Microcontroladores

Profesor: Kalun José Lau Gan

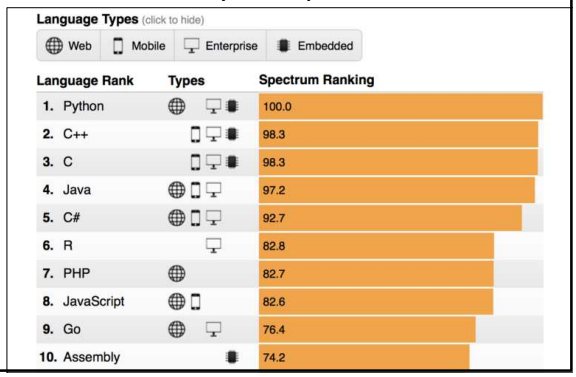
Semestre 2024-1

Semana 9-10

1

¿Preguntas previas?

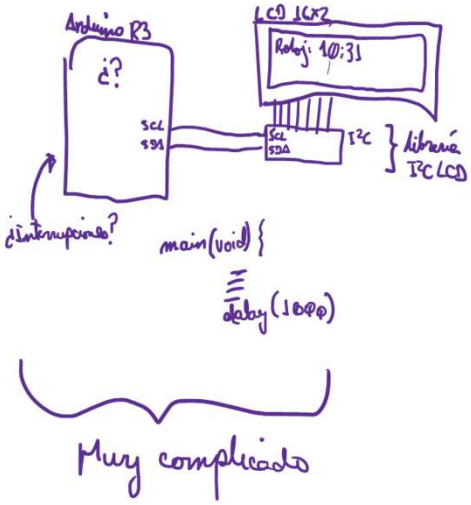
- ¿El hecho de usar C lo hace más fácil?
  - El proceso de diseño es el mismo tanto para lenguaje Assembler como en lenguaje C, se tiene que implementar primero el hardware, desarrollar el algoritmo y finalmente aterrizar el algoritmo en un lenguaje de programación.
  - El lenguaje XC se va a permitir el uso de librerías y funciones especializadas, además de manejar mejor las operaciones matemáticas y manipulación de datos

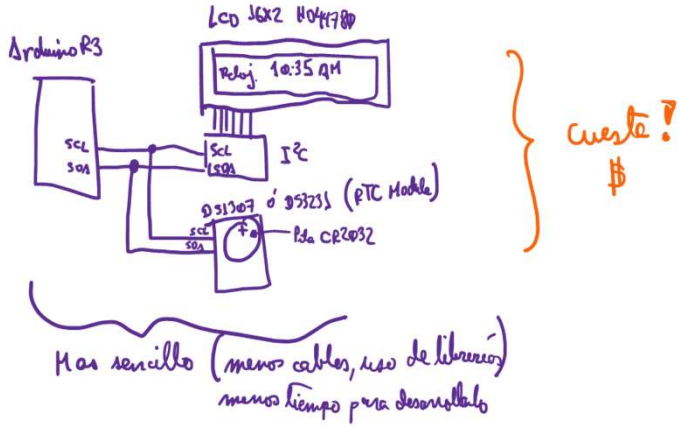


2

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

- El reloj debe de ser preciso





3

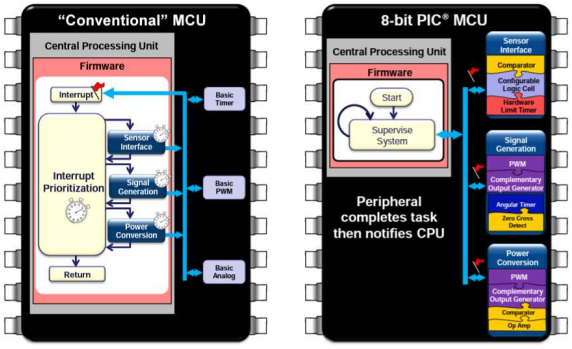
Agenda

- 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
  - MCC
- 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

4

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

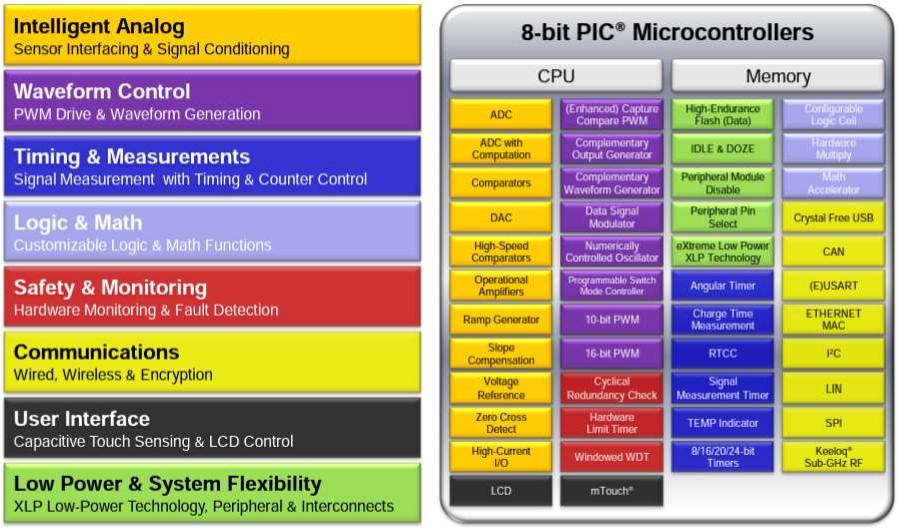
- Independizando el funcionamiento de los periféricos en el microcontrolador se logra un mejor rendimiento del CPU
- Los periféricos independientes administran sus tareas sin código ni supervisión del CPU para mantener su funcionamiento



5

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

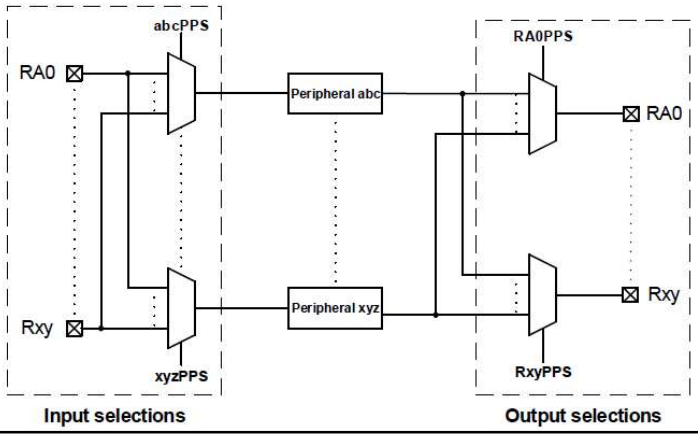
- Dependiendo del modelo y familia el microcontrolador tendrá un conjunto de periféricos CIP
- Los CIPs poseen conexiones adicionales con otros CIP para lograr la independencia del CPU
- Estos CIP se apoyan en el sistema de interconexión de pines (PPS)



6

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



7

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.

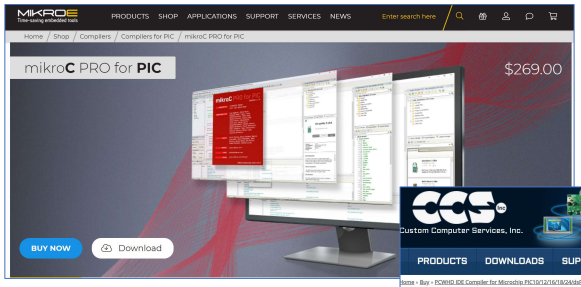
Problema de compatibilidad entre distintas plataformas de desarrollo aún empleando un mismo lenguaje.

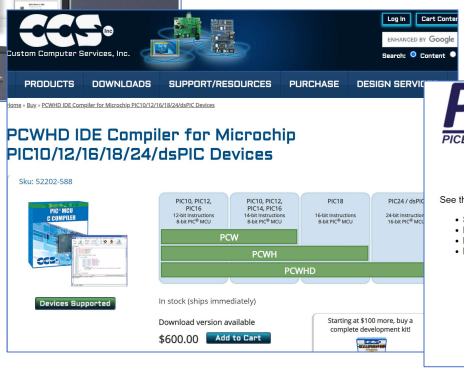
- Basic
- C
- Java (muy poco utilizado y soportado)
- Python (Rpi Pico y el ESP32)

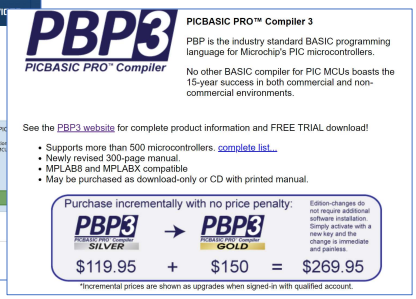
8

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.



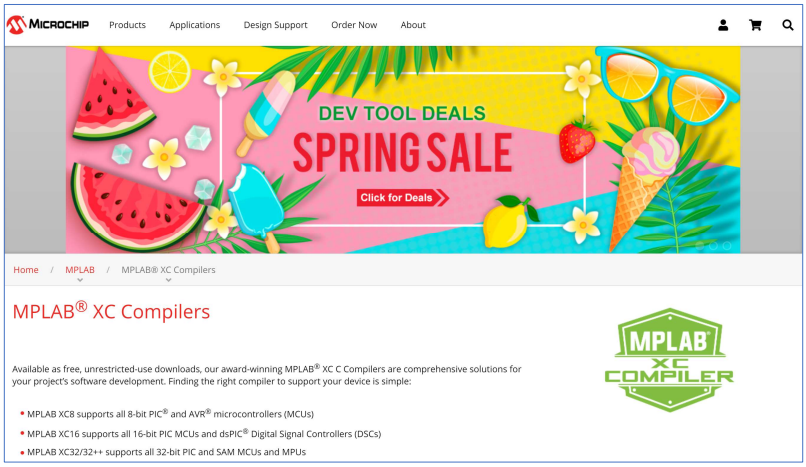




9

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



10

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

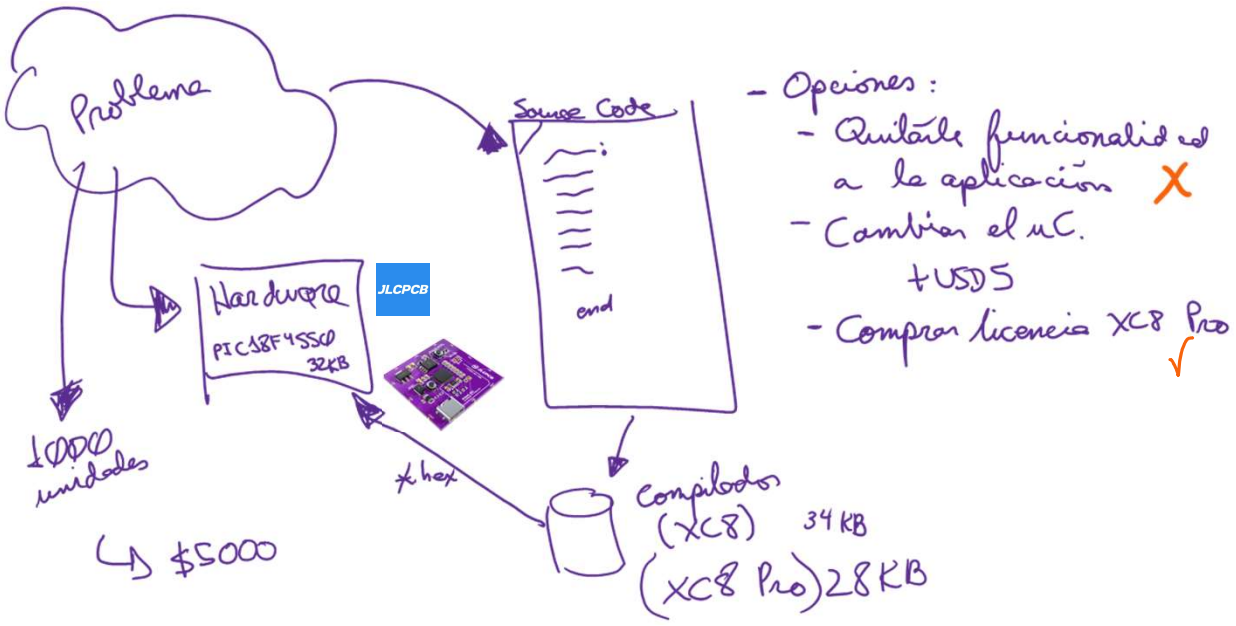
Order now, up to **9** can ship on **20-May-2020**

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

Quantity:

11

Caso:



12

Optimización en el compilador XC

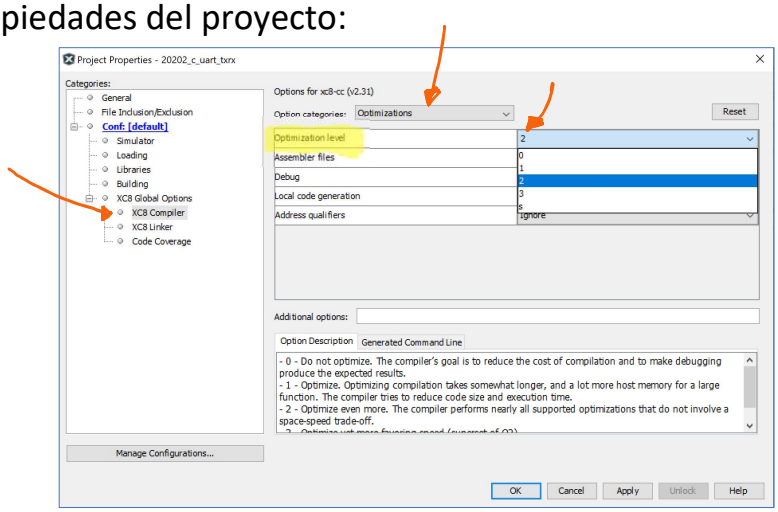
- **Optimization levels:**
  - Level 0: Fastest compilation time, minimal optimizations
  - Level 1: Optimizations with low debugging impact
  - Level 2: All optimizations that give the best balance between speed and size
  - Level 3: Program execution will be as fast as possible
  - Level s: Code size 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

13

Optimización en el compilador XC

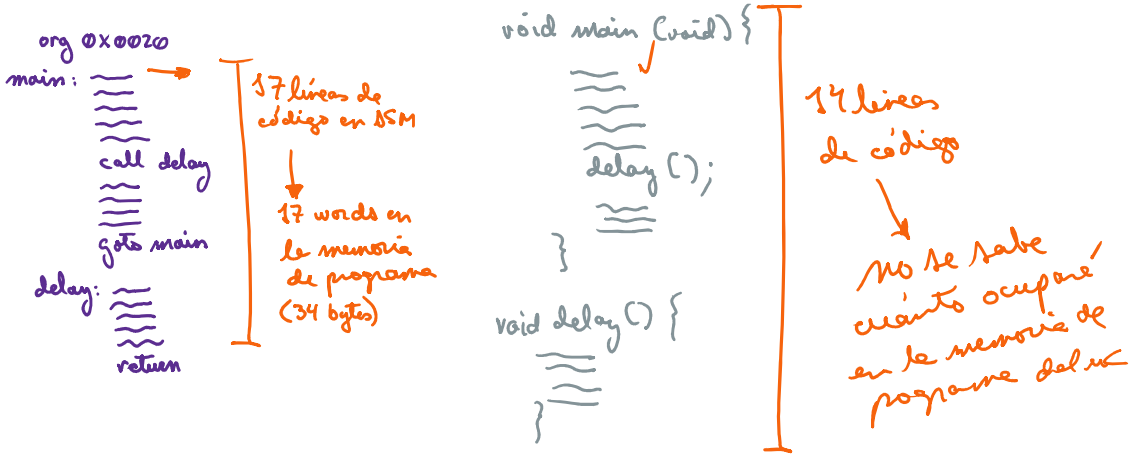
- Para acceder a las opciones de optimización ingresamos a las propiedades del proyecto:



14

Eficiencia de código en Assembler vs C

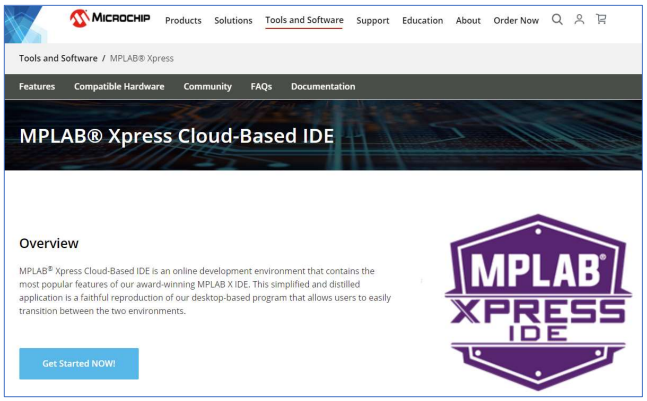
¿Por qué Assembler y por qué C?



15

El MPLAB Xpress

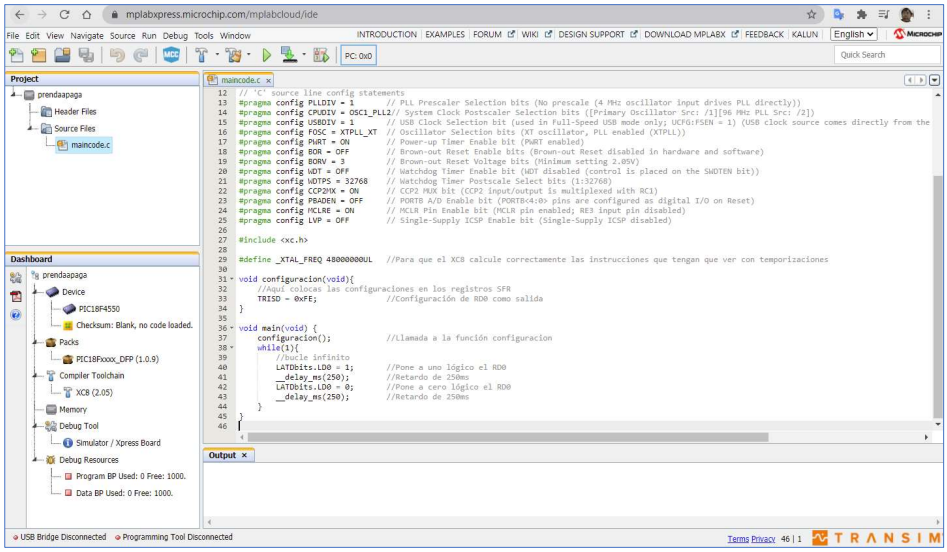
- Link: https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xpress
- Entorno de trabajo online
- Permite realizar proyectos en XC sin tener que instalar el MPLAB X en el computador.



16

El MPLAB Xpress

- Los proyectos se almacenan en la nube de Microchip
- Soporte solo para algunos modelos de microcontrolador PIC



17

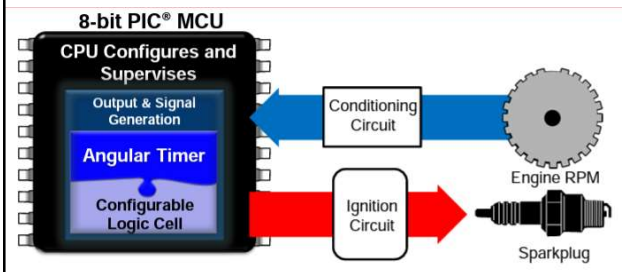
El Microchip Code Configurator (MCC)

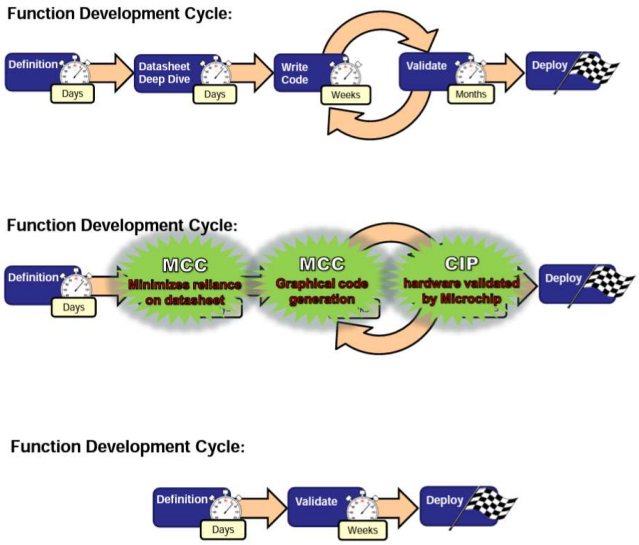
- Entorno de programación/configuración gráfico
- Interface intuitiva para desarrollos rápidos de prototipos
- Configuración automatizada para periféricos y funciones
  - Minimiza el uso de la hoja técnica para configuraciones
  - Reduce el esfuerzo y tiempo dedicado a la configuración

18

El Microchip Code Configurator (MCC)

- Caso de uso





19

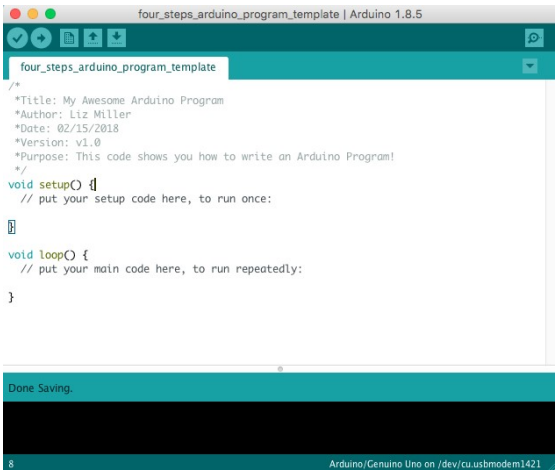
MPLAB X: Creación de un proyecto en XC8 (lenguaje de alto nivel)

- Link del manual de XC8:
  - http://ww1.microchip.com/downloads/en/devicedoc/50002053g.pdf
- Link de descarga del XC8 v2.46:
  - https://www.microchip.com/mplabxc8windows

20

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.



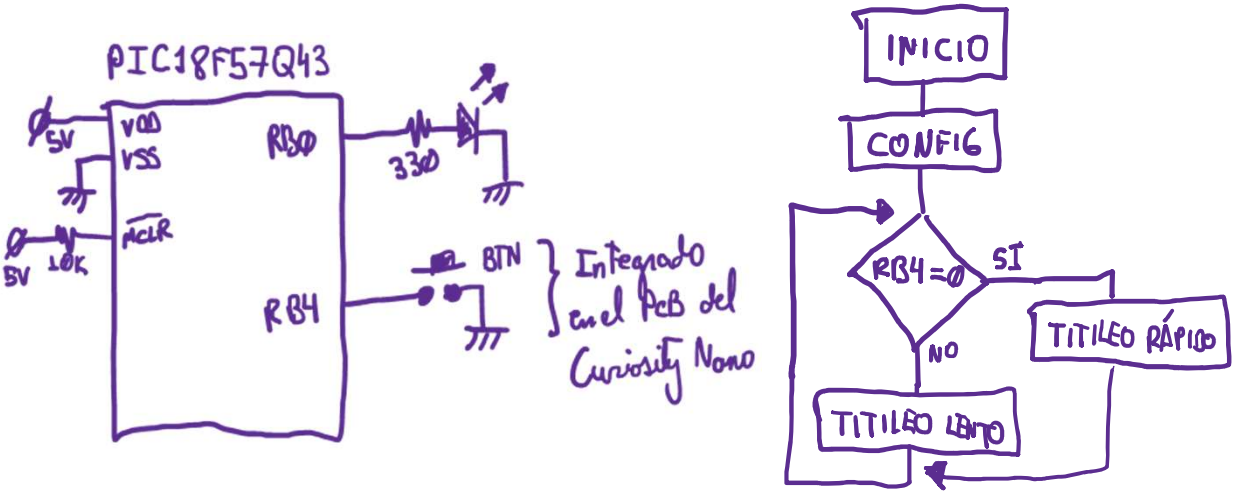
21

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
    }
}
```

22

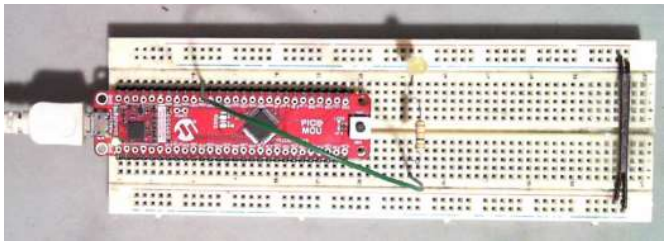
Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4



23

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)
```

24

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

- Código en XC8 (con HFINTOSC a 48MHz)

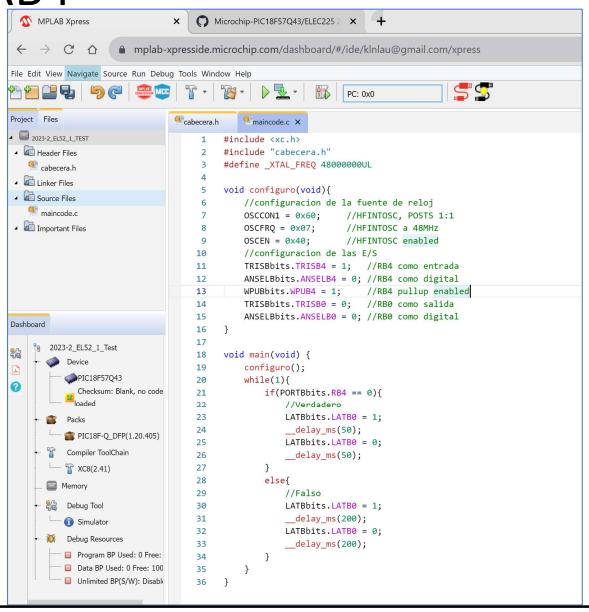
```c
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include "cabecera.h"

#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);
        }
    }
}
```

25

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

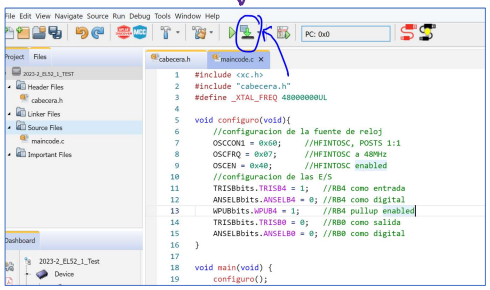


26

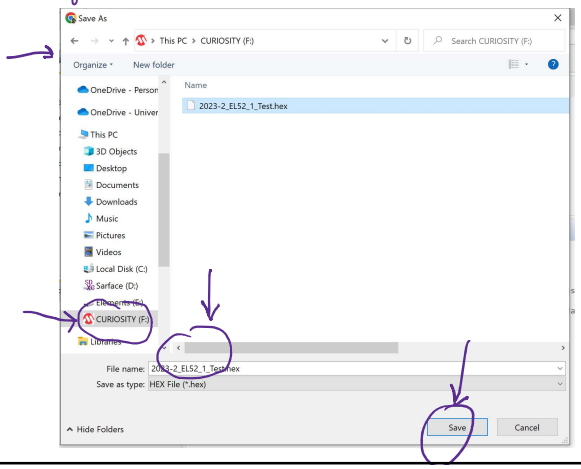
Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4

- Procedimiento para implementar este ejemplo utilizando el MPLAB Xpress

1º Presionar el boton "Make and Program Device"



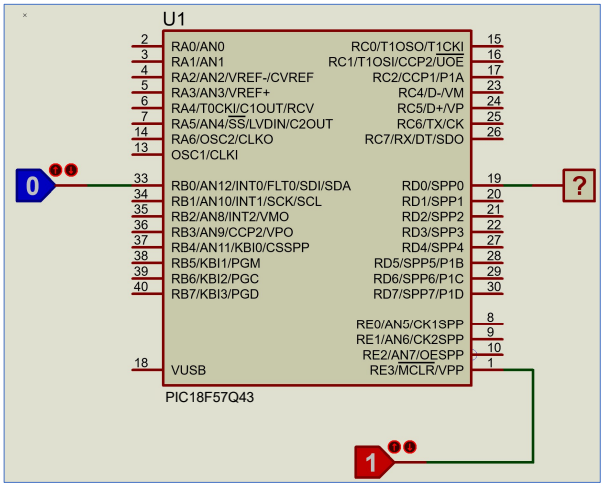
2. Se abrirá una ventana de explorador de archivos
3: Conectar el Curiosity Nano a la PC
4: Aparecerá una unidad de disco "Curiosity"
5: Elegir dicha unidad para grabar el archivo Hex



27

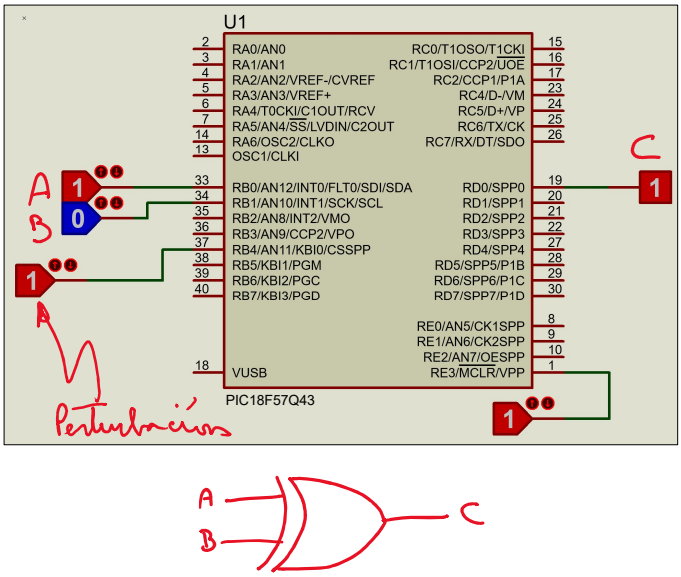
Ejemplo en XC8: Negador lógico de un bit

```
#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;
        }
    }
}
```



28

Ejemplo en XC8: Compuerta XOR

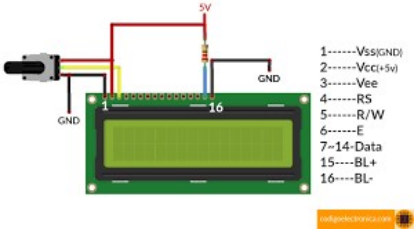



```
void configuracion(void) {
    TRISDbits.RD0 = 0;          //Puerto RD0 como salida
}

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;
        }
    }
}
```

29

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



30

El LCD alfanumérico HD44780

- ROM de caracteres:
  - 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()

31

El LCD alfanumérico HD44780

- Tabla de caracteres ASCII de 7 bits

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

32

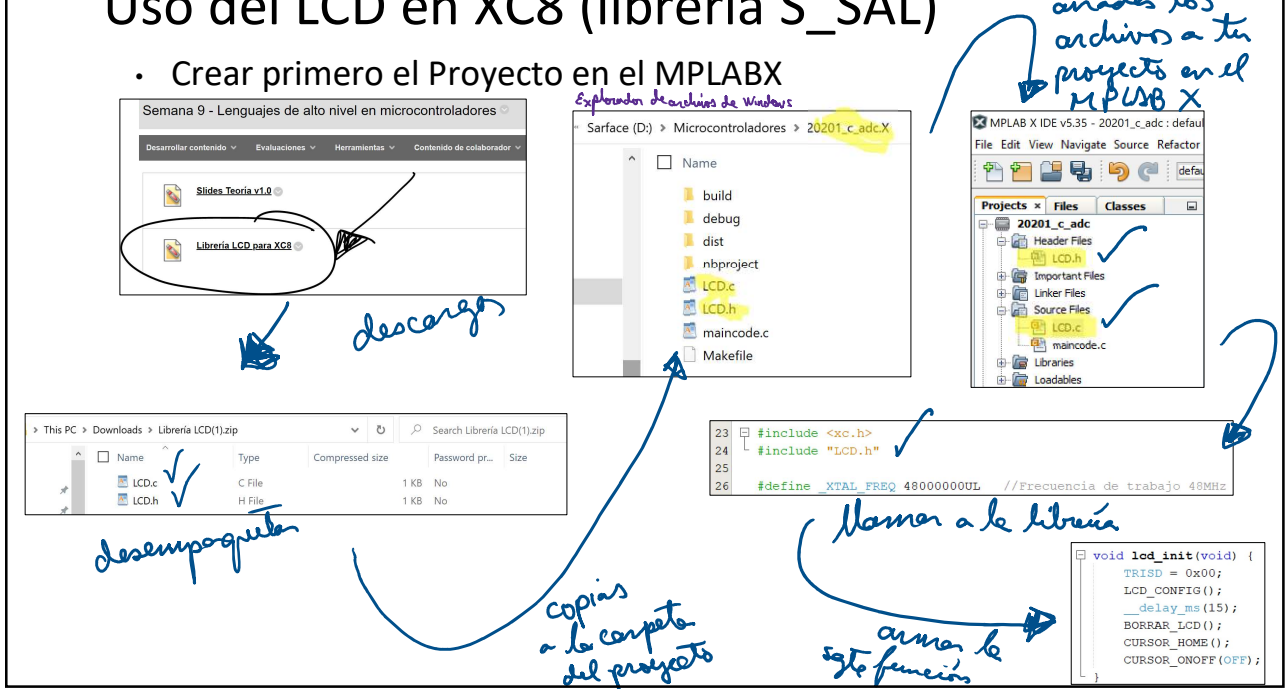
El LCD alfanumérico HD44780

- 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)
- Video de manipulación de LCD sin microcontrolador:
  - https://www.youtube.com/watch?v=cXpeTxC3_A4

33

Uso del LCD en XC8 (librería S_SAL)

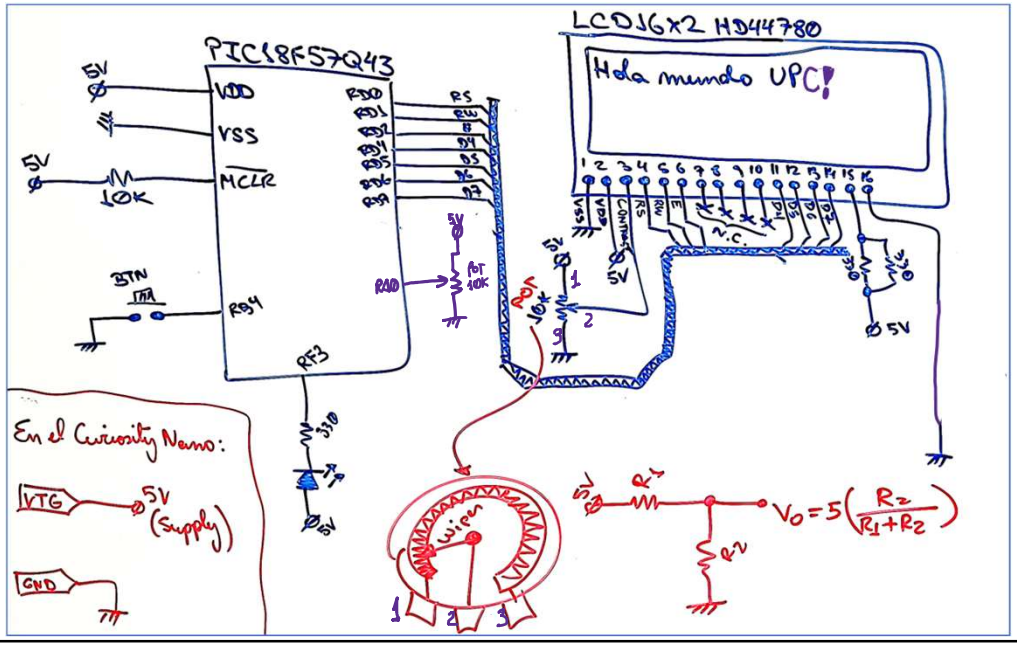
- Crear primero el Proyecto en el MPLABX



34

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

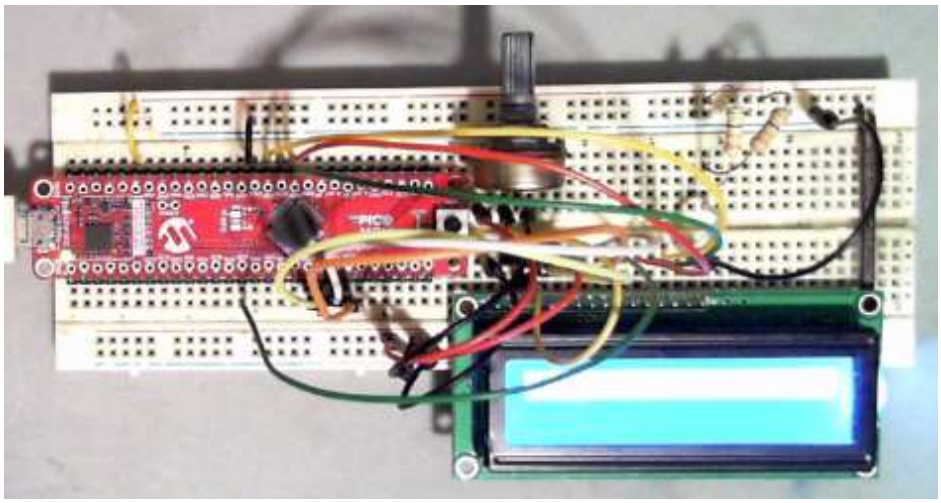
- Hardware



35

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

- Hardware



36

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
    }
}
```

37

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);
        }
    }
}
```

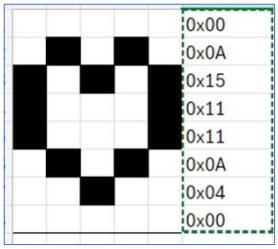
38

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/

39

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

```
11    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

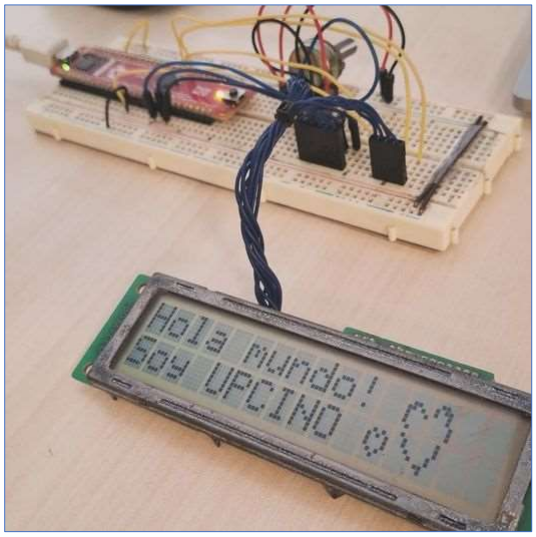
4. Para llamar al caracter personalizado, utilizar la función ENVIA_CHAR()

```
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);
```

40

Caracteres personalizados

Visualización de una figura empleando multiples caracteres personalizados



H	I	J	K	L	M	N	O	P	Q	R	S
										0x00	0x00
										0x00	0x0C
										0x0C	0x16
										0x1A	0x12
										0x11	0x02
										0x10	0x01
										0x10	0x01
										0x08	0x01
										0x08	0x02
										0x04	0x02
										0x04	0x02
										0x02	0x04
										0x01	0x04
										0x00	0x18
										0x00	0x08
										0x00	0x00

41

Conversor A/D

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?

42

Conversor A/D

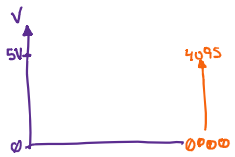
- Resolución: 12bits (ADRESH:ADRESL)
  - Posee un bit ADFM (justificación del resultado)
  - Total 4096 escalones (rango de 0 a 4095)
- Cantidad de canales analógicos: 43
- Se lee un canal analógico a la vez
- ¿Interviene el teorema de muestreo? Si.
- Rango máximo de voltaje de entrada por canal: 0-5V?
  (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

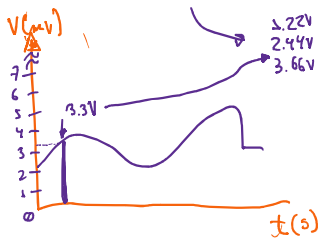
43

Sensibilidad del A/D del PIC18F57Q43

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



Ejemplo:



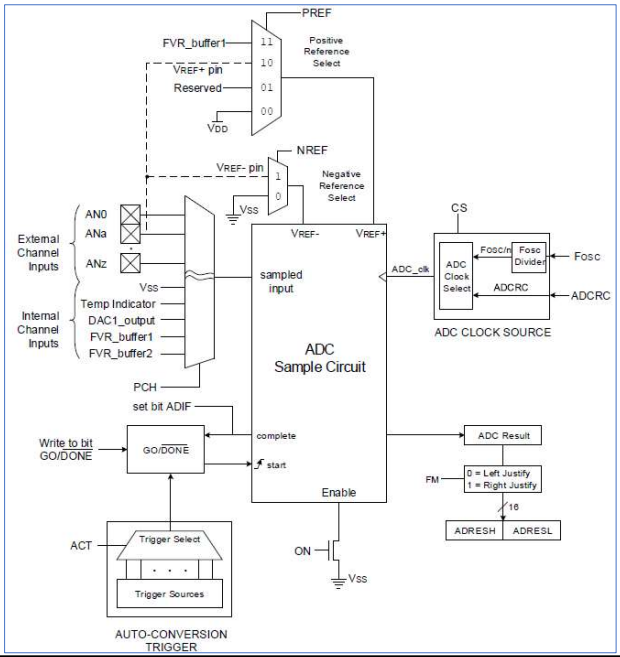
1.22V
2.44V
3.66V → Error de cuantificación

44

Alternativa de A/D de mayor resolución

45

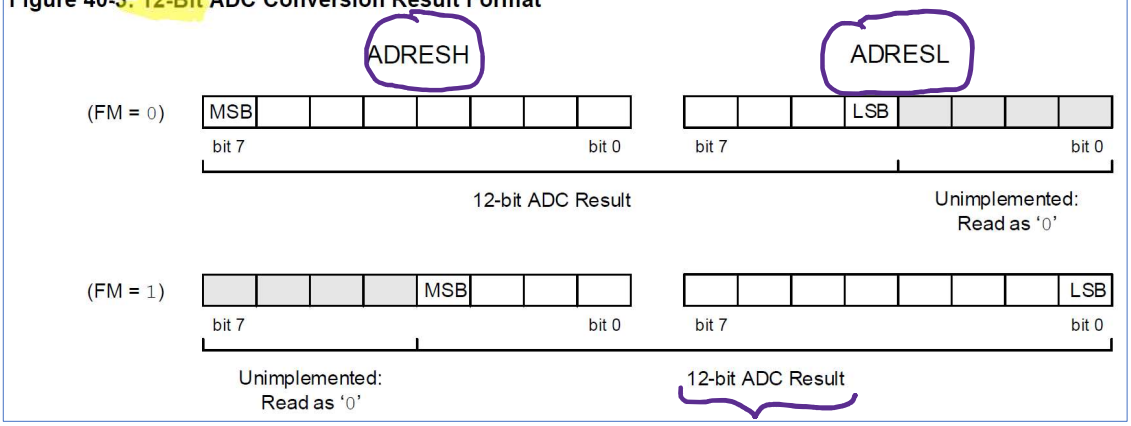
Conversor A/D: Diagrama de bloques



46

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
```

47

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]					

48

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

49

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

50

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.

51

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		Application
	DSEN = 0 Single-Sample Mode	DSEN = 1 CVD Double-Sample Mode(1)	
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

52

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[15:18]
- ADRESL: Accesses the low byte ADRES[7:0]

53

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.

54

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}

55

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)

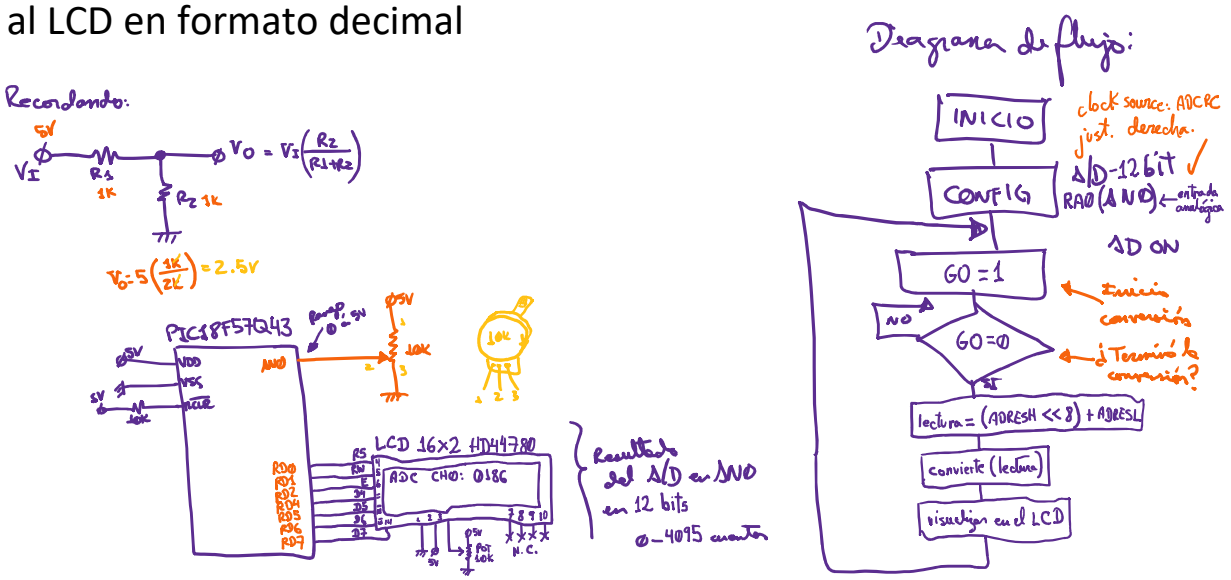
1. Establecer qué puerto será el que reciba la señal analógica
2. Configurar dicho puerto para que sea del tipo entrada y analógico (TRISx.y y ANSELx.y donde “x” es el puerto y “y” el pin)
3. Definir el bit FM: 0 para justificación del resultado a la izquierda, 1 para justificación a la derecha
4. Definir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

56

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



57

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**.

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

58

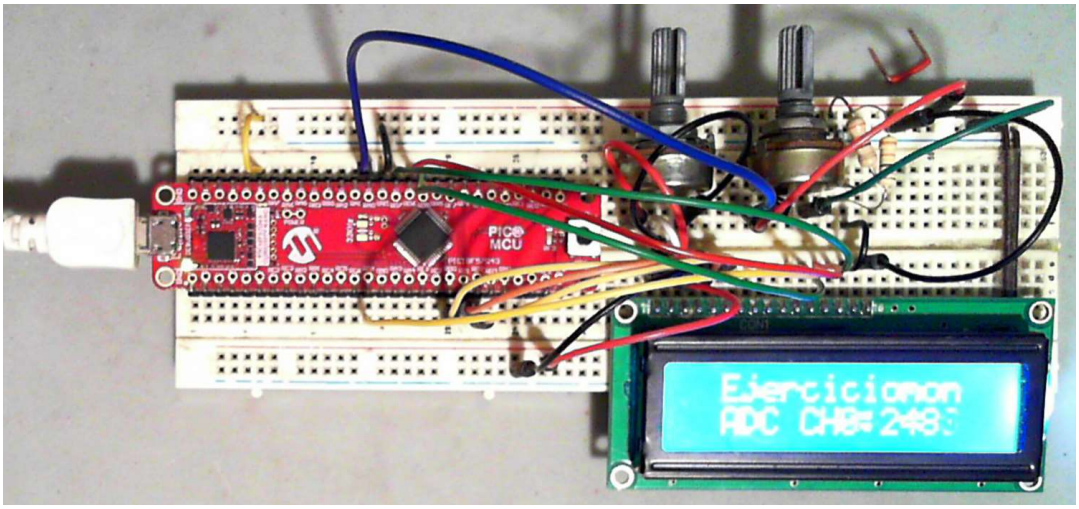
Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal

```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);
    }
}
```

59

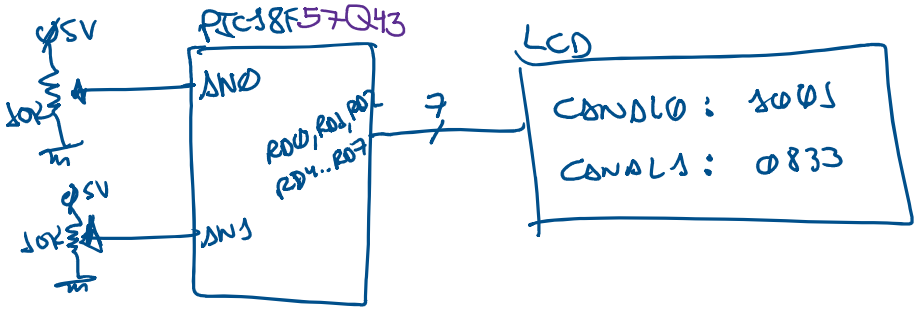
Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal



60

Asignación:

- Leer dos canales analógicos y mostrarlos en el LCD



El módulo DAC del PIC18F57Q43

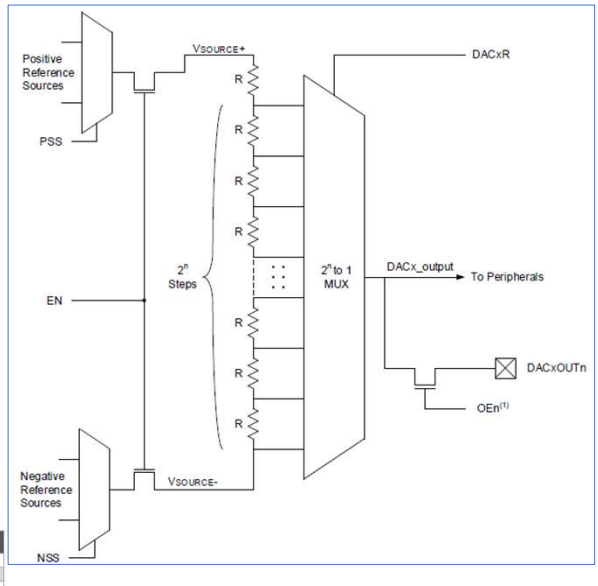
- Referencia: capítulo 41 del datasheet
- Resolución: 8bits
- Tipo arreglo R-2R
- Se puede usar el FVR como voltaje de referencia positivo (bits PSS)
- Fórmula:

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}

63

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)
  - Habilitar el módulo con DAC1CON bit 7 (EN)
  - Establecer el valor de 8 bits en DAC1DATL

64

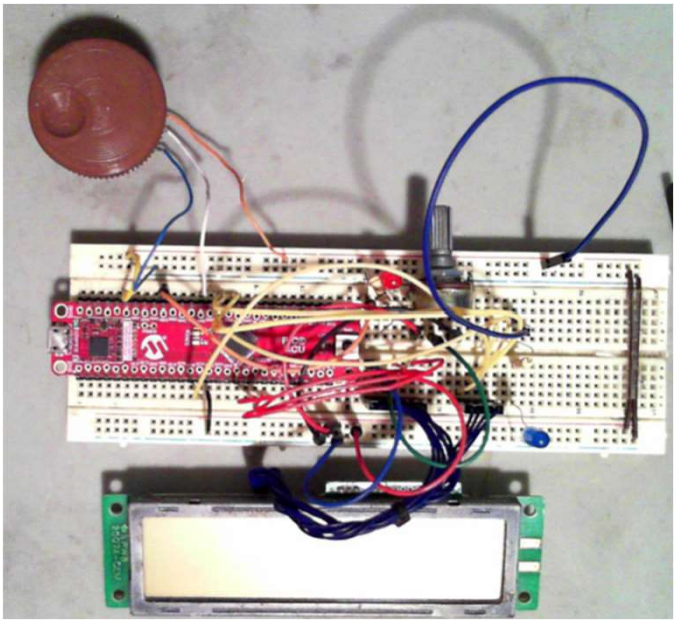
Evidencias de ejemplos 2024-1

65

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



66

Implementación 2024-1

Prueba inicial:

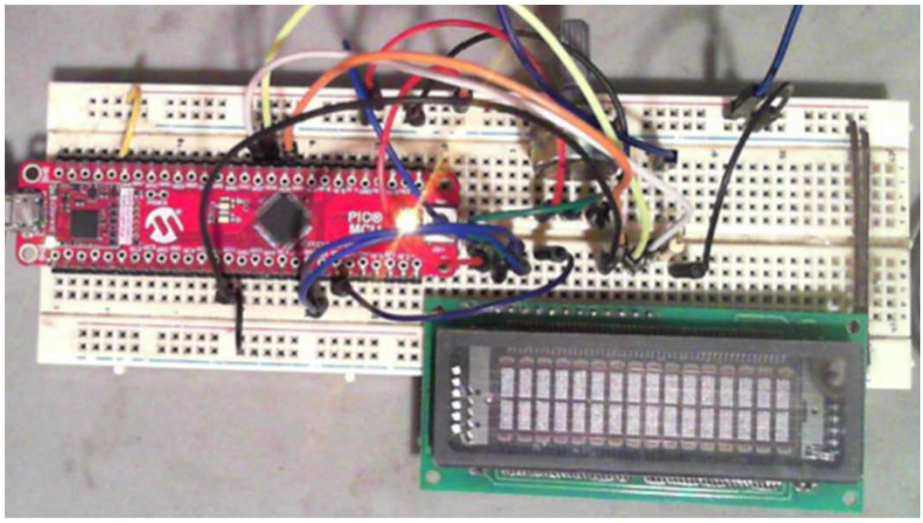
- Frecuencia de trabajo 48MHz con HFINTOSC
- Titilar el LED integrado ubicado en RF3 con un periodo de 400ms

```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
    }
}
```

67

Implementación 2024-1

Evidencia de titileo de LED en RF3



68

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
        }
    }
}
```

69

```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);
        }
    }
}
```

Implementación 2024-1

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.

70

Implementación 2024-1

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

unsigned char indicador = 0;    //variable global

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x07;
    OSCEN = 0x40;
    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 con weakpullup
    PIE1bits.INT0IE = 1;        //INT0 habilitada
    INTCON0bits.GIE = 1;        //global habilitado
    INTCON0bits.INT0EDG = 0;    //detecccion falling edge
    INT0PPS = 0x0C;             //ruteando INT0 para RB4
}
```

```c
void main(void) {
    configuro();
    while(1){
        if(indicador == 0){
            LATFbits.LATF3 = 0;     //enciendo LED
            __delay_ms(300);        //retardo de 300ms
            LATFbits.LATF3 = 1;     //apago el LED
            __delay_ms(300);        //retardo de 300ms
        }
        else if(indicador == 1){
            LATFbits.LATF3 = 1;     //apago el LED
        }
    }
}
```

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

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

}
```

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.

71

Implementación 2024-1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 48000000UL /*para que el compilador sepa
                                la frq que esta trabajando el

unsigned char estado_LED = 0;

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 main(void) {
    configuro();
    while(1){
        switch(estado_LED){
            case 0:
                LATFbits.LATF3 = 1; //LED off
                break;
            case 1:
                LATFbits.LATF3 = 0; //LED on
                __delay_ms(400);    //retardo de 400ms
                LATFbits.LATF3 = 1; //LED off
                __delay_ms(400);    //retardo de 400ms
                break;
            case 2:
                LATFbits.LATF3 = 0; //LED on
                __delay_ms(100);    //retardo de 100ms
                LATFbits.LATF3 = 1; //LED off
                __delay_ms(100);    //retardo de 100ms
                break;
        }
    }
}
```

```c
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;        //bajamos bandera INT0IF
    if(estado_LED == 2){
        estado_LED = 0;
    }
    else{
        estado_LED++;
    }
}

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

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.

72

Implementación 2024-1

```c
#include <xc.h>
#include "cabecera.h"
#define _XTAL_FREQ 48000000UL /*para que el compilador sepa
                              la frq que esta trabajando el M
#define apagado 0
#define parpadeando 1

unsigned char estado_LED = apagado;

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 main(void) {
    configuro();
    while(1){
        if(estado_LED == apagado){  //pregunto estado
            LATFbits.LATF3 = 1; //LED off
        }
        else if(estado_LED == parpadeando){
            LATFbits.LATF3 = 0; //LED on
            __delay_ms(400);    //retardo de 400ms
            LATFbits.LATF3 = 1; //LED off
            __delay_ms(400);    //retardo de 400ms
        }
        else{
        }
    }
}
```

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

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

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.

73

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)

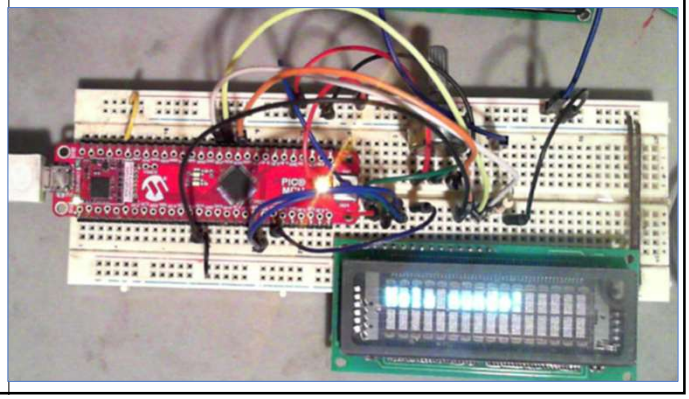
74

Implementación 2024-1

```
#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);
    }
}
```

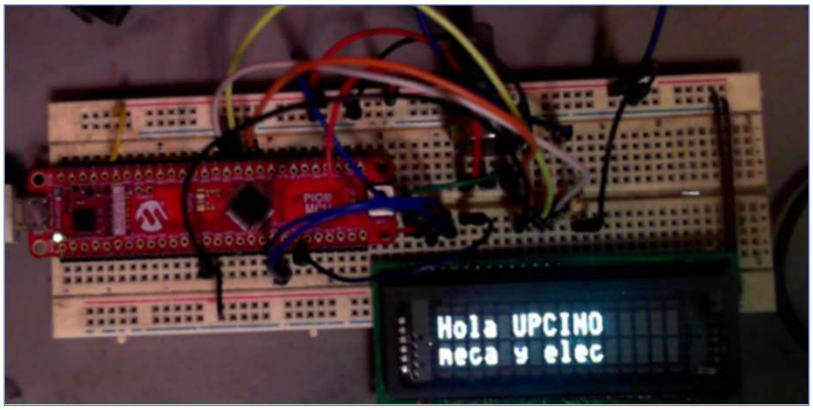
Segunda prueba: Visualizar "Hola mundo!" en el LCD:

- 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.



75

Implementación 2024-1



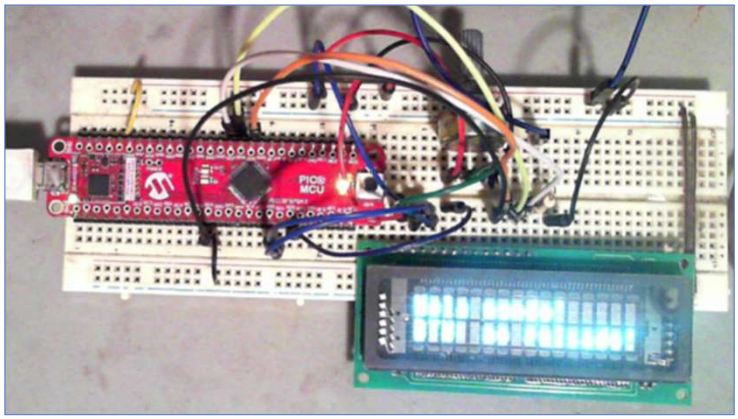
Ejemplo de visualización de mensajes en ambas líneas del LCD:

- 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

76

Implementación 2024-1

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



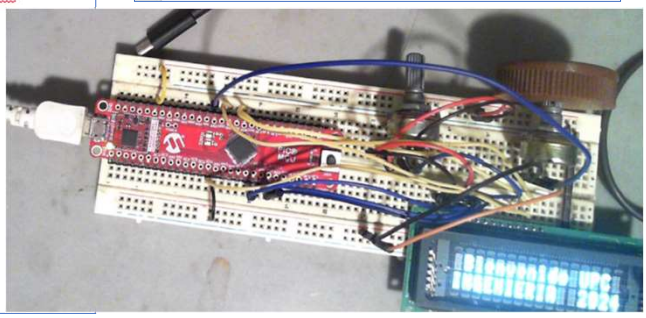
```c
#include <stdio.h>
#include <stdlib.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

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!");
        }
    }
}
```

77

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){

    }
}
```

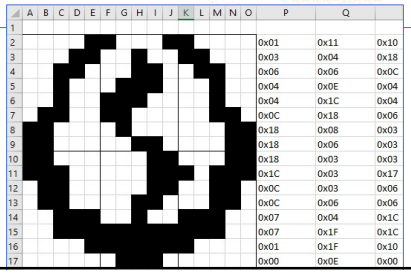


78

EL54 2024_1 Sem10

Logo de UPC empleando seis caracteres personalizados

```
#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
}
```



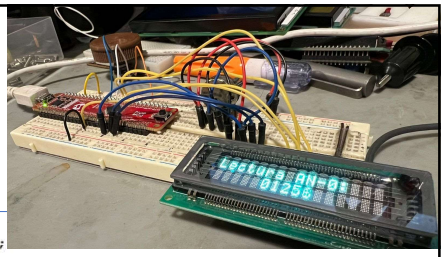
0x01	0x11	0x10
0x03	0x04	0x18
0x06	0x06	0x0C
0x04	0x0E	0x04
0x04	0x1C	0x04
0x0C	0x18	0x06
0x18	0x08	0x03
0x18	0x06	0x03
0x18	0x03	0x03
0x1C	0x03	0x17
0x0C	0x03	0x06
0x0C	0x06	0x06
0x07	0x04	0x1C
0x07	0x1F	0x1C
0x01	0x1F	0x10
0x00	0x0E	0x00

```
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);
}
```

```
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){

    }
}
```

79

Implementación ADC 2024-1




```
#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 un
    while(ADCON0bits.GO == 1);
    //el resultado esta en ADRESH:ADRESL
}
```

```
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

80

```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);   //viz 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
    }
}
```

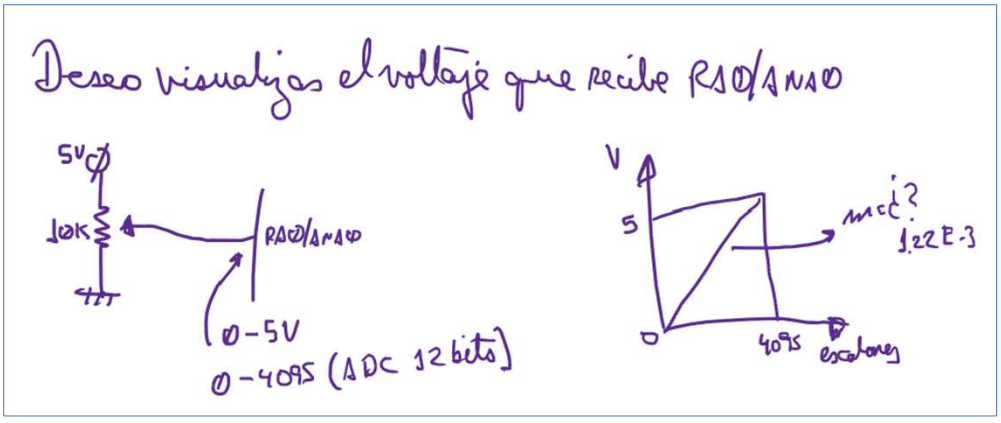
```c
void LCD_init(void){
    TRISD = 0x00;
    ANSELD = 0x00;
    __delay_ms(19);
    LCD_CONFIG();
    __delay_ms(21);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
    __delay_ms(100);
    GENERACARACTER(logo_upc_1, 0);
    GENERACARACTER(logo_upc_2, 1);
    GENERACARACTER(logo_upc_3, 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

81

Implementación ADC 2024-1



82

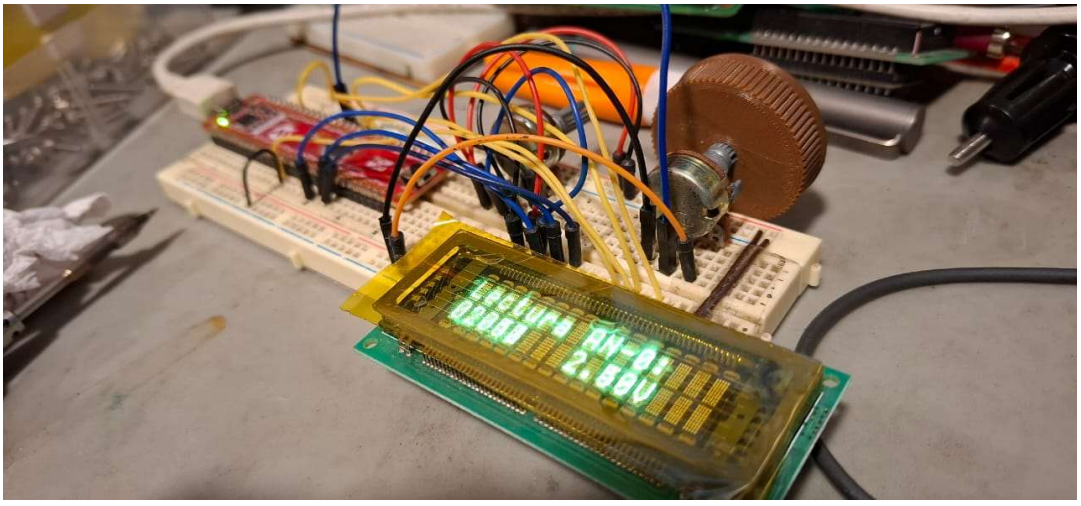
EL54 2024_1 Sem10
Modificación del main() para que se visualice el valor en voltaje

```
54  void main(void) {
55      unsigned int resultado;
        float calculo;
        unsigned int voltaje;
58      configuro();
59      LCD_init();
60      POS_CURSOR(1,6);
61      ENVIA_CHAR(0);
62      ENVIA_CHAR(1);
63      ENVIA_CHAR(2);
64      POS_CURSOR(2,6);
65      ENVIA_CHAR(3);
66      ENVIA_CHAR(4);
67      ENVIA_CHAR(5);
68      __delay_ms(5000);
69      BORRAR_LCD();
```

```
70      while(1){
71          POS_CURSOR(1,0);
72          ESCRIBE_MENSAJE("Lectura RA0:",12);
73          resultado = lectura_ADC_0();
74          POS_CURSOR(2,0);
75          ENVIA_CHAR((resultado / 10000) + 0x30);   //vis diez millar
76          ENVIA_CHAR(((resultado % 10000) / 1000) + 0x30);  //viz millar
77          ENVIA_CHAR(((resultado % 1000) / 100) + 0x30);  //viz centena
78          ENVIA_CHAR(((resultado % 100) / 10) + 0x30);    //viz decena
79          ENVIA_CHAR((resultado % 10) + 0x30);    //viz unidad
80          calculo = resultado * 0.122;
81          voltaje = calculo;          //convierte float a integer
82          POS_CURSOR(2,8);
83          ENVIA_CHAR(((voltaje % 1000) / 100) + 0x30);  //viz centena
84          ENVIA_CHAR('.');
85          ENVIA_CHAR(((voltaje % 100) / 10) + 0x30);    //viz decena
86          ENVIA_CHAR((voltaje % 10) + 0x30);    //viz unidad
87          ENVIA_CHAR('V');
88      }
89  }
```

83

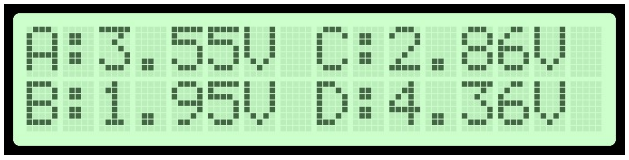
Implementación 2024-1



84

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.

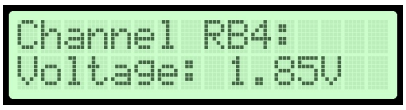


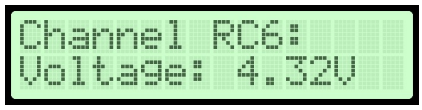
- Carpeta compartida: https://bit.ly/3KeDi3L
- Formato de nombre de archivo de video de evidencia: EL256_EL51_[tu primer apellido]_[tu primer nombre]_Sem10.mp4
  - Ejemplo de nombre de archivo de video de evidencia: EL256_EL51_Perez_Juan_Sem10.mp4
- Video grabarlo en calidad 720p como máximo y 30 segundos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

85

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:



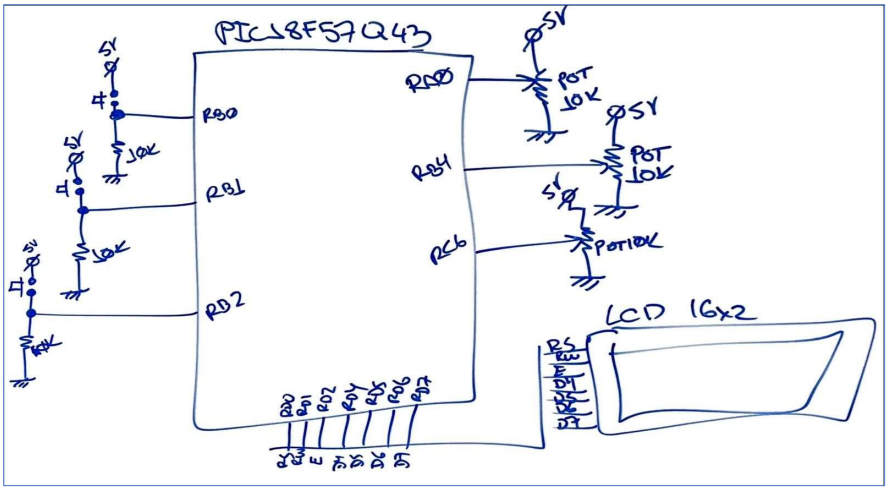


- El intercambio de pantallas será al pulsar un botón activo en alto en INT0 para que se visualice la pantalla de RA0,al pulsar el botón activo en alto en INT1 para que se visualice la pantalla de RB4 y al pulsar el botón actvo en alto en INT2 para visualizar la pantalla de RC6
- Carpeta compartida: 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.

86

Asignación Semana 10 2024-1

- Hardware



87

Solución

```
#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;
}

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);
}

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;
        }
    }
}

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();
}

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;
}
```

88

Anexos: Ejemplos similares a los anteriores

89

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){
}
```

90

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){

}
```

91

Implementación 2024-1

```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){

}
```

92

Fin de la sesión

93