ANSELF0 = 0; //RF0 como digital
    TRISEbits.TRISE0 = 0;   //RE0 como salida
    ANSELEbits.ANSELE0 = 0; //RE0 como digital
    ANSELB = 0xF3;       //RB2 y RB3 como digitales
    WPUB = 0x0C;         //RB2 y RB3 con pullup
    //conf de las interrupciones
    INT0PPS = 0x0A;      //la INT0 entra por RB2
    INT1PPS = 0x0B;      //la INT1 entra por RB3
    PIE1bits.INT0IE = 1;    //INT0 enabled
    PIE6bits.INT1IE = 1;    //INT1 enabled
    INTCON0bits.GIE = 0;    //GIE disabled
    INTCON0bits.INT0EDG = 0;    //INT0 falling edge
    INTCON0bits.INT1EDG = 0;    //INT1 falling edge
    //condiciones iniciales
    LATEbits.LATE0 = 1; //encendemos el display de 7seg
}

void splash_screen(void){
    unsigned char x_var;
    for(x_var=0;x_var<16;x_var++){
        LATD = deco_bin[x_var];
        __delay_ms(100);
    }
    __delay_ms(3000);
}
```

```c
void splash_screen(void){
    unsigned char x_var;
    for(x_var=0;x_var<16;x_var++){
        LATD = deco_bin[x_var];
        __delay_ms(100);
    }
    __delay_ms(3000);
}

void main(void) {
    configuro();
    splash_screen();
    INTCON0bits.GIE = 1;    //GIE enabled
    while(1){
        LATD = deco_bin[cuenta_actual];
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){...9 lines }

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){...9 lines }

void __interrupt(irq(default)) DEFAULT_ISR(void){
}
```

- Agregando función de splash screen antes del funcionamiento del contador

Asignación minilab 2025-2 Turno Jueves 14:00

- Modificar el ejemplo desarrollado para que:
  - En el splash screen el LED en RF0 parpadee por cada cambio en el display.
  - Al momento de llegar a los extremos NO SE DESBORDE.
- Carga de video en la actividad del AV Unidad 3 Semana 9:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 17:45

Asignación minilab 2025-2 Turno Jueves 18:00

- Modificar el ejemplo desarrollado para que:
  - Al momento de llegar a los extremos NO DEBE DESBORDARSE, se mostrará el símbolo "-" en el display y deberá de parpadear el LED en RF0.
- Carga de video en la actividad del AV Unidad 3 Semana 9:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 21:30

Ejemplo 2025-2 NRC8448

- El presente circuito va a detectar si hay la presencia de un imán y de luz.
  - Si hay imán encenderá LED en RE0
  - Si hay luz encenderá LED en RF0

Ejemplo 2025-2 NRC8448

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

void configuro(void){
    //conf el modulo de oscilador}
    OSCCON1 = 0x60; //NOSC HFINTOSC NDIV 1:1
    OSCFRQ = 0x02;  //HFINTOSC 4MHz
    OSCEN = 0x40;   //HFINTOSC enabled
    //conf las E/S
    TRISD = 0x80;   //RD(6:0) salidas
    ANSELD = 0x80;  //RD(6:0) digitales
    TRISEbits.TRISE0 = 0;   //RE0 salida
    ANSELEbits.ANSELE0 = 0; //RE0 digital
    TRISFbits.TRISF0 = 0;   //RF0 salida
    ANSELFbits.ANSELF0 = 0; //RF0 digital
    ANSELB = 0xC7;  //RB3, RB4, RB5 digitales
    WPUBbits.WPUB5 = 1; //RB5 pullup enabled
}
```

```c
void main(void) {
    configuro();
    while(1){
        //prueba 1: prender un apagar los LEDs
        /*...6 lines */
        //prueba 2: probar los sensores
        if(PORTBbits.RB3 == 1){     //sensor Hall
            LATEbits.LATE0 = 1;
        }
        else{
            LATEbits.LATE0 = 0;
        }
        if(PORTBbits.RB4 == 0){     //sensor LDR
            LATFbits.LATF0 = 1;
        }
        else{
            LATFbits.LATF0 = 0;
        }

    }
}
```

Ejemplo 2025-2 NRC8448

- Ahora hacer un contador binario de 4 bits con visualización 0-F
  - Si hay imán incrementará la cuenta
  - Si hay luz decrementará la cuenta
  - Si presionas BTN1 reinicias la cuenta a cero
- Utilizar interrupciones!

Ejemplo 2025-2 NRC8448

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

const unsigned char efectazo[] = {0x01, 0x02, 0x40, 0x10,
                                  0x08, 0x04, 0x40, 0x20};
const unsigned char deco7s[] = {0x3F, 0x06, 0x5B, 0x4F,
                                0x66, 0x6D, 0x7D, 0x07,
                                0x7F, 0x67, 0x5F, 0x7C,
                                0x58, 0x5E, 0x7B, 0x71};
unsigned char cuenta_actual = 0;

void configuro(void){
    //conf el modulo de oscilador}
    OSCCON1 = 0x60; //NOSC HFINTOSC NDIV 1:1
    OSCFRQ = 0x02;  //HFINTOSC 4MHz
    OSCEN = 0x40;   //HFINTOSC enabled
    //conf las E/S
    TRISD = 0x80;   //RD(6:0) salidas
    ANSELD = 0x80;  //RD(6:0) digitales
    TRISEbits.TRISE0 = 0;   //RE0 salida
    ANSELEbits.ANSELE0 = 0; //RE0 digital
    TRISFbits.TRISF0 = 0;   //RF0 salida
    ANSELFbits.ANSELF0 = 0; //RF0 digital
    ANSELB = 0xC7;  //RB3, RB4, RB5 digitales
    WPUBbits.WPUB5 = 1; //RB5 pullup enabled
    //conf interrupciones
    INT0PPS = 0x0B; //INT0 viene de RB3
    INT1PPS = 0x0C; //INT1 viene de RB4
    INT2PPS = 0x0D; //INT2 viene de RB5
    PIE1bits.INT0IE = 1;    //INT0 enabled
    PIE6bits.INT1IE = 1;    //INT1 enabled
    PIE10bits.INT2IE = 1;   //INT2 enabled
    INTCON0bits.GIE = 1;     //global interrupt enabled
    INTCON0bits.INT1EDG = 0;    //INT1 falling edge
    INTCON0bits.INT2EDG = 0;    //NT2 falling edge
}
```

```c
void pantallazo(void){
    unsigned char x_var,y_var;
    for(y_var=0;y_var<10;y_var++){
        for(x_var=0;x_var<8;x_var++){
            LATD = efectazo[x_var];
            __delay_ms(50);
        }
    }
}

void main(void) {
    configuro();
    pantallazo();
    while(1){
        //prueba 1: prender un apagar los LEDs
        /*...6 lines */
        //prueba 2: probar los sensores
        /*...12 lines */
        LATD = deco7s[cuenta_actual];
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos flag INT0IF
    if(cuenta_actual == 15){
        cuenta_actual = 0;
    }
    else{
        cuenta_actual++;
    }
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajamos flag INT0IF
    if(cuenta_actual == 0){
        cuenta_actual = 15;
    }
    else{
        cuenta_actual--;
    }
}
```

```c
void __interrupt(irq(IRQ_INT2)) INT2_ISR(void){
    PIR10bits.INT2IF = 0;    //bajamos flag INT0IF
    cuenta_actual = 0;
}

void __interrupt(irq(default)) DEFAULT_ISR(void){
}
```

- Si es que hay falsos disparos en los sensores, añadir un retardo de 50ms al final de las funciones de interrupción.

Asignación minilab 2025-2 Turno Viernes 09:00

- Modificar el ejemplo desarrollado para que:
  - Al momento de llegar a los extremos NO DEBE DESBORDARSE.
    - Deberá de parpadear el LED en RF0 cuando se encuentre en 0
    - Deberá de parpadear el LED en RE0 cuando se encuentre en F
- Carga de video en la actividad del AV Unidad 3 Semana 9:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 12:45

Ejemplo 2025-2 NRC 8449

- Efecto "gusanito" en el display de siete segmentos

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

const unsigned char gusanito[] = {0x01, 0x02, 0x40, 0x10,
                                  0x08, 0x04, 0x40, 0x20};

void configuro(void){
    //conf modulo de oscilador
    OSCCON1 = 0x60; //NOSC HFINTOSC NDIV 1:1
    OSCFRQ = 0x02;  //HFINTOSC a 4MHz
    OSCEN = 0x40;   //HFINTOSC enabled
    //conf las E/S
    TRISD = 0x80;   //RD(6:0) salidas
    ANSELD = 0x80;  //RD(6:0) digitales
    TRISFbits.TRISF1 = 0;   //RF1 salida
    ANSELFbits.ANSELF1 = 0; //RF1 digital
    TRISEbits.TRISE0 = 0;   //RE0 salida
    ANSELEbits.ANSELE0 = 0; //RE0 digital
    ANSELB = 0xE3;          //RB2 RB3 RB4 digitales
    WPUBbits.WPUB2 = 1;     //RB2 pullup enabled
    //condiciones iniciales
    LATFbits.LATF1 = 1;     //habilito display7s
}

void anima_gusanito_r(unsigned char veces){
    unsigned char x_var, y_var;
    for(y_var=0;y_var<veces;y_var++){
        for(x_var=0;x_var<8;x_var++){
            LATD = gusanito[x_var];
            __delay_ms(50);
        }
    }
}
```

```c
void anima_gusanito_l(unsigned char veces){
    unsigned char x_var, y_var;
    for(y_var=0;y_var<veces;y_var++){
        for(x_var=0;x_var<8;x_var++){
            LATD = gusanito[7-x_var];
            __delay_ms(50);
        }
    }
}

void main(void) {
    configuro();
    anima_gusanito_r(4);
    anima_gusanito_l(4);
    while(1){
        LATEbits.LATE0 = 1; //LED on
        __delay_ms(300);
        LATEbits.LATE0 = 0; //LED off
        __delay_ms(300);
    }
}
```

Ejemplo 2025-2 NRC8449

- La funcionalidad de visualización del gusanito ahora provienen de interrupciones externas INT0 e INT1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 4000000UL

const unsigned char gusanito[] = {0x01, 0x02, 0x40, 0x10,
                                  0x08, 0x04, 0x40, 0x20};

void configuro(void){
    //conf modulo de oscilador
    OSCCON1 = 0x60; //NOSC HFINTOSC NDIV 1:1
    OSCFRQ = 0x02;  //HFINTOSC a 4MHz
    OSCEN = 0x40;   //HFINTOSC enabled
    //conf las E/S
    TRISD = 0x80;   //RD(6:0) salidas
    ANSELD = 0x80;  //RD(6:0) digitales
    TRISFbits.TRISF1 = 0;   //RF1 salida
    ANSELFbits.ANSELF1 = 0; //RF1 digital
    TRISEbits.TRISE0 = 0;   //RE0 salida
    ANSELEbits.ANSELE0 = 0; //RE0 digital
    ANSELB = 0xE3;          //RB2 RB3 RB4 digitales
    WPUBbits.WPUB2 = 1;     //RB2 pullup enabled
    //condiciones iniciales
    LATFbits.LATF1 = 1;     //habilito display7s
    //conf de las interrupts
    INT0PPS = 0x0B; //INT0 proviene de RB3
    INT1PPS = 0x0C; //INT1 proviene de RB4
    PIE1bits.INT0IE = 1;    //INT0 enabled
    PIE6bits.INT1IE = 1;    //INT1 enabled
    INTCON0bits.GIE = 1;    //Global interrupt enabled
}

void anima_gusanito_r(unsigned char veces){
    unsigned char x_var, y_var;
    for(y_var=0;y_var<veces;y_var++){
        for(x_var=0;x_var<8;x_var++){
            LATD = gusanito[x_var];
            __delay_ms(50);
        }
    }
}

void anima_gusanito_l(unsigned char veces){
    unsigned char x_var, y_var;
    for(y_var=0;y_var<veces;y_var++){
        for(x_var=0;x_var<8;x_var++){
            LATD = gusanito[7-x_var];
            __delay_ms(50);
        }
    }
}
```

```c
void main(void) {
    configuro();
    anima_gusanito_r(4);
    anima_gusanito_l(4);
    while(1){
        LATEbits.LATE0 = 1; //LED on
        __delay_ms(300);
        LATEbits.LATE0 = 0; //LED off
        __delay_ms(300);
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;
    anima_gusanito_l(4);
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;
    anima_gusanito_r(4);
}

void __interrupt(irq(default)) DEFAULT_ISR(void){
}
```

Asignación minilab 2025-2 Turno Viernes 15:00

- Preservar la "animación del gusanito" al inicio y modificar la visualización cuando se activen las INT0 e INT1 según lo siguiente:
  - INT0 disparado: visualización de cuenta 0 al 9 una vez con periodo de cambio 50ms.
  - INT1 disparado: visualización de cuenta 9 al 0 una vez con periodo de cambio 50ms.
- Carga de video en la actividad del AV Unidad 3 Semana 9:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 18:45

Ej. 2025-2 Implementar el siguiente circuito

Ej. 2025-2 Implementar el siguiente circuito

- Verificaciones iniciales de la implementación:
  - Conectar el Circuito a la PC
  - El LED del Curiosity Nano se encuentre encendido de manera continua
  - Ajustar el constraste del LCD para que se vea una barra sólida en la primera línea
  - No olvidar de instalar condensadores en los pulsadores para el antirrebote

Ej. 2025-2 Implementar el siguiente circuito

- No olvidar primero crear el proyecto
- Seguir el procedimiento para descargar la librería para el LCD y cargarlo en el proyecto creado
- No te olvides de configurar el Curiosity Nano para que entregue 5V por el VTG
- **No olvidar de especificar el valor correcto de _XTAL_FREQ en LCD.h**

Ej. 2025-2 Implementar el siguiente circuito

- Todo lo actuado se encuentra en la imagen de la izquierda:
  - Proyecto creado con PIC18F57Q43 corriendo a 4MHz
  - Librería LCD con archivos puestos en la carpeta del proyecto actual
  - Archivos de librería llamados al proyecto en el MPLABX
  - LCD.h llamado en el código fuente
  - Función de inicialización creado en el código fuente
  - Ajustar _XTAL_FREQ del LCD.h a 4MHz
  - Energía del VTG a 5V
  - Funcion de inicialización del LCD llamado en el main()

Ej. 2025-2 Implementar el siguiente circuito

- Visualización inicial de mensajes

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 4000000UL

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
}

void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}
```

```c
void pantallazo(void){
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Hola mundo UPC!",15);
}

void main(void) {
    configuro();
    LCD_init();
    pantallazo();
    while(1){
    }
}
```

- Adicionalmente añadir mensaje en la segunda línea del LCD

Caracteres personalizados

- Hasta 8 caracteres personalizados (0 al 7)
- Link: https://maxpromer.github.io/LCD-Character-Creator/
- Nos interesa solo los códigos generados en hexadecimal
- Seguir el procedimiento que se encuentra en la diapositiva 49

Ej. 2025-2 Implementar el siguiente circuito

- Visualización con caracteres personalizados

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 4000000UL

const unsigned char arbolazo[] = {0x04,0x0E,0x04,0x0E,
                                  0x1F,0x04,0x04,0x1F};
const unsigned char arbolito[] = {0x00,0x00,0x00,0x00,
                                  0x04,0x0E,0x04,0x1F};

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
}

void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    GENERACARACTER(arbolazo, 0);
    GENERACARACTER(arbolito, 1);
}
```

```c
void pantallazo(void){
    POS_CURSOR(1,2);
    ESCRIBE_MENSAJE("+* Merry Xmas *+",16);
    POS_CURSOR(2,0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
}

void main(void) {
    configuro();
    LCD_init();
    pantallazo();
    while(1){
    }
}
```

Ej. 2025-2 Implementar el siguiente circuito

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 4000000UL

const unsigned char arbolazo[] = {0x04,0x0E,0x04,0x0E,
                                  0x1F,0x04,0x04,0x1F};
const unsigned char arbolito[] = {0x00,0x00,0x00,0x00,
                                  0x04,0x0E,0x04,0x1F};

void configuro(void){
    //con del modulo oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //conf de las E/S
    ANSELB = 0xE7;  //RB3 y RB4 como digitales
    WPUB = 0x18;    //RB3 y RB4 pullup enabled
    //conf las INTs
    INT0PPS = 0x0B; //INT0 entra por RB3
    INT1PPS = 0x0C; //INT1 entra por RB4
    PIE1bits.INT0IE = 1;    //INT0 enabled
    PIE6bits.INT1IE = 1;    //INT1 enabled
    INTCON0bits.GIE = 1;    //GIE = 1
    INTCON0bits.INT0EDG = 0;    //INT0 falling edge
    INTCON0bits.INT1EDG = 0;    //INT1 falling edge
}

void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    GENERACARACTER(arbolazo, 0);
    GENERACARACTER(arbolito, 1);
}
```

```c
void pantallazo(void){
    POS_CURSOR(1,2);
    ESCRIBE_MENSAJE("+' Merry Xmas '+",16);
    POS_CURSOR(2,0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
}

void main(void) {
    configuro();
    LCD_init();
    pantallazo();
    while(1){
    }
}

void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos bandera de INT0
    POS_CURSOR(2,0);
    ENVIA_CHAR(0);
    ESCRIBE_MENSAJE("uC class - UPC",14);
    ENVIA_CHAR(0);
}
```

```c
void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajamos bandera de INT1
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Help Lordcchelly",16);
}

void __interrupt(irq(default)) DEFAULT_ISR(void){
}
```

- Visualización con cambio de mensajes usando los pulsadores y en interrupción

Ej. 2025-2 Implementar el siguiente circuito

```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 4000000UL

const unsigned char arbolazo[] = {0x04,0x0E,0x04,0x0E,
                                  0x1F,0x04,0x04,0x1F};
const unsigned char arbolito[] = {0x00,0x00,0x00,0x00,
                                  0x04,0x0E,0x04,0x1F};
unsigned int res_ADC = 0;

void configuro(void){
    //con del modulo oscilador
    OSCCON1 = 0x60;
    OSCFRQ = 0x02;
    OSCEN = 0x40;
    //conf de las E/S
    ANSELB = 0xE7;  //RB3 y RB4 como digitales
    WPUB = 0x18;    //RB3 y RB4 pullup enabled
    //conf las INTs
    INT0PPS = 0x0B; //INT0 entra por RB3
    INT1PPS = 0x0C; //INT1 entra por RB4
    PIE1bits.INT0IE = 1;    //INT0 enabled
    PIE6bits.INT1IE = 1;    //INT1 enabled
    INTCON0bits.GIE = 1;    //GIE = 1
    INTCON0bits.INT0EDG = 0;    //INT0 falling edge
    INTCON0bits.INT1EDG = 0;    //INT1 falling edge
    //conf del ADC
    ADCON0bits.ADFM = 1;    //resultado just derecha
    ADCON0bits.ADCS = 1;    //clk source ADCRC
    ADPCH = 0x00;       //RA0 seleccionado
    ADCON0bits.ADON = 1;    //ADC encendido
}

void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    LCD_CONFIG();
    __delay_ms(19);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    GENERACARACTER(arbolazo, 0);
    GENERACARACTER(arbolito, 1);
}
```

```c
void pantallazo(void){
    POS_CURSOR(1,2);
    ESCRIBE_MENSAJE("+' Merry Xmas '+",16);
    POS_CURSOR(2,0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
    ENVIA_CHAR(1);
    ENVIA_CHAR(0);
}

void main(void) {
    configuro();
    LCD_init();
    pantallazo();
    __delay_ms(3000);
    BORRAR_LCD();
    while(1){
        ADCON0bits.GO_nDONE = 1;    //inicio la conversion el ADC
        while(ADCON0bits.GO_nDONE == 1);    //esperar a que termine
        res_ADC = (ADRESH << 8) + ADRESL;
        INTCON0bits.GIE = 0;
        POS_CURSOR(1,0);
        ESCRIBE_MENSAJE("ADC:", 4);
        ENVIA_CHAR((res_ADC / 1000) + 0x30);    //imprimir millar
        ENVIA_CHAR(((res_ADC % 1000) / 100) + 0x30); //imprimir centena
        ENVIA_CHAR(((res_ADC % 100) / 10) + 0x30);  //imprimir decena
        ENVIA_CHAR((res_ADC % 10) + 0x30); //imprimir unidad
        INTCON0bits.GIE = 1;
    }
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;    //bajamos bandera de INT0
    POS_CURSOR(2,0);
    ENVIA_CHAR(0);
    ESCRIBE_MENSAJE("uC class - UPC",14);
    ENVIA_CHAR(0);
}

void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
    PIR6bits.INT1IF = 0;    //bajamos bandera de INT1
    POS_CURSOR(2,0);
    ESCRIBE_MENSAJE("Help Lordcchelly",16);
}

void __interrupt(irq(default)) DEFAULT_ISR(void){
}
```

- Lectura de potenciómetro en RA0 con el ADC y visualización del resultado en decimal en el LCD y cambio de mensajes con pulsadores en Interrupción

Minilab Sem10 2025-2 Turno Lunes 13:00

- Terminar de desarrollar el presente ejemplo y grabar el video de evidencia.
- Carga de video en la actividad del AV Unidad 3 Semana 10:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 16:45

Minilab Sem10 2025-2 Turno Lunes 17:00

- Terminar de desarrollar el presente ejemplo y grabar el video de evidencia.
- Carga de video en la actividad del AV Unidad 3 Semana 10:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 20:45

Minilab Sem10 2025-2 Turno Jueves 14:00

- Terminar de desarrollar el presente ejemplo y grabar el video de evidencia.
- Carga de video en la actividad del AV Unidad 3 Semana 10:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 17:55

Minilab Sem10 2025-2 Turno Jueves 18:00

- Terminar de desarrollar el presente ejemplo y grabar el video de evidencia.
- Carga de video en la actividad del AV Unidad 3 Semana 10:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 21:00

Minilab Sem10 2025-2 Turno Viernes 09:00

- Terminar de desarrollar el presente ejemplo y grabar el video de evidencia.
- Carga de video en la actividad del AV Unidad 3 Semana 10:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 12:00

Minilab Sem10 2025-2 Turno Viernes 15:00

- Terminar de desarrollar el presente ejemplo y grabar el video de evidencia.
- Carga de video en la actividad del AV Unidad 3 Semana 10:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 18:00

Fin de la sesión