

Microcontroladores

Profesor: Kalun José Lau Gan

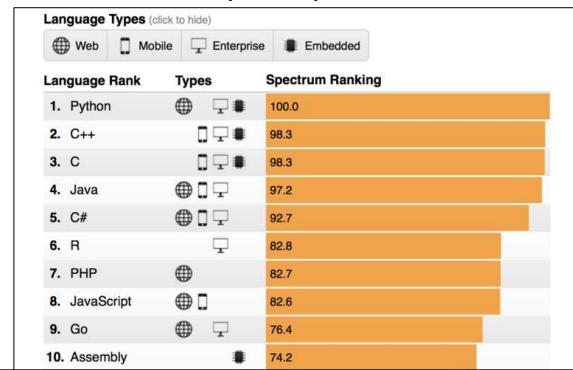
Semestre 2024-1

Semana 9-10

1

¿Preguntas previas?

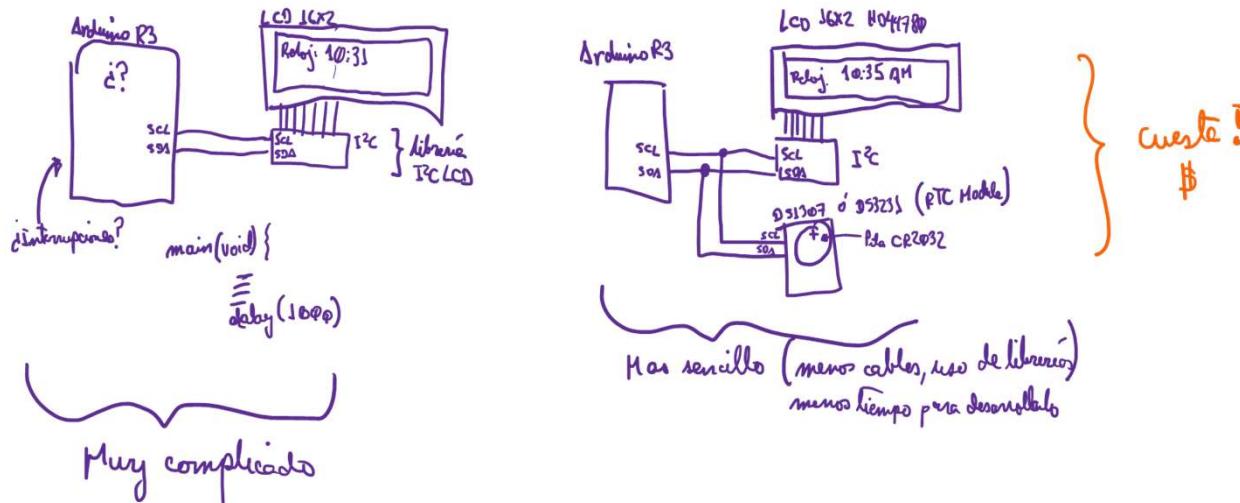
- ¿El hecho de usar C lo hace más fácil?
 - El proceso de diseño es el mismo tanto para lenguaje Assembler como en lenguaje C, se tiene que implementar primero el hardware, desarrollar el algoritmo y finalmente aterrizar el algoritmo en un lenguaje de programación.
 - El lenguaje XC se va a permitir 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 desarrollan aplicaciones con Arduino

- El reloj debe de ser preciso



3

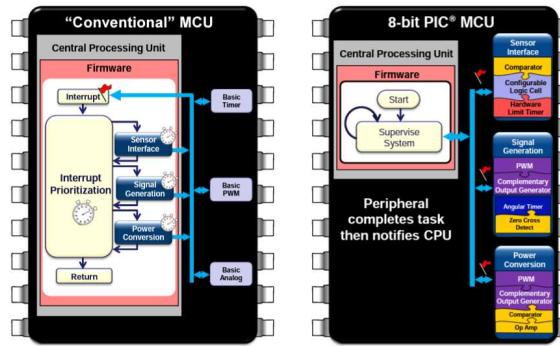
Agenda

- 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)

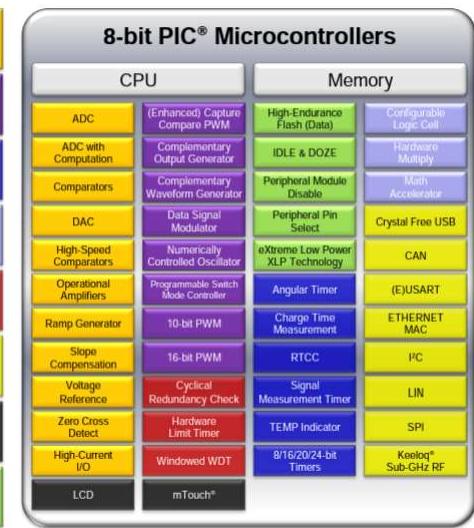
- Independizando el funcionamiento de los periféricos en el microcontrolador se logra un mejor rendimiento del CPU
- Los periféricos independientes administran sus tareas sin código ni supervisión del CPU para mantener su funcionamiento



5

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

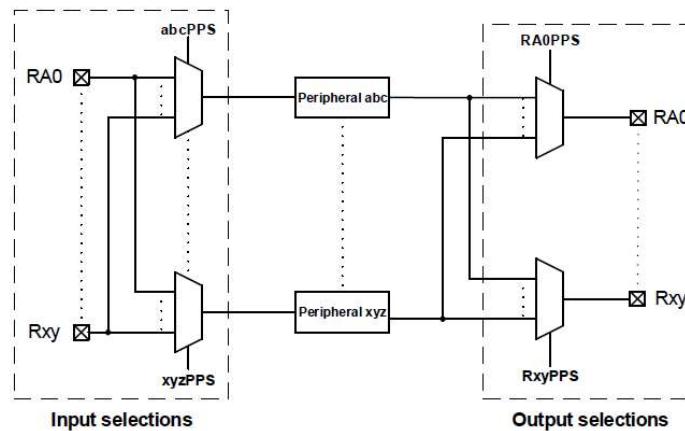
- Dependiendo del modelo y familia el microcontrolador tendrá un conjunto de periféricos CIP
- Los CIPs poseen conexiones adicionales con otros CIP para lograr la independencia del CPU
- Estos CIP se apoyan en el sistema de interconexión de pines (PPS)



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.

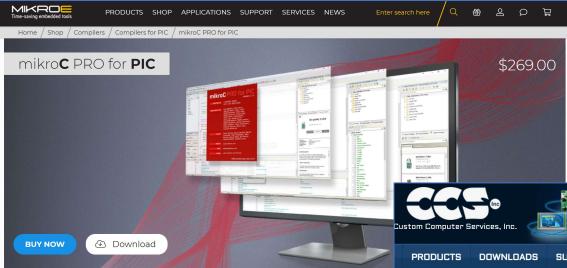
Problema de compatibilidad entre distintas plataformas de desarrollo aún empleando un mismo lenguaje.

- 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) más 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.



PCWHD IDE Compiler for Microchip PIC10/12/16/18/24/dsPIC Devices

SKU: 52202-588

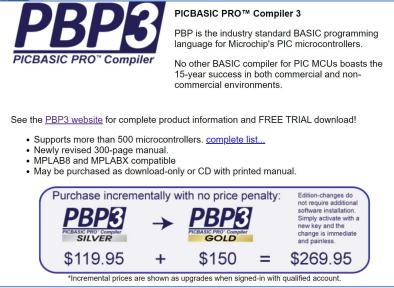
Devices Supported:

PIC10, PIC12, PIC16, PIC18, PIC24, dsPIC	12-bit Instructions 4.8 kB PIC MAC	PIC10, PIC12, PIC14, PIC16, PIC18, PIC24, dsPIC	14-bit Instructions 8.8 kB PIC MAC	PIC18	16-bit Instructions 16 kB PIC MAC
POW	PCWH	PCWHD			

In stock (ships immediately)

Download version available \$600.00 [Add to Cart](#)

Starting at \$100 more, buy a complete development kit [PICkit 3](#)



PBP3 PICBASIC PRO™ Compiler 3

PBP is the industry standard BASIC programming language for Microchip's PIC microcontrollers. No other BASIC compiler for PIC MCUs boasts the 15-year success in both commercial and non-commercial environments.

See the [PBP3 website](#) for complete product information and FREE TRIAL download!

- Supports more than 500 microcontrollers: [complete list](#).
- Newly revised 300-page manual.
- MPLAB® and MPLABX compatible.
- May be purchased as download-only or CD with printed manual.

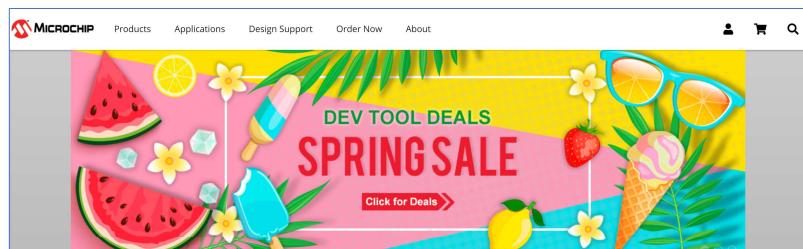
Purchase incrementally with no price penalty:
 → 
\$119.95 + \$150 = \$269.95

*Incremental prices are shown as upgrades when signed-in with qualified account.

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



DEV TOOL DEALS SPRING SALE

[Click for Deals](#)

MPLAB® XC Compilers

Available as free, unrestricted-use downloads, our award-winning MPLAB® XC Compilers are comprehensive solutions for your project's software development. Finding the right compiler to support your device is simple:

- MPLAB XC8 supports all 8-bit PIC® and AVR® microcontrollers (MCUs)
- MPLAB XC16 supports all 16-bit PIC MCUs and dsPIC® Digital Signal Controllers (DSCs)
- MPLAB XC32/32++ supports all 32-bit PIC and SAM MCUs and MPUs



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
1+

USD per Unit
\$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**

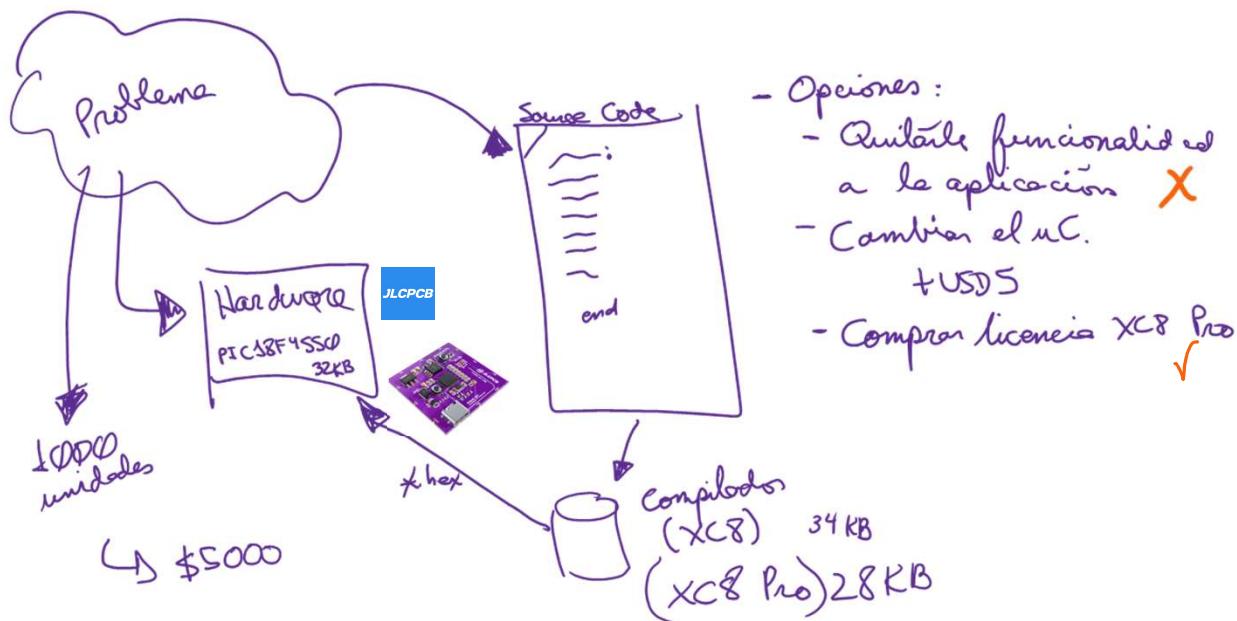
Lead Time For Additional Quantities [»](#)
Additional quantities can ship by **11-Aug-2020**

Quantity:



11

Caso:



12

Optimización en el compilador XC

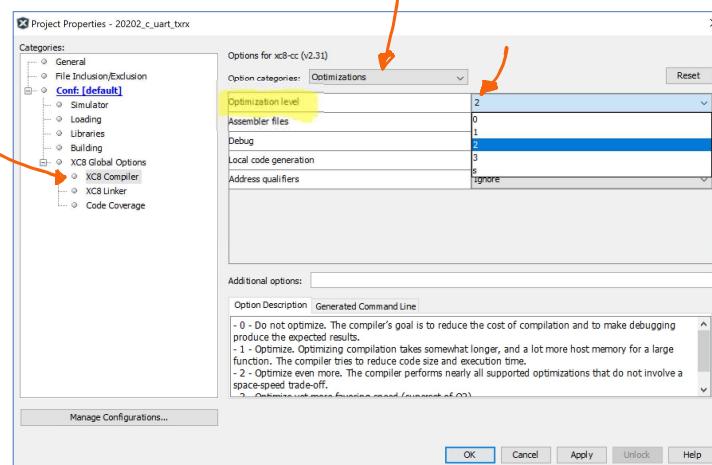
- **Optimization levels:**
 - Level 0: Fastest compilation time, minimal optimizations
 - Level 1: Optimizations with low debugging impact
 - Level 2: All optimizations that give the best balance between speed and size
 - Level 3: Program execution will be as fast as possible
 - Level s: Code size will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

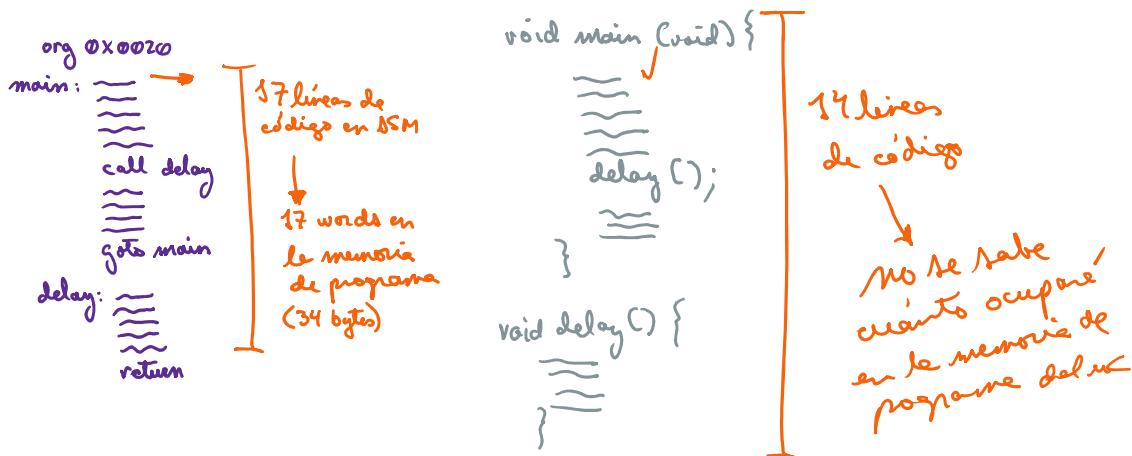
- Para acceder a las opciones de optimización ingresamos a las propiedades del proyecto:



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

```

12 // 'C' source line config statements
13 #pragma config PLLDIV = 1           // PLL Prescaler Selection bits (No prescale (a High oscillator input drives PLL directly))
14 #pragma config FCKSM = SOSC1_PLL2 // System Clock Postscale Selection bits ((Primary oscillator Src: /11 (M0 Main PLL Src: /2))
15 #pragma config USOSELV = 1          // USB Clock Selection bit (used in Full-Speed USB mode only; UCFQIFSDN = 1) (USB clock source comes directly from the
16 #pragma config FOSC = XTPLL_XT    // Oscillator Selection bits (XT oscillator, PLL enabled (XTPLL))
17 #pragma config PR2 = ON            // Power-up Timer Enable bit (PWRT enabled)
18 #pragma config PWRT = OFF          // Power-up Timer Reset bit (PWRT disabled)
19 #pragma config BORV = 3             // Brown-out Reset Voltage bits (Minimum setting 2.05V)
20 #pragma config WDT = OFF           // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
21 #pragma config FOSCSEL = ITR0_32768 // Oscillator Selection bits (XT oscillator, fosc = 32768 Hz)
22 #pragma config CCP2MX = ON          // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
23 #pragma config PBADEN = OFF         // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)
24 #pragma config MCLEN = ON           // MCLR Pin Enable bit (MCLR pin enabled; MES input pin disabled)
25 #pragma config IUP = OFF            // Single-Supply ICEP Faultie Bit (Single-Supply ICEP disabled)
26
27 #include <xc.h>
28
29 #define _XTAL_FREQ 48000000UL //Para que el XC8 calcule correctamente las instrucciones que tengan que ver con temporizaciones
30
31 void configuration(void){ //Aqui colocas las configuraciones en los registros SFR
32     TRISD = 0x0E; //Configuración de RD8 como salida
33 }
34
35
36 void main(void){ //Llamada a la función configuración
37     configuration();
38     while(1){ //Bucle infinito
39         LATDDbits.LD0 = 1; //Pone a uno Logico el RD0
40         delay_ms(250); //Retardo de 250ms
41         LATDDbits.LD0 = 0; //Pone a cero Logico el RD0
42         delay_ms(250); //Retardo de 250ms
43     }
44 }
45
46

```

USB Bridge Disconnected Programming Tool Disconnected Terms Privacy 46 | 1 TRAN SIM

17

El Microchip Code Configurator (MCC)



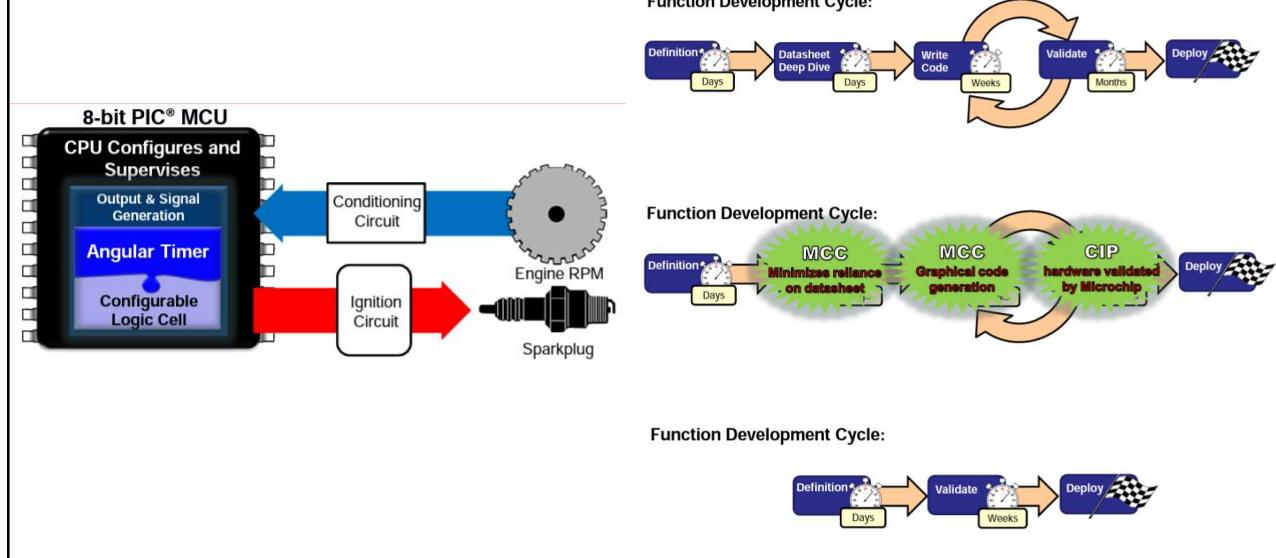
- Entorno de programación/configuración gráfico
- Interface intuitiva para desarrollos rápidos de prototipos
- Configuración automatizada para periféricos y funciones
 - **Minimiza el uso de la hoja técnica para configuraciones**
 - Reduce el esfuerzo y tiempo dedicado a la configuración

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.

```

four_steps_arduino_program_template | Arduino 1.8.5
four_steps_arduino_program_template
/*
 *Title: My Awesome Arduino Program
 *Author: Liz Miller
 *Date: 02/15/2018
 *Version: v1.0
 *Purpose: This code shows you how to write an Arduino Program!
 */
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

Done Saving.

8 Arduino/Genuino Uno on /dev/cu.usbmodem1421

```

21

Plantilla de código en XC8:

```

#include <xc.h>

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

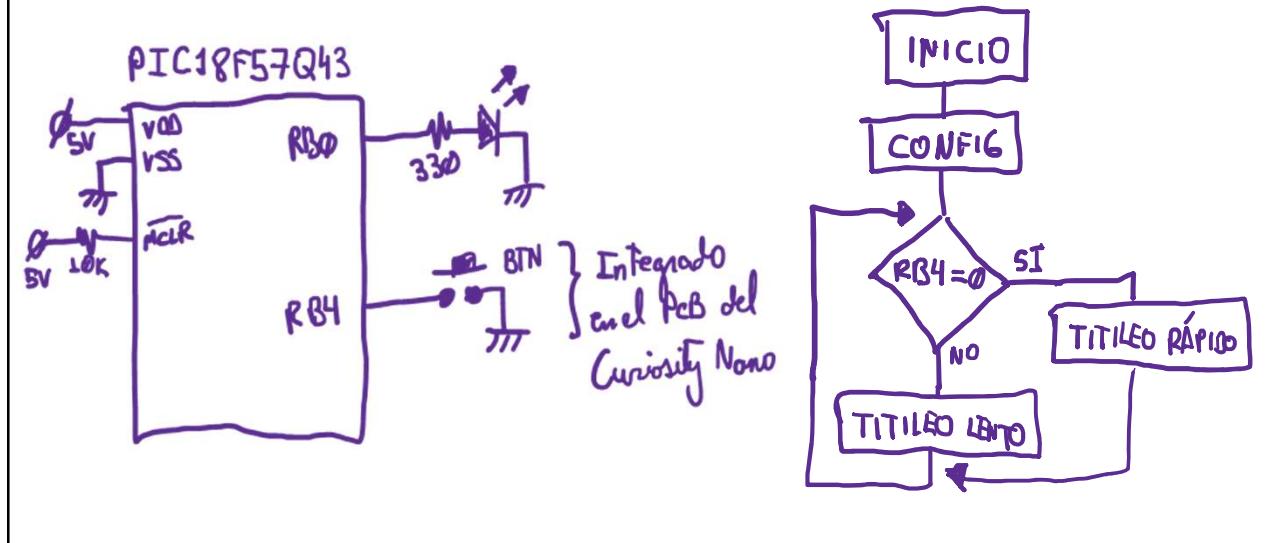
void configuracion(void) {
  //Aqui colocas las configuraciones iniciales
}

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

```

22

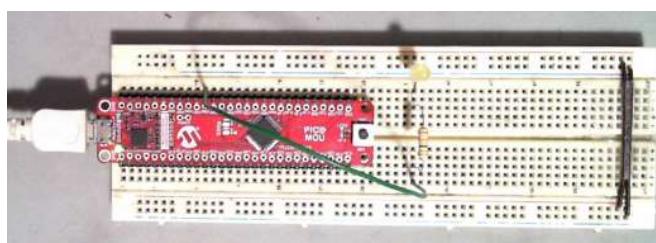
Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4



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)

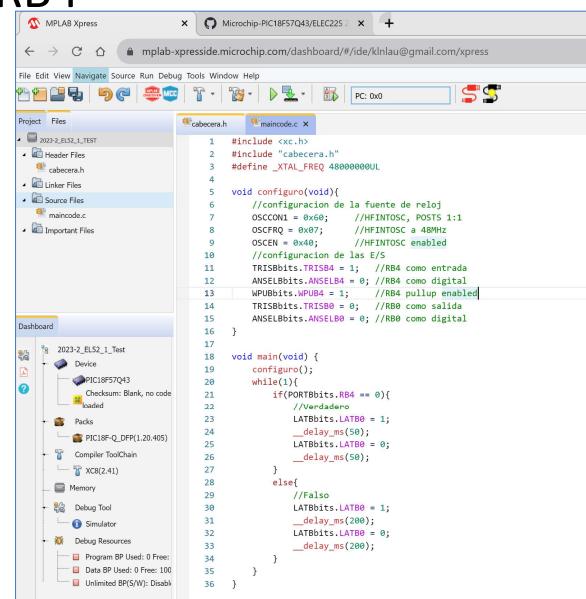
```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <xc.h>
4  #include "cabecera.h"
5
6  #define _XTAL_FREQ 48000000UL
7
8  void configuro(void){
9      //configuración del reloj
10     OSCCON1 = 0x60;
11     OSCFRQ = 0x07;      //HFINTOSC a 48MHz
12     OSCEN = 0x40;
13     //configuración de las E/S
14     TRISBbits.TRISB0 = 0;    //RB0 como salida
15     ANSELBbits.ANSELB0 = 0; //RB0 como digital
16     TRISBbits.TRISB4 = 1;   //RB4 como entrada
17     ANSELBbits.ANSELB4 = 0; //RB4 como digital
18     WPUBbits.WPUB4 = 1;    //RB4 pullup activado
19 }
20
21 void main(void){
22     configuro();
23     while(1){
24         if(PORTBbits.RB4 == 0) {
25             //parpadeo rápido
26             LATBbits.LATB0 = 1;        //enciendo LED
27             __delay_ms(100);
28             LATBbits.LATB0 = 0;        //apago LED
29             __delay_ms(100);
30         }
31         else{
32             //parpadeo lento
33             LATBbits.LATB0 = 1;        //enciendo LED
34             __delay_ms(200);
35             LATBbits.LATB0 = 0;        //apago LED
36             __delay_ms(200);
37         }
38     }
39 }
```

25

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

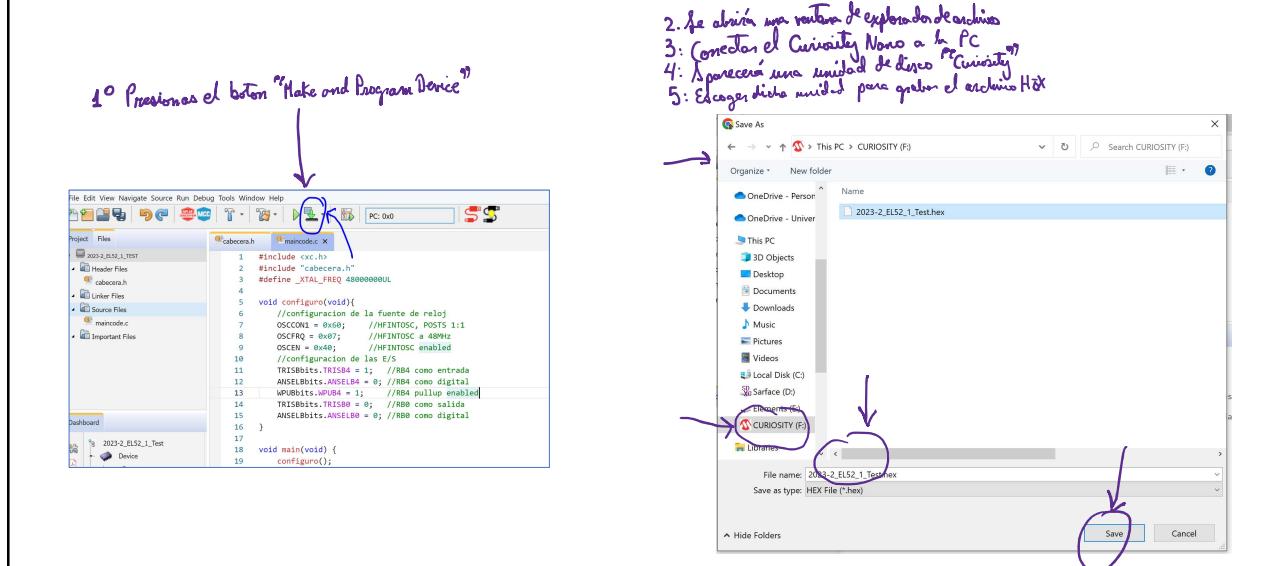
- Procedimiento para implementar este ejemplo utilizando el MPLAB Xpress
 - URL: <https://www.mplab-xpresside.microchip.com/>
 - Debes de registrarte para crear una cuenta dentro de dicha plataforma
 - Seguir las mismas indicaciones para crear un proyecto como en el MPLABX



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

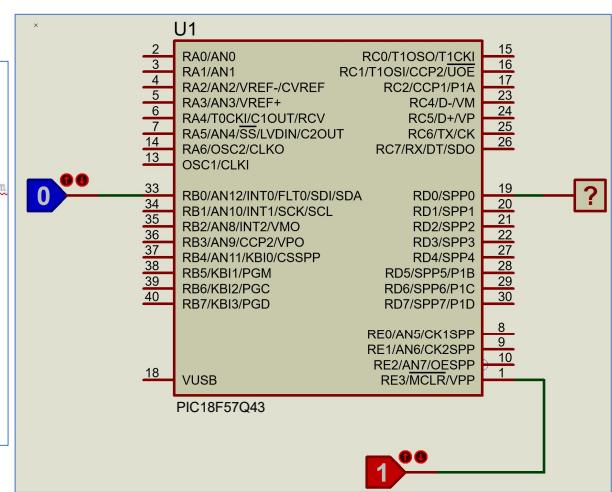


27

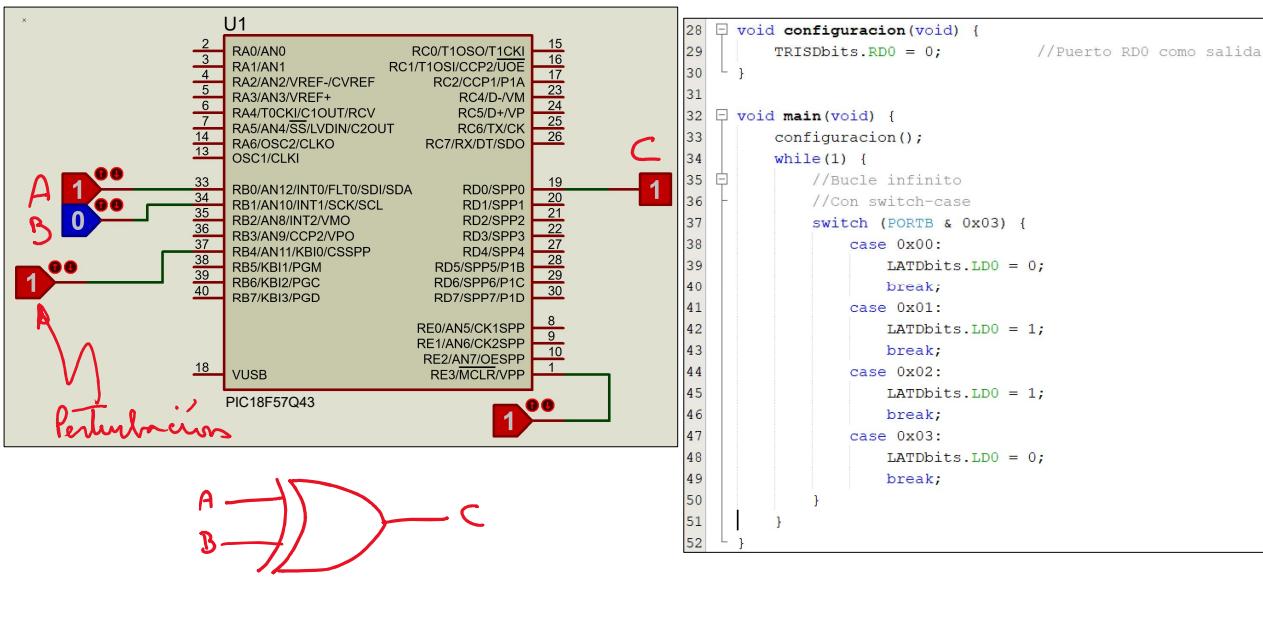
Ejemplo en XC8: Negador lógico de un bit

```

15 #include <xc.h>
16 #define _XTAL_FREQ 48000000UL
17
18 void init_conf(void) {
19     //Aca colocaremos las configuraciones iniciales de la aplicacion
20     //TRISDbits.RD0 = 0;           // RD0 como salida
21     asm("bcf TRISD, 0");        // Escribiendo instrucciones en mpasm
22 }
23
24 void main(void) {
25     init_conf();
26     while(1) {
27         if (PORTBbits.RB0 == 1) {
28             LATDbits.LD0 = 0;
29         }
30         else{
31             LATDbits.LD0 = 1;
32         }
33     }
34 }
```



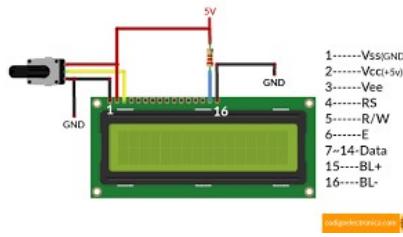
Ejemplo en XC8: Compuerta XOR



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()

Linea	Columna	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	(1)	0	0	P	^	F				-	フ	ミ	ア	ル	ル	ル	ル
xxxx0001	(2)	!	1	A	Q	a	q			。	ア	フ	ル	ル	ル	ル	ル
xxxx0010	(3)	"	2	B	R	b	r			フ	イ	フ	ル	ル	ル	ル	ル
xxxx0011	(4)	#	3	C	S	c	s			フ	ウ	テ	ミ	ミ	ミ	ミ	ミ
xxxx0100	(5)	\$	4	D	T	d	t			、	エ	ト	ル	ル	ル	ル	ル
xxxx0101	(6)	%	5	E	U	e	u			・	オ	ナ	ル	ル	ル	ル	ル
xxxx0110	(7)	&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ル	ル	ル	ル
xxxx0111	(8)	'	7	G	W	g	w			ア	キ	ス	ラ	ル	ル	ル	ル
xxxx1000	(1)	(8	H	X	h	x			イ	ク	ネ	リ	ル	ル	ル	ル
xxxx1001	(2))	9	I	Y	i	y			カ	ケ	ル	ル	ル	ル	ル	ル
xxxx1010	(3)	*	:	J	Z	j	z			コ	ハ	ル	ル	ル	ル	ル	ル
xxxx1011	(4)	+	;	K	[k]			ア	サ	ヒ	ロ	ル	ル	ル	ル
xxxx1100	(5)	,	<	L	¥	l	l			シ	フ	ワ	ル	ル	ル	ル	ル
xxxx1101	(6)	-	=	M]	m	>			ス	ヘ	ン	ル	ル	ル	ル	ル
xxxx1110	(7)	.	>	N	^	n	→			エ	ヤ	ホ	ル	ル	ル	ル	ル
xxxx1111	(8)	/	?	O	_	o	~			レ	ウ	マ	ル	ル	ル	ル	ル

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	4#32;	Space	64	40 100	4#64;	Ø	96	60 140	4#96;	`	102	70 160	4#102;	p
1	1 001	SOH	(start of heading)	33	21 041	4#33;	!	65	41 101	4#65;	A	97	61 141	4#97;	a	103	71 161	4#103;	q
2	2 002	STX	(start of text)	34	22 042	4#34;	"	66	42 102	4#66;	B	98	62 142	4#98;	b	104	72 162	4#104;	r
3	3 003	ETX	(end of text)	35	23 043	4#35;	#	67	43 103	4#67;	C	99	63 143	4#99;	c	105	73 163	4#105;	s
4	4 004	EOT	(end of transmission)	36	24 044	4#36;	\$	68	44 104	4#68;	D	100	64 144	4#100;	d	106	74 164	4#106;	t
5	5 005	ENQ	(enquiry)	37	25 045	4#37;	%	69	45 105	4#69;	E	101	65 145	4#101;	e	107	75 165	4#107;	k
6	6 006	ACK	(acknowledge)	38	26 046	4#38;	&	70	46 106	4#70;	F	102	66 146	4#102;	f	108	76 166	4#108;	l
7	7 007	BEL	(bell)	39	27 047	4#39;	'	71	47 107	4#71;	G	103	67 147	4#103;	g	114	77 167	4#114;	m
8	8 010	BS	(backspace)	40	28 050	4#40;	{	72	48 110	4#72;	H	104	68 150	4#104;	h	115	78 168	4#115;	n
9	9 011	TAB	(horizontal tab)	41	29 051	4#41;	}	73	49 111	4#73;	I	105	69 151	4#105;	i	116	79 169	4#116;	o
10	A 012	LF	(NL line feed, new line)	42	2A 052	4#42;	*	74	4A 112	4#74;	J	106	6A 152	4#106;	j	117	7A 170	4#117;	p
11	B 013	VT	(vertical tab)	43	2B 053	4#43;	+	75	4B 113	4#75;	K	107	6B 153	4#107;	k	118	7B 171	4#118;	q
12	C 014	FF	(NP form feed, new page)	44	2C 054	4#44;	,	76	4C 114	4#76;	L	108	6C 154	4#108;	l	119	7C 172	4#119;	r
13	D 015	CR	(carriage return)	45	2D 055	4#45;	-	77	4D 115	4#77;	M	109	6D 155	4#109;	m	120	7D 173	4#120;	s
14	E 016	SO	(shift out)	46	2E 056	4#46;	.	78	4E 116	4#78;	N	110	6E 156	4#110;	n	121	7E 174	4#121;	t
15	F 017	SI	(shift in)	47	2F 057	4#47;	/	79	4F 117	4#79;	O	111	6F 157	4#117;	o	122	7F 175	4#122;	z
16	10 020	DLE	(data link escape)	48	30 060	4#80;	Ø	80	50 120	4#80;	P	112	70 160	4#112;	p	123	71 161	4#113;	q
17	11 021	DC1	(device control 1)	49	31 061	4#49;	1	81	51 121	4#81;	Q	113	71 161	4#113;	q	124	72 162	4#114;	r
18	12 022	DC2	(device control 2)	50	32 062	4#50;	2	82	52 122	4#82;	R	114	72 162	4#114;	r	125	73 163	4#115;	s
19	13 023	DC3	(device control 3)	51	33 063	4#51;	3	83	53 123	4#83;	S	115	73 163	4#115;	s	126	74 164	4#116;	t
20	14 024	DC4	(device control 4)	52	34 064	4#52;	4	84	54 124	4#84;	T	116	74 164	4#116;	t	127	75 165	4#117;	u
21	15 025	NAK	(negative acknowledge)	53	35 065	4#53;	5	85	55 125	4#85;	U	117	75 165	4#117;	u	128	76 166	4#118;	v
22	16 026	SYN	(synchronous idle)	54	36 066	4#54;	6	86	56 126	4#86;	V	118	76 166	4#118;	v	129	77 167	4#119;	w
23	17 027	ETB	(end of trans. block)	55	37 067	4#55;	7	87	57 127	4#87;	W	119	77 167	4#119;	w	130	78 170	4#120;	x
24	18 030	CAN	(cancel)	56	38 070	4#56;	8	88	58 130	4#88;	X	120	78 170	4#120;	x	131	79 171	4#121;	y
25	19 031	EM	(end of medium)	57	39 071	4#57;	9	89	59 131	4#89;	Y	121	79 171	4#121;	y	132	7A 172	4#122;	z
26	1A 032	SUB	(substitute)	58	3A 072	4#58;	:	90	5A 132	4#90;	Z	122	7A 172	4#122;	z	133	7B 173	4#123;	{
27	1B 033	ESC	(escape)	59	3B 073	4#59;	:	91	5B 133	4#91;	[123	7B 173	4#123;	{	134	7C 174	4#124;	
28	1C 034	FS	(file separator)	60	3C 074	4#60;	<	92	5C 134	4#92;	\	124	7C 174	4#124;		135	7D 175	4#125;]
29	1D 035	GS	(group separator)	61	3D 075	4#61;	=	93	5D 135	4#93;	J	125	7D 175	4#125;]	136	7E 176	4#126;	^
30	1E 036	RS	(record separator)	62	3E 076	4#62;	>	94	5E 136	4#94;	^	126	7E 176	4#126;	^	137	7F 177	4#127;	DEL
31	1F 037	US	(unit separator)	63	3F 077	4#63;	?	95	5F 137	4#95;	_	127	7F 177	4#127;	DEL				

Source: www.LookupTables.com

32

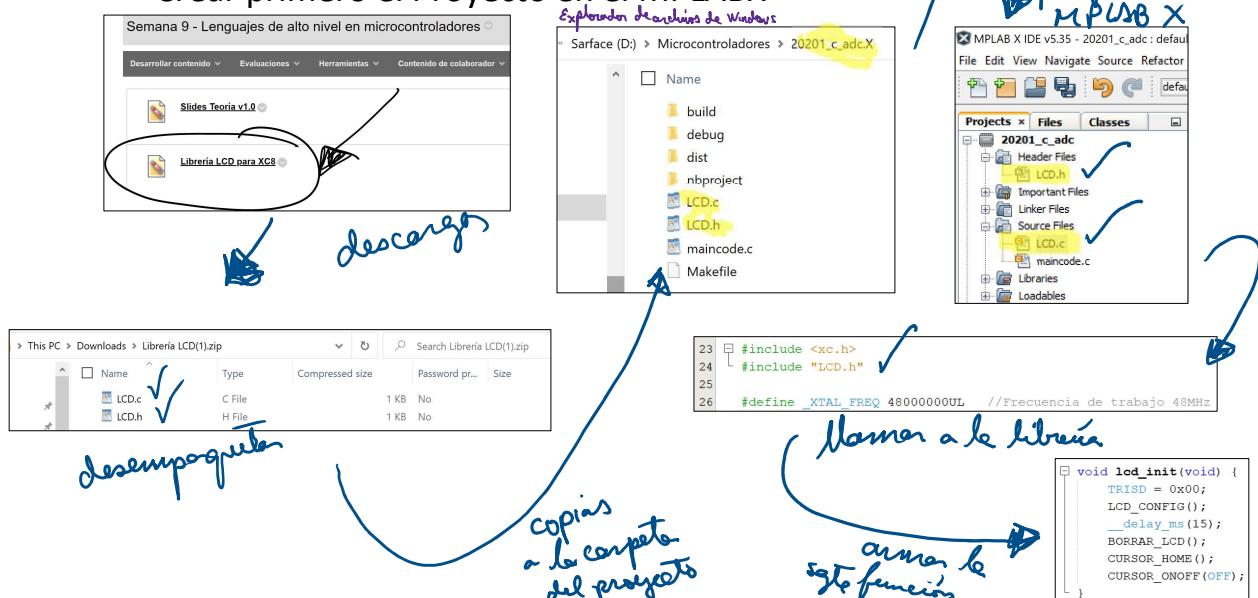
El LCD alfanumérico HD44780

- 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)

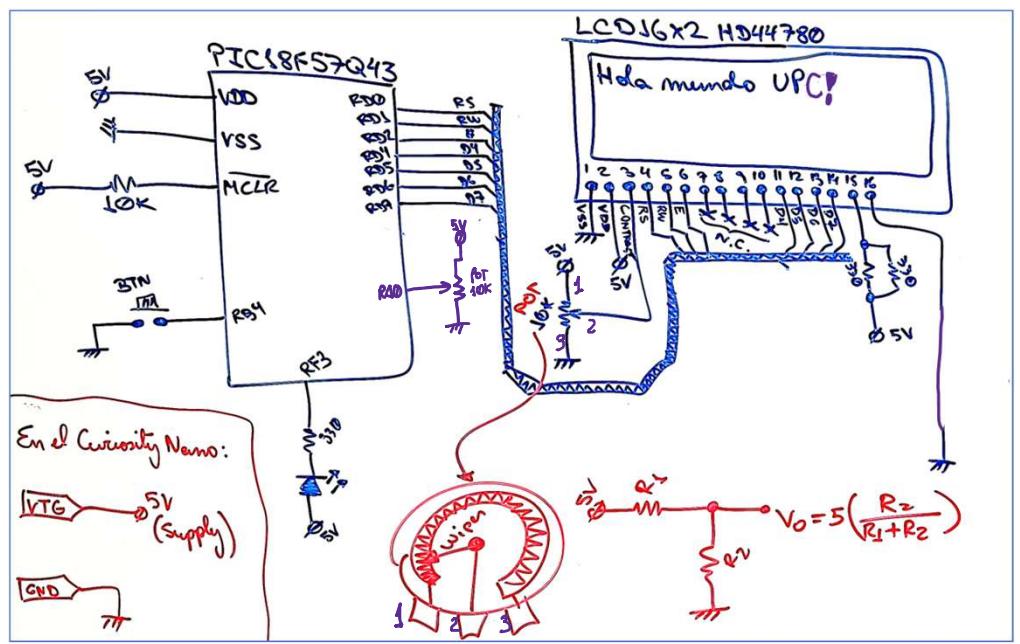
- Crear primero el Proyecto en el MPLABX



34

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

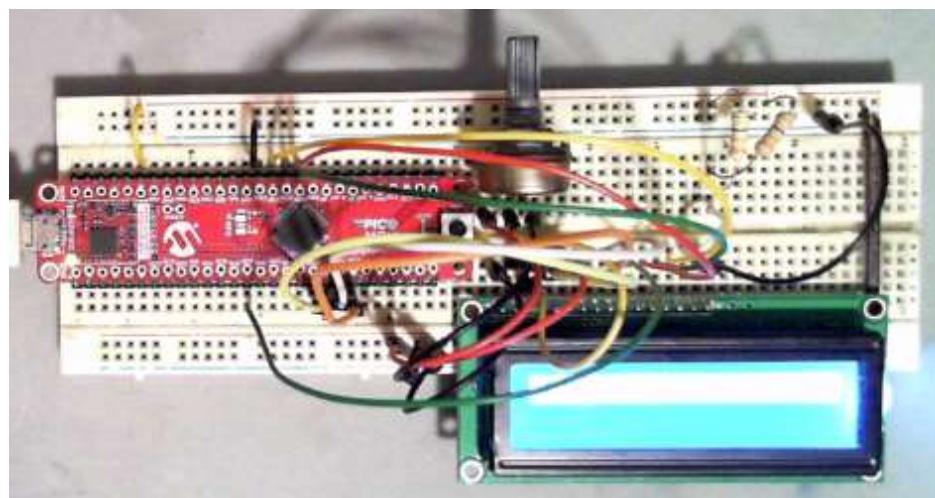
- Hardware



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

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <xc.h>
4  #include "cabecera.h"
5  #include "LCD.h"
6  #define _XTAL_FREQ 48000000UL
7
8  void configuro(void){
9      //configuracion del reloj
10     OSCCON1 = 0x60;
11     OSCFRC = 0x07;           //HFINTOSC a 48MHz
12     OSCEN = 0x40;
13     //configuracion de las E/S
14     TRISD = 0x00;
15     ANSELD = 0x00;
16     LCD_CONFIG();
17     __delay_ms(15);
18     BORRAR_LCD();
19     CURSOR_HOME();
20     CURSOR_ONOFF(OFF);
21 }
22
23 void lcd_init(void){
24     TRISD = 0x00;
25     ANSELD = 0x00;
26     LCD_CONFIG();
27     __delay_ms(15);
28     BORRAR_LCD();
29     CURSOR_HOME();
30     CURSOR_ONOFF(OFF);
31 }
32
33 void main(void){
34     configuro();
35     lcd_init();
36     POS_CURSOR(1,0);
37     ESCRIBE_MENSAJE("Hola mundo UPC!",15);
38     POS_CURSOR(2,0);
39     ESCRIBE_MENSAJE("Kalun Lau Gan",13);
40     while(1){
41         //aqui va el codigo de la aplicacion
42     }
43 }
44
45 
```

37

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

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <xc.h>
4  #include "cabecera.h"
5  #include "LCD.h"
6  #define _XTAL_FREQ 48000000UL
7
8  void configuro(void){
9      //configuracion del reloj
10     OSCCON1 = 0x60;
11     OSCFRC = 0x07;           //HFINTOSC a 48MHz
12     OSCEN = 0x40;
13     //configuracion de las E/S
14     TRISFbits.TRISF3 = 0;   //RF3 como salida
15     ANSELFBbits.ANSELF3 = 0; //RF3 como digital
16     TRISBbits.TRISB4 = 1;   //RB4 como entrada
17     ANSELBbits.ANSELB4 = 0; //RB4 como digital
18     WPUBbits.WPUB4 = 1;    //RB4 pullup activado
19 }
20
21 void lcd_init(void){
22     TRISD = 0x00;
23     ANSELD = 0x00;
24     LCD_CONFIG();
25     __delay_ms(15);
26     BORRAR_LCD();
27     CURSOR_HOME();
28     CURSOR_ONOFF(OFF);
29 }
30
31 void main(void){
32     configuro();
33     lcd_init();
34     POS_CURSOR(1,0);
35     ESCRIBE_MENSAJE("Kalun Lau Gan",13);
36     while(1){
37         //aqui va el codigo de la aplicacion
38         if(PORTBbits.RB4 == 0){
39             POS_CURSOR(2,0);
40             ESCRIBE_MENSAJE("Parradeo rapido",15);
41             //parradeo rapido
42             LATFbits.LATF3 = 1;           //enciendo LED
43             __delay_ms(100);
44             LATFbits.LATF3 = 0;          //apago LED
45             __delay_ms(100);
46         }
47         else{
48             POS_CURSOR(2,0);
49             ESCRIBE_MENSAJE("Parradeo lento ",15);
50             //parradeo lento
51             LATFbits.LATF3 = 1;           //enciendo LED
52             __delay_ms(200);
53             LATFbits.LATF3 = 0;          //apago LED
54             __delay_ms(200);
55         }
56     }
57 }
58 
```

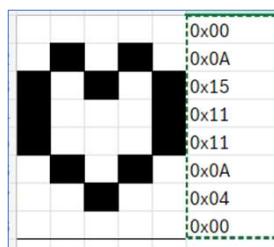
38

Caracteres personalizados en el LCD 16x2 HD44780

- Podemos incluir caracteres personalizados en nuestras visualizaciones.
- Cada carácter 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 carácter personalizado y extraer los ocho datos

2. Declarar la constante global con los ocho datos del carácter 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 carácter personalizado a la memoria CGRAM del LCD

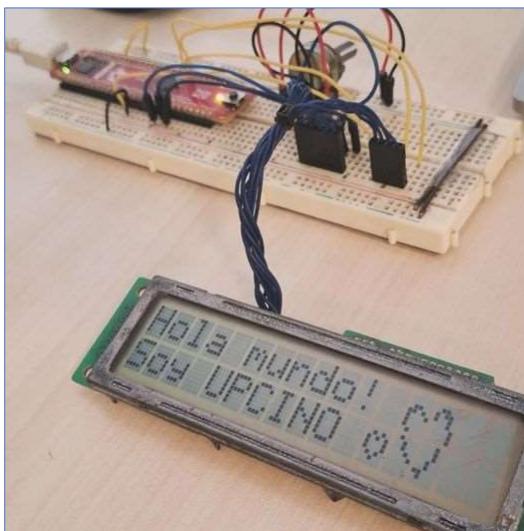
4. Para llamar al carácter personalizado, utilizar la función ENVIA_CHAR()

```
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?
($V_{ref+} = VDD$, $V_{ref-} = VSS$)
 - 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 $V_{ref+} = 5V$; $V_{ref-} = 0V$
 - El rango es $0-5V$
- Resolución es de 12 bits (4096 escalones) $\leftarrow 2^{12}$
- Altura del escalón: $\frac{5}{4096} = 1.22mV$ //



Ejemplo:

Graph of current V (mA) versus time t (s). The current starts at 0, rises to a peak of 3.3V at $t = 0.5\text{ s}$, then oscillates between 3.3V and 3.66V. Three points are marked on the curve: 3.3V at $t = 0.5\text{ s}$, 2.44V at $t = 1.5\text{ s}$, and 3.66V at $t = 2.2\text{ s}$. An arrow points from these points to the text "Error de cuantificación".

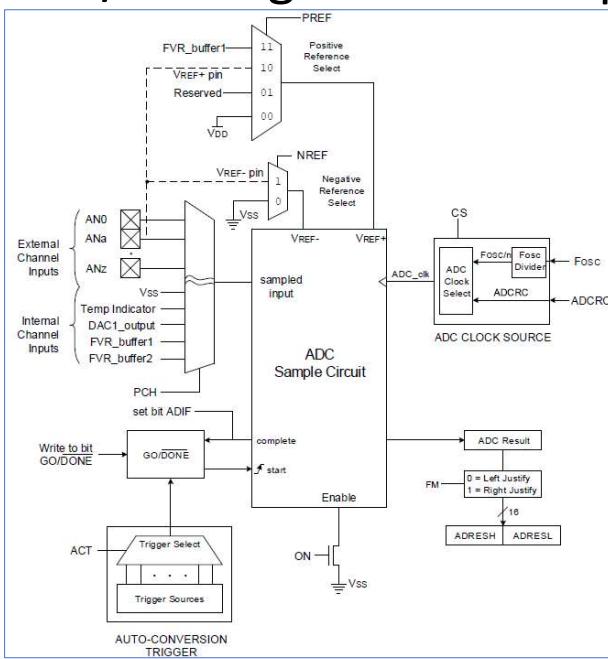
44

Alternativa de A/D de mayor resolución



45

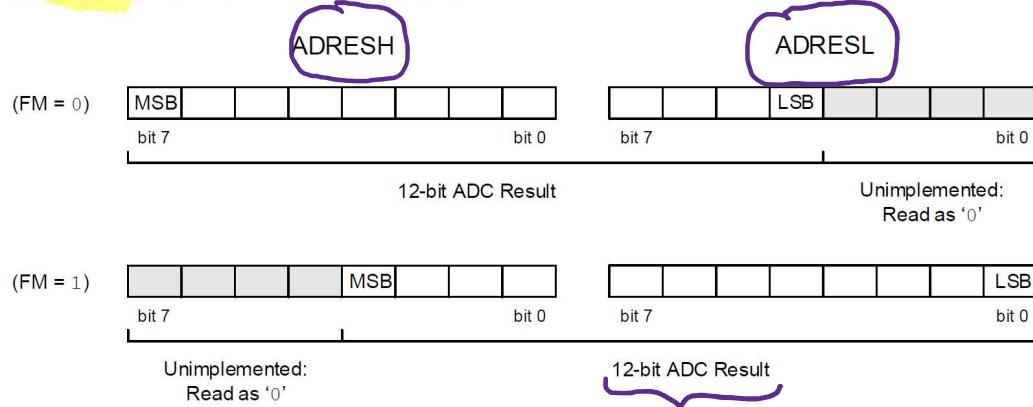
Conversor A/D: Diagrama de bloques



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]				
0x03D9	ADLTH	15:8				LTH[15:8]				
0x03DB	ADUTH	7:0				UTH[7:0]				
0x03DB	ADUTH	15:8				UTH[15:8]				
0x03DD	ADERR	7:0				ERR[7:0]				
0x03DD	ADERR	15:8				ERR[15:8]				
0x03DF	ADSTPT	7:0				STPT[7:0]				
0x03DF	ADSTPT	15:8				STPT[15:8]				
0x03E1	ADFLTR	7:0				FLTR[7:0]				
0x03E1	ADFLTR	15:8				FLTR[15:8]				
0x03E3	ADACC	7:0				ACC[7:0]				
0x03E3	ADACC	15:8				ACC[15:8]				
0x03E3	ADACC	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]				
0x03E8	ADPREV	15:8				PREV[15:8]				
0x03EA	ADRES	7:0				RES[7:0]				
0x03EA	ADRES	15:8				RES[15:8]				
0x03EC	ADPCH	7:0				PCH[5:0]				
0x03ED	Reserved	7:0								
0x03EE	ADACQ	7:0				ACQ[7:0]				
0x03EE	ADACQ	15:8				ACQ[12:8]				
0x03F0	ADCAP	7:0					CAP[4:0]			
0x03F1	ADPRE	7:0				PRE[7:0]				
0x03F1	ADPRE	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]
0x03F5	ADCON2	7:0				CALC[2:0]		SOI		TMD[2:0]
0x03F6	ADCON3	7:0								STAT[2:0]
0x03F7	ADSTAT	7:0	AOV	UTHR	LTHR	MATH	NREF			PREF[1:0]
0x03F8	ADREF	7:0								
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
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
Access	R/W	R/W	R/W				DSEN	R/W
Reset	0	0	0				0	0

Bit 7 – PPOL Precharge Polarity

Action During 1 st 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															
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													
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													
Bits 6:4 – CALC[2:0] ADC Error Calculation Mode Select																					
CALC	DSEN = 0 Single-Sample Mode	DSEN = 1 CVD Double-Sample Mode ⁽¹⁾	ADERR																		
111	Reserved	Reserved	Application																		
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⁾																		
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 RES[15:8] 11 10 9 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								
<p>Notes: The individual bytes in this multibyte register can be accessed with the following register names:</p> <ul style="list-style-type: none"> • 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 PCH[5:0] 2 1 0																																																																									
Access	R/W	R/W	R/W	R/W	PCH[5:0]	R/W	R/W	R/W																																																																	
Reset	0	0	0	0	0	0	0	0																																																																	
Bits 5:0 – PCH[5:0] ADC Positive Input Channel Selection																																																																									
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>PCH</th> <th>ADC Positive Channel Input</th> </tr> </thead> <tbody> <tr><td>111111</td><td>Fixed Voltage Reference (FVR) Buffer 2⁽¹⁾</td></tr> <tr><td>111110</td><td>Fixed Voltage Reference (FVR) Buffer 1⁽¹⁾</td></tr> <tr><td>111101</td><td>DAC1 output⁽²⁾</td></tr> <tr><td>111100</td><td>Temperature Indicator⁽³⁾</td></tr> <tr><td>111011</td><td>V_{SS} (Analog Ground)</td></tr> <tr><td>111010-110000</td><td>Reserved. No channel connected.</td></tr> <tr><td>101111</td><td>R7/ANF7⁽⁵⁾</td></tr> <tr><td>101110</td><td>R6/ANF6⁽⁵⁾</td></tr> <tr><td>101101</td><td>R5/ANF5⁽⁵⁾</td></tr> <tr><td>101100</td><td>R4/ANF4⁽⁵⁾</td></tr> <tr><td>101011</td><td>R3/ANF3⁽⁵⁾</td></tr> <tr><td>101010</td><td>R2/ANF2⁽⁵⁾</td></tr> <tr><td>101001</td><td>R1/ANF1⁽⁵⁾</td></tr> <tr><td>101000</td><td>R0/ANF0⁽⁵⁾</td></tr> <tr><td>100111-100011</td><td>Reserved. No channel connected.</td></tr> <tr><td>100010</td><td>R2/ANE2⁽⁴⁾</td></tr> <tr><td>100001</td><td>R1/ANE1⁽⁴⁾</td></tr> <tr><td>100000</td><td>R0/ANE0⁽⁴⁾</td></tr> <tr><td>011111</td><td>RDT/ANDT⁽⁴⁾</td></tr> <tr><td>011110</td><td>RDB/ANDB⁽⁴⁾</td></tr> <tr><td>011101</td><td>RDS/ANDS⁽⁴⁾</td></tr> <tr><td>011100</td><td>RD4/AND4⁽⁴⁾</td></tr> <tr><td>011011</td><td>RD3/AND3⁽⁴⁾</td></tr> <tr><td>011010</td><td>RD2/AND2⁽⁴⁾</td></tr> <tr><td>011001</td><td>RD1/AND1⁽⁴⁾</td></tr> <tr><td>011000</td><td>RD0/AND0⁽⁴⁾</td></tr> <tr><td>010111</td><td>RC7/ANC7</td></tr> <tr><td>010110</td><td>RC6/ANC6</td></tr> <tr><td>010101</td><td>RC5/ANC5</td></tr> <tr><td>010100</td><td>RC4/ANC4</td></tr> <tr><td>010011</td><td>RC3/ANC3</td></tr> <tr><td>010010</td><td>RC2/ANC2</td></tr> <tr><td>010001</td><td>RC1/ANC1</td></tr> <tr><td>010000</td><td>RC0/ANC0</td></tr> <tr><td>001111</td><td>RB7/ANB7</td></tr> </tbody> </table>		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	R7/ANF7 ⁽⁵⁾	101110	R6/ANF6 ⁽⁵⁾	101101	R5/ANF5 ⁽⁵⁾	101100	R4/ANF4 ⁽⁵⁾	101011	R3/ANF3 ⁽⁵⁾	101010	R2/ANF2 ⁽⁵⁾	101001	R1/ANF1 ⁽⁵⁾	101000	R0/ANF0 ⁽⁵⁾	100111-100011	Reserved. No channel connected.	100010	R2/ANE2 ⁽⁴⁾	100001	R1/ANE1 ⁽⁴⁾	100000	R0/ANE0 ⁽⁴⁾	011111	RDT/ANDT ⁽⁴⁾	011110	RDB/ANDB ⁽⁴⁾	011101	RDS/ANDS ⁽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
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	R7/ANF7 ⁽⁵⁾																																																																								
101110	R6/ANF6 ⁽⁵⁾																																																																								
101101	R5/ANF5 ⁽⁵⁾																																																																								
101100	R4/ANF4 ⁽⁵⁾																																																																								
101011	R3/ANF3 ⁽⁵⁾																																																																								
101010	R2/ANF2 ⁽⁵⁾																																																																								
101001	R1/ANF1 ⁽⁵⁾																																																																								
101000	R0/ANF0 ⁽⁵⁾																																																																								
100111-100011	Reserved. No channel connected.																																																																								
100010	R2/ANE2 ⁽⁴⁾																																																																								
100001	R1/ANE1 ⁽⁴⁾																																																																								
100000	R0/ANE0 ⁽⁴⁾																																																																								
011111	RDT/ANDT ⁽⁴⁾																																																																								
011110	RDB/ANDB ⁽⁴⁾																																																																								
011101	RDS/ANDS ⁽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																																																																									
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>PCH</th> <th>ADC Positive Channel Input</th> </tr> </thead> <tbody> <tr><td>001110</td><td>RB6/ANB6</td></tr> <tr><td>001101</td><td>RB5/ANB5</td></tr> <tr><td>001100</td><td>RB4/ANB4</td></tr> <tr><td>001011</td><td>RB3/ANB3</td></tr> <tr><td>001010</td><td>RB2/ANB2</td></tr> <tr><td>001001</td><td>RB1/ANB1</td></tr> <tr><td>001000</td><td>RB0/ANB0</td></tr> <tr><td>000111</td><td>RA7/ANA7</td></tr> <tr><td>000110</td><td>RA6/ANA6</td></tr> <tr><td>000101</td><td>RA5/ANA5</td></tr> <tr><td>000100</td><td>RA4/ANA4</td></tr> <tr><td>000011</td><td>RA3/ANA3</td></tr> <tr><td>000010</td><td>RA2/ANA2</td></tr> <tr><td>000001</td><td>RA1/ANA1</td></tr> <tr><td>000000</td><td>RA0/ANA0</td></tr> </tbody> </table>		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																																								
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:																																																																									
<ol style="list-style-type: none"> 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
Access				NREF			PREF[1:0]	
Reset				R/W			R/W	R/W
				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

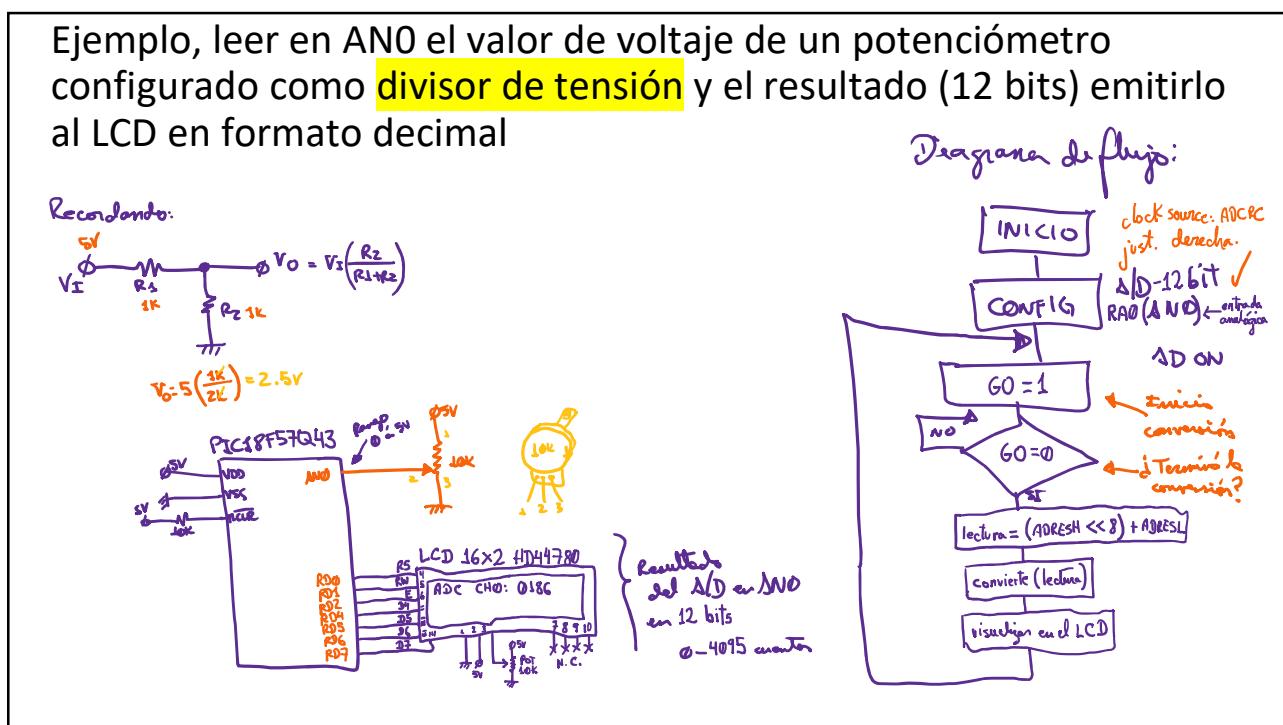
Configuración: (Recordar que el rango de voltaje de entrada es de 0V a 5V)

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

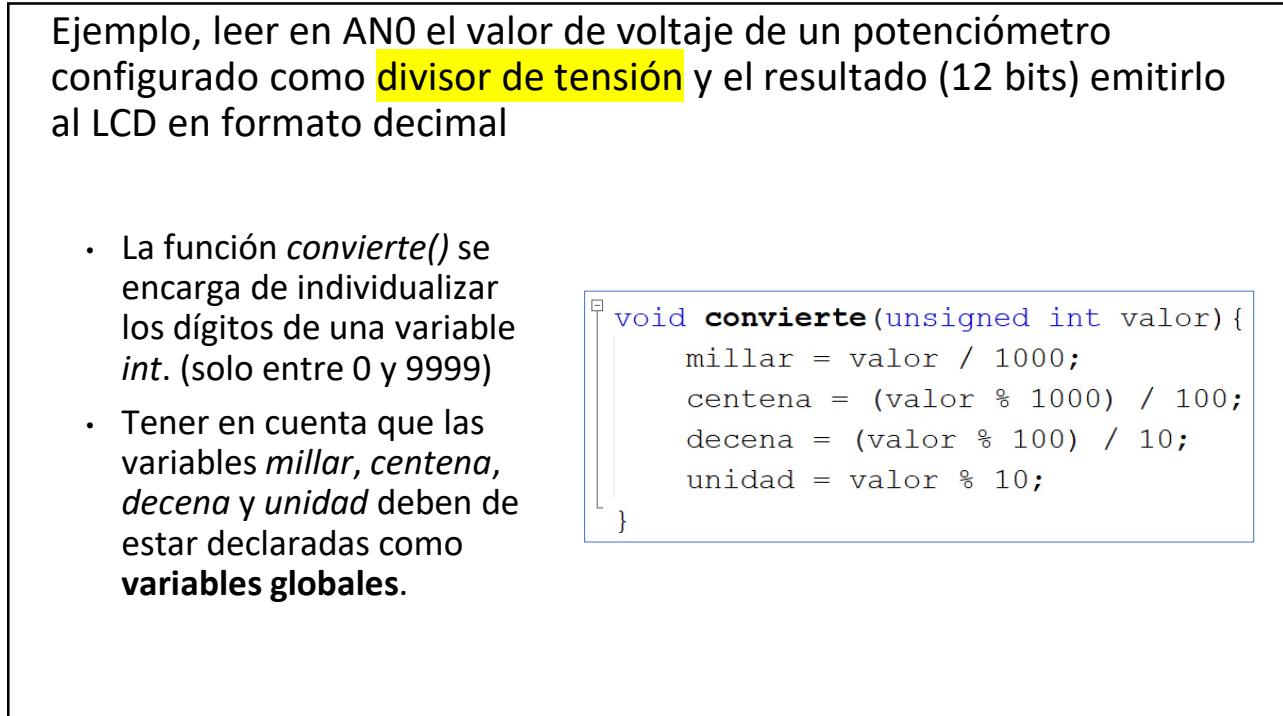
Adquisición (toma una muestra):

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



57



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

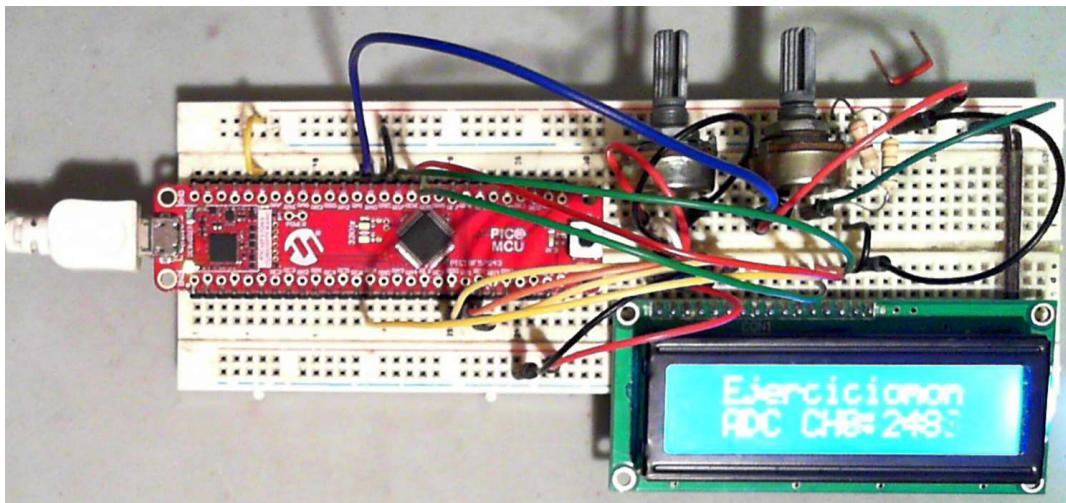
```

1  #include <xc.h>
2  #include "cabecera.h"
3  #include "LCD.h"
4  #define _XTAL_FREQ 48000000UL
5
6  unsigned int resultado = 0; //resultado del ADC en 12bits
7  unsigned int millar, centena, decena, unidad;
8
9  void configuro(void){
10    //configuracion del oscilador
11    OSCCON1 = 0x60; //HFINTOSC, POSTS 1:1
12    OSCFRQ = 0x07; //HFINTOSC a 48MHz
13    OSCEN = 0x40; //HFINTOSC enabled
14
15    //configuracion de E/S
16    TRISBbits.TRISB4 = 1; //RB4 como entrada
17    ANSELBbits.ANSELB4 = 0; //RB4 como digital
18    WPUBbits.WPUB4 = 1; //RB4 con pullup activado
19    TRISFbits.TRISF3 = 0; //RF3 como salida
20    ANSELFbits.ANSELF3 = 0; //RF3 como digital
21
22    //configuracion del ADC
23    ADCON0bits.FM = 1; //right justify
24    ADCON0bits.CS = 1; //ADCRC Clock
25    ADPCH = 0X00; //RA0 is Analog channel
26    TRISAbits.TRISA0 = 1; //Set RA0 to input
27    ANSELAbits.ANSELAO = 1; //Set RA0 to analog
28    ADCON0bits.ON = 1; //Turn ADC On
29
30    void lcd_init(void){
31      TRISD = 0x00; //RD salidas
32      ANSELD = 0x00; //RD digitales
33      LCD_CONFIG();
34      delay_ms(19);
35      BORRAR_LCD();
36      CURSOR_HOME();
37      CURSOR_ONOFF(OFF);
38    }
39
40    void convierte(unsigned int valor){
41      millar = valor / 1000;
42      centena = (valor % 1000) / 100;
43      decena = (valor % 100) / 10;
44      unidad = valor % 10;
45
46    void main(void) {
47      configuro();
48      lcd_init();
49      POS_CURSOR(1,2);
50      ESCRIBE_MENSAJE("Ejerciciomon", 12);
51      while(1){
52        ADCON0bits.GO = 1; //Start conversion
53        while(ADCON0bits.GO); //Wait for conversion done
54        resultado = (ADRESH << 8) + ADRESL;
55        convierte(resultado);
56        POS_CURSOR(2,2);
57        ESCRIBE_MENSAJE("ADC CH0:", 8);
58        ENVIA_CHAR(millar+0x30);
59        ENVIA_CHAR(centena+0x30);
60        ENVIA_CHAR(decena+0x30);
61        ENVIA_CHAR(unidad+0x30);
62      }
63    }

```

59

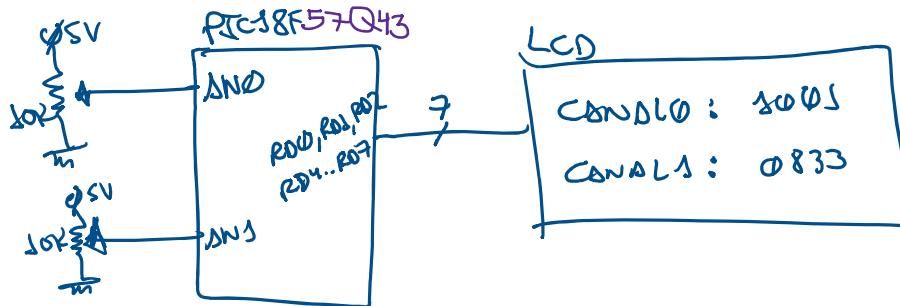
Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal



60

Asignación:

- Leer dos canales analógicos y mostrarlos en el LCD



61

El módulo DAC del PIC18F57Q43

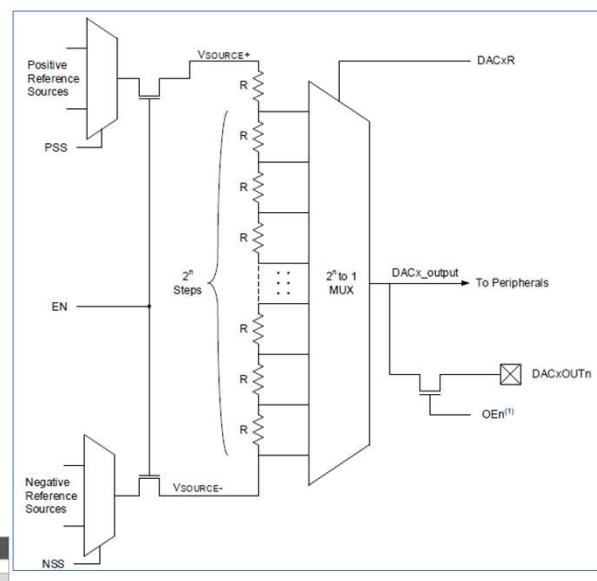
- Referencia: capítulo 41 del datasheet
- Resolución: 8bits
- Tipo arreglo R-2R
- Se puede usar el FVR como voltaje de referencia positivo (bits PSS)
- Fórmula:

$$\text{Equation 41-1. DAC Output Equation}$$

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

- 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]			NS



62

El módulo DAC del PIC18F57Q43

Name:	DACxCON							
Address:	0x7F							
Digital-to-Analog Converter Control Register								
Bit	7	6	5	4	3	2	1	0
Access	EN		OE[1:0]		PSS[1:0]		NSS	
Reset	0		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

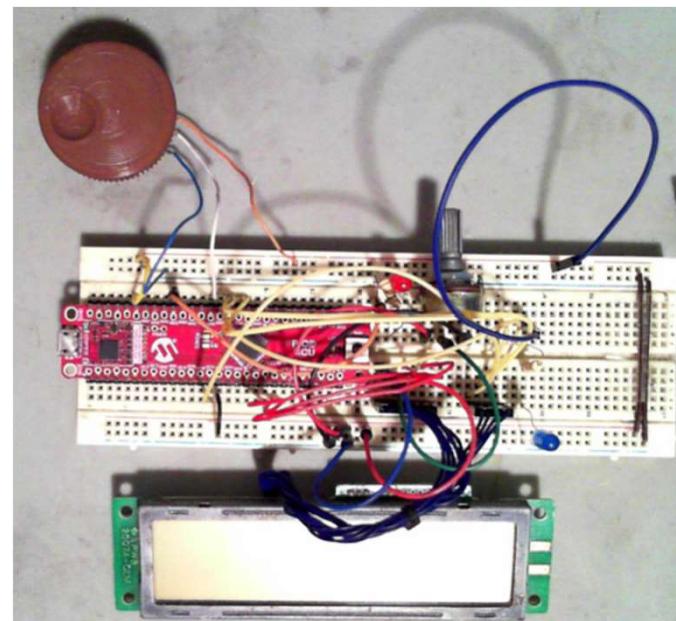
Evidencias de ejemplos 2024-1

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

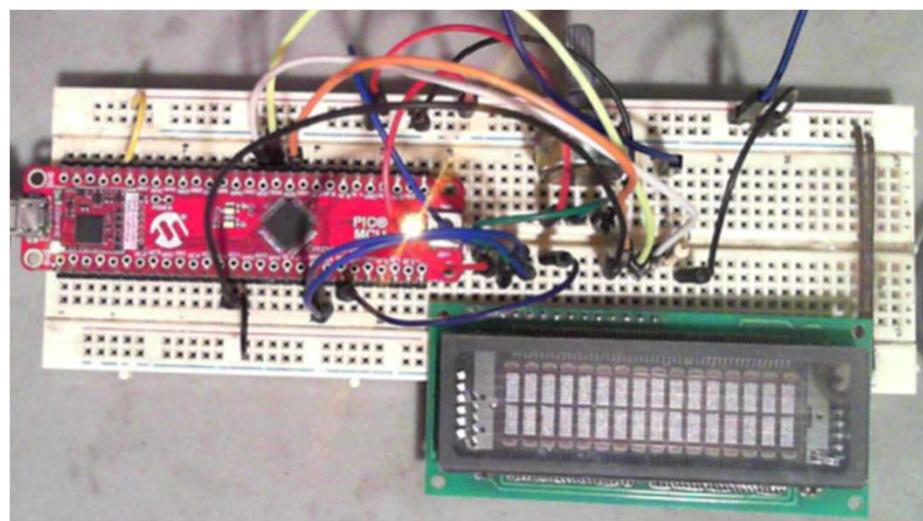
```

1  #include <xc.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define _XTAL_FREQ 48000000UL
6
7  void configuro(){
8      OSCCON1 = 0x60;           //NOSC=HFINTOSC NDIV1:1
9      OSCFRQ = 0x07;           //HFINTOSC 48MHz
10     OSCEN = 0x40;            //HFINTOSC enabled
11     TRISFbits.TRISF3 = 0;    //RF3 output
12     ANSELFbits.ANSELF3 = 0;  //RF3 digital
13 }
14
15 void main(void) {
16     configuro();              //calling configuro function
17     while(1){                  //do it forever
18         LATFbits.LATF3 = 0;    //LED on
19         __delay_ms(200);       //delay of 200ms
20         LATFbits.LATF3 = 1;    //LED off
21         __delay_ms(200);       //delay of 200ms
22     }
23 }
```

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 titilero del LED en RF3.
- Solo titilará el LED si se mantiene presionando el pulsador

```

1  #include <xc.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define _XTAL_FREQ 48000000UL

5
6  void configuro(void) {
7      OSCCON1 = 0x60;          //NOSC=HFINTOSC NDIVl:1
8      OSCFRQ = 0x07;          //HFINTOSC 48MHz
9      OSCEN = 0x40;           //HFINTOSC enabled
10     TRISFBits.TRISF3 = 0;   //RF3 salida
11     ANSELFbits.ANSELF3 = 0; //RF3 digital
12     TRISBbits.TRISB4 = 1;   //RB4 entrada
13     ANSELBbits.ANSELB4 = 0; //RB4 digital
14     WPUBbits.WPUB4 = 1;    //RB4 pullup enabled
15 }

16 void main(void) {
17     configuro();             //calling configuro function
18     while(1){                //do it forever
19         if(PORTBbits.RB4 == 0){
20             LATFBits.LATF3 = 0; //LED on
21             __delay_ms(100);   //delay of 100ms
22             LATFBits.LATF3 = 1; //LED off
23             __delay_ms(100);   //delay of 100ms
24         }
25     }
26     else{
27         LATFBits.LATF3 = 0; //LED on
28         __delay_ms(200);   //delay of 200ms
29         LATFBits.LATF3 = 1; //LED off
30         __delay_ms(200);   //delay of 200ms
31     }
32 }
33 }
```

69

```

1  #include <xc.h>
2  #include "cabecera.h"
3  #define _XTAL_FREQ 48000000UL

4
5  void configuro(void) {
6      OSCCON1 = 0x60;          //NOSC=HFINTOSC NDIVl:1
7      OSCFRQ = 0x07;          //HFINTOSC 48MHz
8      OSCEN = 0x40;           //HFINTOSC enabled
9      TRISFBits.TRISF3 = 0;   //RF3 salida
10     ANSELFbits.ANSELF3 = 0; //RF3 digital
11     TRISBbits.TRISB4 = 1;   //RB4 entrada
12     ANSELBbits.ANSELB4 = 0; //RB4 digital
13     WPUBbits.WPUB4 = 1;    //RB4 pullup enabled
14 }

15 void main(void) {
16     configuro();
17     while(1){
18         if(PORTBbits.RB4 == 0){
19             LATFBits.LATF3 = 0; //LED on
20             __delay_ms(100);
21             LATFBits.LATF3 = 1; //LED off
22             __delay_ms(100);
23         }
24     }
25     else{
26         LATFBits.LATF3 = 0; //LED on
27         __delay_ms(300);
28         LATFBits.LATF3 = 1; //LED off
29         __delay_ms(300);
30     }
31 }
32 }
```

Implementación 2024-1

Modificación del ejemplo inicial:

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

70

Implementación 2024-1

```
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; //detecction falling edge
19     INTOPPS = 0x0C; //ruteando INT0 para RB4
20 }
```

```
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 }
36
37 void _interrupt(irq(IRQ_INTO)) INTO_ISR(void){
38     PIR1bits.INT0IF = 0;      //abajamos 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 está empleando el pulsador integrado para la interrupción externa INT0, para ello que debe de rutaar 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

```
1 #include <xc.h>
2 #include "cabecera.h"
3 #define XTAL_FREQ 48000000UL /*para que el compilador se
4                                     la frq que esta trabajando
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;          //rutando INT0 a RB4
18    INTCON0bits.INT0EDG = 0; //INT0 falling edge
19    PIE1bits.INT0IE = 1;     //habilita INT0
20    INTCON0bits.GIE = 1;     //int global enabled
21 }
```

```
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 }
```

46 void __interrupt(irq(IRQ_INTERRUPT)) INT0_ISR(void){
47 PIR1bits.INT0IF = 0; //abajamos 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 está 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 titilado del LED.

72

Implementación 2024-1

```

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 = 0xC; //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 }

```

```

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     }
38 }
39
40 }

```

```

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 }

```

```

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.

73

Implementación 2024-1

```

1 #include <xc.h>
2 #include "cabecera.h"
3 #define _XTAL_FREQ 48000000UL
4
5 unsigned char indicador = 0;
6 unsigned char velocidad = 0;
7
8 void configuro(void){
9     OSCCON1 = 0x60;
10    OSCFRQ = 0x07;
11    OSCEN = 0x40;
12    TRISBbits.TRISB3 = 0;
13    ANSELBbits.ANSELB3 = 0;
14    TRISBbits.TRISB4 = 1;
15    ANSELBbits.ANSELB4 = 0;
16    WPUBbits.WPUB4 = 1;
17
18    TRISBbits.TRISB3 = 1;
19    ANSELBbits.ANSELB3 = 0;
20    WPUBbits.WPUB3 = 1;
21
22    PIE1bits.INT0IE = 1;
23    PIE1bits.INT1IE = 1;
24    INTCON0bits.GIE = 1;
25    INTCON0bits.INT0EDG = 0;
26    INTCON0bits.INT1EDG = 0;
27    INT0PPS = 0xC;
28    INT1PPS = 0xB;
29 }

```

```

31 void main(void) {
32     configuro();
33     while(1){
34         if(indicador == 0){
35             if(velocidad == 0){
36                 LATFbits.LATF3 = 0; //enciendo
37                 __delay_ms(300); //retardo
38                 LATFbits.LATF3 = 1; //apago el LED
39                 __delay_ms(300); //retardo
40             }
41             else{
42                 LATFbits.LATF3 = 0; //enciendo
43                 __delay_ms(100); //retardo
44                 LATFbits.LATF3 = 1; //apago el LED
45                 __delay_ms(100); //retardo
46             }
47         }
48         else if(indicador == 1){
49             LATFbits.LATF3 = 1; //apago el LED
50         }
51     }
52 }

```

```

54 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
55     PIR1bits.INT0IF = 0; //bajamos bandera
56     if (indicador == 0){
57         indicador = 1;
58     }
59     else{
60         indicador = 0;
61     }
62 }

```

```

64 void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
65     PIR6bits.INT1IF = 0; //bajo bandera INT1IF
66     if (velocidad == 0){
67         velocidad = 1;
68     }
69     else{
70         velocidad = 0;
71     }
72 }
73
74 void __interrupt(irq(default)) default_ISR(void){
75 }
76

```

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

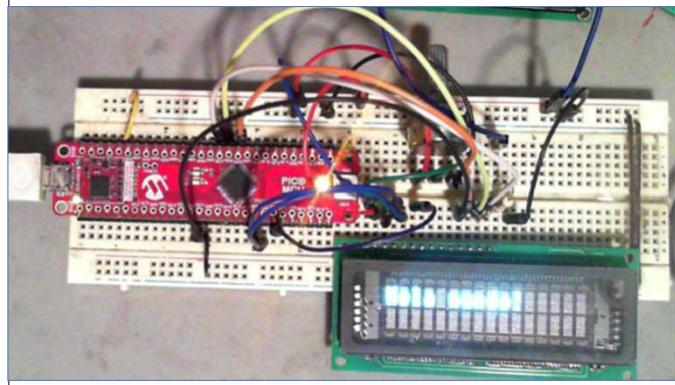
Implementación 2024-1

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "cabecera.h"
4 #include "LCD.h"
5 #define _XTAL_FREQ 48000000UL
6
7 void configuro(void) {
8     OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
9     OSCFRQ = 0x07;       //HFINTOSC=48MHz
10    OSCEN = 0x40;        //HFINTOSC enabled
11    ANSELbbits.ANSELF3 = 0; //RF3 digital
12    TRISFbits.TRISF3 = 0; //RF3 output
13 }
14
15 void main(void) {
16     configuro();
17     LCD_INIT();
18     POS_CURSOR(1,0);
19     ESCRIBE_MENSAJE2("Hola mundo!");
20     while(1){
21         LATFbits.LATF3 = 0; //LED on
22         __delay_ms(300);
23         LATFbits.LATF3 = 1; //LED off
24         __delay_ms(300);
25     }
26 }
```

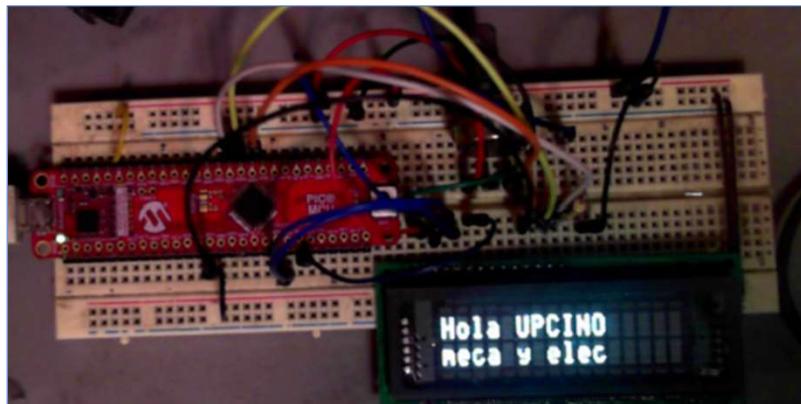
Segunda prueba: Visualizar "Hola mundo!" en el LCD:

- 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



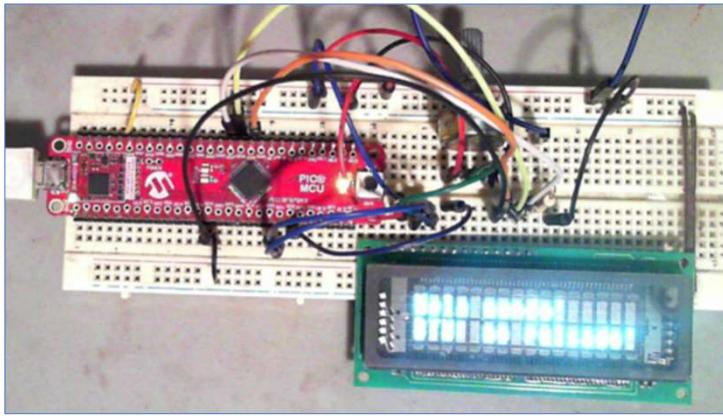
Ejemplo de visualización de mensajes en ambas líneas del LCD:

- 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



```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "cabecera.h"
4 #include "LCD.h"
5 #define _XTAL_FREQ 48000000UL
6
7 void configuro(void){
8     OSCCON1 = 0x60; //NOSC=HFINTOSC NDIV=1:1
9     OSCFRQ = 0x07; //HFINTOSC=48MHz
10    OSCEN = 0x40; //HFINTOSC enabled
11    ANSELFbits.ANSELF3 = 0; //RF3 digital
12    TRISFbits.TRISF3 = 0; //RF3 output
13    ANSELBbits.ANSELB4 = 0; //RB4 digital
14    TRISBbits.TRISB4 = 1; //RB4 input
15    WPUBbits.WPUB4 = 1; //RB4 weak pullup enabled
16 }
17
18 void main(void) {
19     configuro();
20     LCD_INIT();
21     POS_CURSOR(1,0);
22     ESCRIBE_MENSAJE2("Hola mundo!");
23     while(1){
24         if(PORTBbits.RB4 == 0){
25             POS_CURSOR(2,0);
26             ESCRIBE_MENSAJE2("BTN: pulsado   ");
27         }
28         else{
29             POS_CURSOR(2,0);
30             ESCRIBE_MENSAJE2("BTN: no pulsado!");
31         }
32     }
33 }
```

77

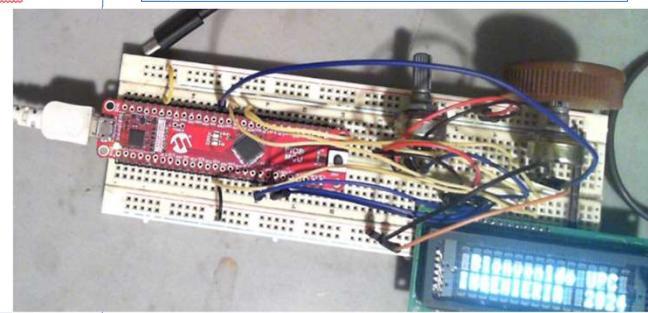
Implementación de “splash screen”

```

7 #include <xc.h>
8 #include "cabecera.h"
9 #include "LCD.h"
10 #define _XTAL_FREQ 32000000UL
11
12 void configuro(void){
13     OSCCON1 = 0x60;
14     OSCFRQ = 0x06; //HFINTOSC a 32MHz
15     OSCEN = 0x40;
16     //configuración del A/D
17     TRISAbits.TRISA0 = 1; //RA0 entrada
18     ANSELAbits.ANSELAO = 1; //RA0 analógica
19     ADCON0 = 0x94; //ADC conv manual, ADCRC, just der, ADC on
20 }
21
22 void LCD_inicia(void){
23     TRISED = 0x00;
24     ANSELD = 0x00;
25     __delay_ms(21);
26     LCD_CONFIG();
27     __delay_ms(22);
28     BORRAR_LCD();
29     CURSOR_HOME();
30     CURSOR_ONOFF(OFF);
31     __delay_ms(100);
32 }
```

```

34 void main(void) {
35     configuro();
36     LCD_inicia();
37     POS_CURSOR(1,1);
38     ESCRIBE_MENSAJE("Bienvenido UPC",14);
39     POS_CURSOR(2,0);
40     ESCRIBE_MENSAJE("INGENIERIA 2024",16);
41     __delay_ms(5000);
42     BORRAR_LCD();
43     while(1){
44     }
45 }
```



78

```

7 #include <xc.h>
8 #include "cabecera.h"
9 #include "LCD.h"
10 #define _XTAL_FREQ 32000000UL
11
12 const unsigned char logo_upc_1[]={0x01,0x03,0x06,0x04,0x04,0x0C,0x18,0x18};
13 const unsigned char logo_upc_2[]={0x11,0x04,0x06,0x0E,0x1C,0x18,0x08,0x06};
14 const unsigned char logo_upc_3[]={0x10,0x18,0x0C,0x04,0x04,0x06,0x03,0x03};
15 const unsigned char logo_upc_4[]={0x18,0x1C,0x0C,0x0C,0x07,0x07,0x01,0x00};
16 const unsigned char logo_upc_5[]={0x03,0x03,0x03,0x06,0x04,0x1F,0x1F,0x0E};
17 const unsigned char logo_upc_6[]={0x03,0x17,0x06,0x06,0x1C,0x1C,0x10,0x00};
18
19 void configuro(void){
20     OSCCON1 = 0x60; //NOSC=HFINTOSC NDIV=1
21     OSCFRQ = 0x06; //HFINTOSC a 32MHz
22     OSCEN = 0x40; //RFINTOSC enabled
23     TRISAbits.TRISA0 = 1; //RA0 entrada
24     ANSELAbits.ANSELA0 = 1; //RA0 analógica
25 }

```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																
22																
23																
24																
25																

```

27 void LCD_init(void){
28     TRISD = 0x00;
29     ANSELD = 0x00;
30     __delay_ms(19);
31     LCD_CONFIG();
32     __delay_ms(21);
33     BORRAR_LCD();
34     CURSOR_HOME();
35     CURSOR_ONOFF(OFF);
36     __delay_ms(100);
37     GENERACARACTER(logo_upc_1, 0);
38     GENERACARACTER(logo_upc_2, 1);
39     GENERACARACTER(logo_upc_3, 2);
40     GENERACARACTER(logo_upc_4, 3);
41     GENERACARACTER(logo_upc_5, 4);
42     GENERACARACTER(logo_upc_6, 5);
43 }

```

```

45 void main(void) {
46     configuro();
47     LCD_init();
48     POS_CURSOR(1,6);
49     ENVIA_CHAR(0);
50     ENVIA_CHAR(1);
51     ENVIA_CHAR(2);
52     POS_CURSOR(2,6);
53     ENVIA_CHAR(3);
54     ENVIA_CHAR(4);
55     ENVIA_CHAR(5);
56     __delay_ms(5000);
57     BORRAR_LCD();
58     while(1){
59         GENERACARACTER(logo_upc_1, 0);
60         GENERACARACTER(logo_upc_2, 1);
61         GENERACARACTER(logo_upc_3, 2);
62         GENERACARACTER(logo_upc_4, 3);
63         GENERACARACTER(logo_upc_5, 4);
64         GENERACARACTER(logo_upc_6, 5);
65     }
66 }

```

79

Implementación ADC 2024-1

```

7 #include <xc.h>
8 #include "cabecera.h"
9 #include "LCD.h"
10 #define _XTAL_FREQ 32000000UL
11
12 void configuro(void){
13     OSCCON1 = 0x60;
14     OSCFRQ = 0x06; //HFINTOSC a 32MHz
15     OSCEN = 0x40;
16     //configuración del ADC
17     TRISAbits.TRISA0 = 1; //RA0 entrada
18     ANSELAbits.ANSELA0 = 1; //RA0 analógica
19     ADCON0 = 0x94; //ADC on, just. derecha,
20 }
21
22 void leer_ADC(void){
23     ADPCH = 0x00; //ANA0 seleccionado
24     ADCON0bits.GO = 1; //iniciamos la toma de muestra
25     while(ADCON0bits.GO == 1);
26     //el resultado está en ADRESH:ADRESL
27 }

```

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

```

29 void main(void) {
30     unsigned int valor_leido = 0;
31     configuro();
32     LCD_INIT();
33     __delay_ms(500);
34     POS_CURSOR(1,1);
35     ESCRIBE_MENSAJE("Ejemplo Sem.10",14);
36     POS_CURSOR(2,1);
37     ENVIA_CHAR(0xE4);
38     ESCRIBE_MENSAJE("C PIC18F57Q43",13);
39     __delay_ms(3000);
40     BORRAR_LCD();
41     while(1){
42         leer_ADC();
43         valor_leido = (ADRESH<<8) + ADRESL;
44         POS_CURSOR(1,0);
45         ESCRIBE_MENSAJE("Lectura AN-0:",13);
46         POS_CURSOR(2,5);
47         ENVIA_CHAR((valor_leido / 10000) + 0x30); //mostrando dig diez millar
48         ENVIA_CHAR(((valor_leido % 10000) / 1000) + 0x30); //muestra dig millar
49         ENVIA_CHAR(((valor_leido % 1000) / 100) + 0x30); //muestra dig centena
50         ENVIA_CHAR(((valor_leido % 100) / 10) + 0x30); //muestra dig decena
51         ENVIA_CHAR((valor_leido % 10) + 0x30); //muestra dig unidad
52     }
53 }

```

80

```

7 #include <xc.h>
8 #include "cabecera.h"
9 #include "LCD.h"
10 #define _XTAL_FREQ 32000000UL
11
12 const unsigned char logo_upc_1[]={0x01,0x03,0x06,0x04,0x04,0x0C,0x18,0x18};
13 const unsigned char logo_upc_2[]={0x11,0x04,0x06,0x0E,0x1C,0x18,0x08,0x06};
14 const unsigned char logo_upc_3[]={0x10,0x10,0x0C,0x04,0x04,0x06,0x03,0x03};
15 const unsigned char logo_upc_4[]={0x18,0x1C,0x0C,0x0C,0x07,0x07,0x01,0x00};
16 const unsigned char logo_upc_5[]={0x03,0x03,0x03,0x06,0x04,0x1F,0x1F,0x0E};
17 const unsigned char logo_upc_6[]={0x03,0x17,0x06,0x06,0x1C,0x1C,0x10,0x00};
18
19 void configuro(void){
20     OSCCON1 = 0x60; //NOSC=HFINTOSC NDIV=1:1
21     OSCFRC = 0x06; //HFINTOSC a 32MHz
22     OSCEN = 0x40; //HFINTOSC enabled
23     //configuración del A/D
24     TRISAbits.TRISA0 = 1; //RA0 entrada
25     ANSELAbits.ANSELA0 = 1; //RA0 analógica
26     ADCON0 = 0x94; //ADON, ADCRC, RIGHT JUST, NO
27 }
28
29 unsigned int lectura_ADC_0(void){
30     ADPCH = 0x00; //Canal RA0/ANA0 seleccionado
31     ADCON0bits.GO = 1; //inicio de conversión
32     while(ADCON0bits.GO == 1); //esperar a que termine
33     return ((ADRESH<<8)+ADRESL);
34 }

```



```

36 void LCD_init(void){
37     TRISE = 0x00;
38     ANSELE = 0x00;
39     __delay_ms(19);
40     LCD_CONFIG();
41     __delay_ms(21);
42     BORRAR_LCD();
43     CURSOR_HOME();
44     CURSOR_ONOFF(OFF);
45     __delay_ms(100);
46     GENERACARACTER(logo_upc_1, 0);
47     GENERACARACTER(logo_upc_2, 1);
48     GENERACARACTER(logo_upc_3, 2);
49     GENERACARACTER(logo_upc_4, 3);
50     GENERACARACTER(logo_upc_5, 4);
51     GENERACARACTER(logo_upc_6, 5);
52 }

```

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

```

54 void main(void) {
55     unsigned int resultado;
56     configuro();
57     LCD_init();
58     POS_CURSOR(1,6);
59     ENVIA_CHAR(0);
60     ENVIA_CHAR(1);
61     ENVIA_CHAR(2);
62     POS_CURSOR(2,6);
63     ENVIA_CHAR(3);
64     ENVIA_CHAR(4);
65     ENVIA_CHAR(5);
66     __delay_ms(5000);
67     BORRAR_LCD();
68     while(1){
69         POS_CURSOR(1,0);
70         ESCRIBE_MENSAJE("Lectura RA0:",12);
71         resultado = lectura_ADC_0();
72         POS_CURSOR(2,5);
73         ENVIA_CHAR((resultado / 10000) + 0x30); //vis diez miles
74         ENVIA_CHAR(((resultado % 10000) / 1000) + 0x30); //vis millar
75         ENVIA_CHAR(((resultado % 1000) / 100) + 0x30); //vis centena
76         ENVIA_CHAR(((resultado % 100) / 10) + 0x30); //vis decena
77         ENVIA_CHAR((resultado % 10) + 0x30); //vis unidad
78     }
79 }

```

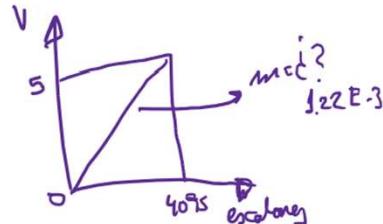
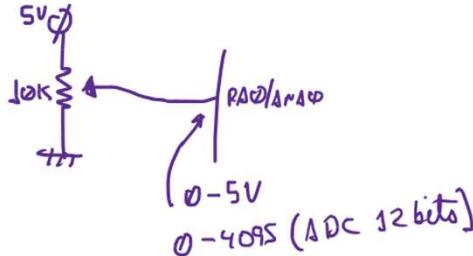
EL54 2024_1

Sem10

81

Implementación ADC 2024-1

Deseo visualizar el voltaje que recibe RA0/ANA0



82

EL54 2024_1 Sem10

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

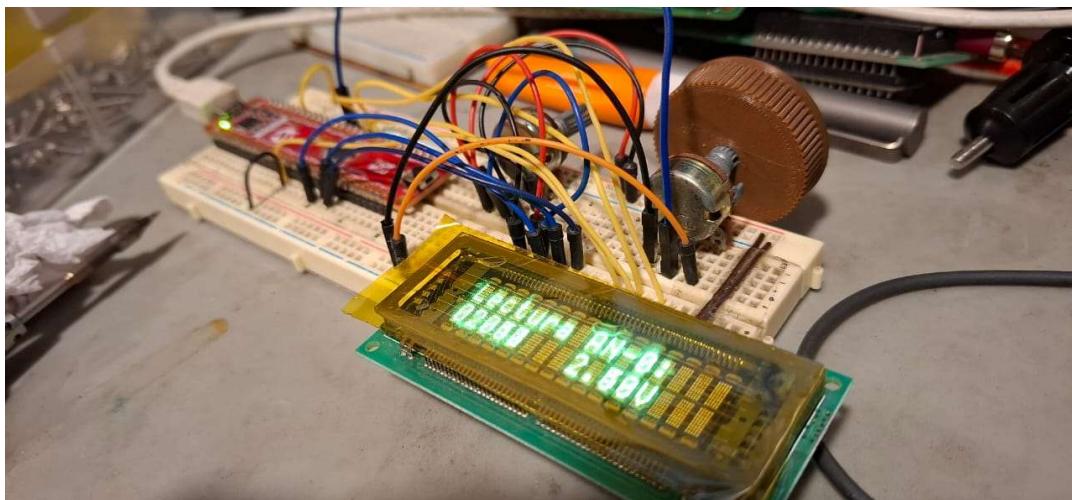
```

54 void main(void) {
55     unsigned int resultado;
56     float calculo;
57     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
71     while(1){
72         POS_CURSOR(1,0);
73         ESCRIBE_MENSAJE("Lectura RA0:",12);
74         resultado = lectura_ADC_0();
75         POS_CURSOR(2,0);
76         ENVIA_CHAR((resultado / 10000) + 0x30); //viz diez millar
77         ENVIA_CHAR(((resultado % 10000) / 1000) + 0x30); //viz millar
78         ENVIA_CHAR(((resultado % 1000) / 100) + 0x30); //viz centena
79         ENVIA_CHAR(((resultado % 100) / 10) + 0x30); //viz decena
80         ENVIA_CHAR((resultado % 10) + 0x30); //viz unidad
81         calculo = resultado * 0.122;
82         voltaje = calculo; //convierte float a integer
83         POS_CURSOR(2,8);
84         ENVIA_CHAR(((voltaje % 1000) / 100) + 0x30); //viz centena
85         ENVIA_CHAR('.');
86         ENVIA_CHAR(((voltaje % 100) / 10) + 0x30); //viz decena
87         ENVIA_CHAR((voltaje % 10) + 0x30); //viz unidad
88         ENVIA_CHAR('V');
89     }

```

83

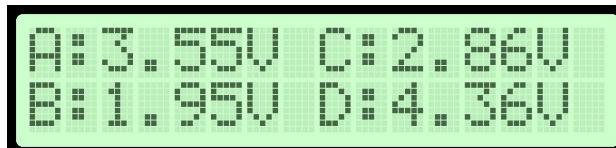
Implementación 2024-1



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:

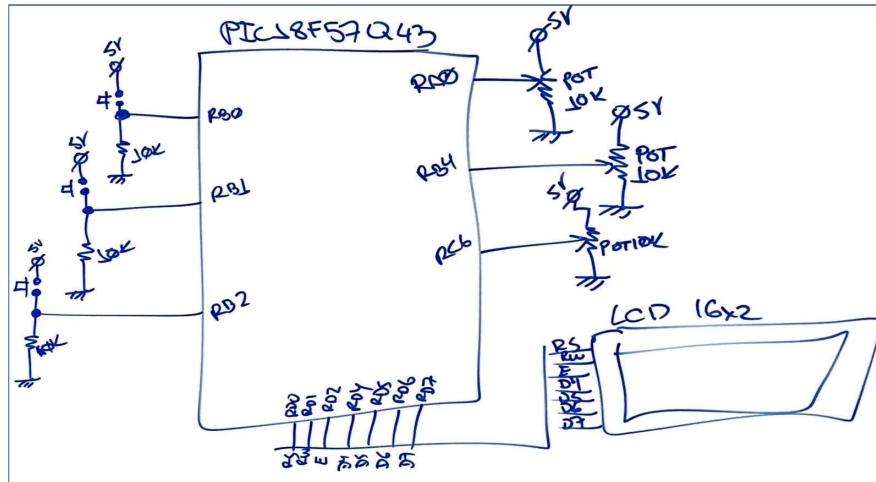


- El intercambio de pantallas será al pulsar un botón activo en alto en INT0 para que se visualice la pantalla de RA0, al pulsar el botón activo en alto en INT1 para que se visualice la pantalla de RB4 y al pulsar el botón activo en alto en INT2 para visualizar la pantalla de RC6
- Carpeta compartida: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

```

7 #include <xc.h>
8 #include "cabecera.h"
9 #include "LCD.h"
10 #define _XTAL_FREQ 32000000UL
11
12 unsigned char pantalla = 0;
13
14 void configuro(void){
15     OSCCON1 = 0x06; //NOSEC=HFINTOSC, NDIV=1:1
16     OSCFRC = 0x06; //HFINTOSC a 32MHz
17     OSCEN = 0x40; //HFINTOSC enabled
18     //config de las I/O
19     TRISBbits.TRISB0 = 1; //RA0 entrada
20     ANSELAbits.ANSELB0 = 1; //RA0 analógica
21     TRISB = 0xFF; //RB todos entradas
22     ANSELB = 0x1E; //RB4 analógica, RB2 RB1 RB0 digitales
23     TRISCbits.TRISC6 = 1; //RC6 entrada
24     ANSELBbits.ANSELB6 = 1; //RC6 analógica
25     //configuración de ADC
26     ADCON0 = 0x44;
27     //config de las ints
28     PIE1bits.INT0IE = 1; //delay_ms(31);
29     PIE6bits.INT1IE = 1; //LCD_CONFIG();
30     PIE10bits.INT2IE = 1; //BORRAR_LCD();
31     INTCON0bits.GIE = 1; //CURSOR_HOME();
32 }
33
34 void LCD_inicializa(void){
35     TRISD = 0x00;
36     ANSELD = 0x00;
37     delay_ms(31);
38     LCD_CONFIG();
39     delay_ms(19);
40     BORRAR_LCD();
41     CURSOR_HOME();
42     CURSOR_ONOFF(GFF);
43     delay_ms(100);
44 }
45
46 void splash_screen(void){
47     unsigned char splash_l[] = {"HOLA UPCINO!!!"};
48     unsigned char var_x;
49     POS_CURSOR(1,1);
50     for(var_x=0;var_x<14;var_x++){
51         ENVIA_CHAR(splash_l[var_x]);
52         delay_ms(100);
53     }
54     POS_CURSOR(2,0);
55     ESCRIBE_MENSAJE("loading",7);
56     for(var_x=0;var_x<7;var_x++){
57         ENVIA_CHAR('.');
58         delay_ms(100);
59     }
60     ESCRIBE_MENSAJE("OK",2);
61     delay_ms(5000);
62     BORRAR_LCD();
63 }
64
65 unsigned int tomamuestra_ADC(unsigned char canal){
66     ADCPCH = canal; //canal RA0
67     ADCON0bits.GO = 1; //inicio conversion
68     while(ADCON0bits.GO == 1); //espero a que termine de convertir
69     return((ADRESH << 8) + ADRESL);
70 }
71
72 void main(void) {
73     unsigned int lectura;
74     float calculo;
75     unsigned int voltaje;
76     configuro();
77     LCD_inicializa();
78     splash_screen();
79     while(1){
80         switch(pantalla){
81             case 0:
82                 POS_CURSOR(1,0);
83                 ESCRIBE_MENSAJE("Channel RA0:",12);
84                 POS_CURSOR(2,0);
85                 ESCRIBE_MENSAJE("Voltage: ", 9);
86                 lectura = tomamuestra_ADC(0x00); //toma muestra de RA0
87                 calculo = lectura * 0.122;
88                 voltaje = calculo;
89                 ENVIA_CHAR((voltaje % 1000) / 100) + 0x30); //muestra centena
90                 ENVIA_CHAR('.');
91                 ENVIA_CHAR((lectura % 100) / 10) + 0x30); //muestra decena
92                 ENVIA_CHAR((lectura % 10) + 0x30); //muestra unidad
93                 ENVIA_CHAR('V');
94                 break;
95             case 1:
96                 POS_CURSOR(1,0);
97                 ESCRIBE_MENSAJE("Channel RB4:",12);
98                 POS_CURSOR(2,0);
99                 ESCRIBE_MENSAJE("Voltage: ", 9);
100                lectura = tomamuestra_ADC(0x00); //toma muestra de RB4
101                calculo = lectura * 0.122;
102                voltaje = calculo;
103                ENVIA_CHAR((voltaje % 1000) / 100) + 0x30); //muestra centena
104                ENVIA_CHAR('.');
105                ENVIA_CHAR((lectura % 100) / 10) + 0x30); //muestra decena
106                ENVIA_CHAR((lectura % 10) + 0x30); //muestra unidad
107                ENVIA_CHAR('V');
108                break;
109            case 2:
110                POS_CURSOR(1,0);
111                ESCRIBE_MENSAJE("Channel RC6:",12);
112                POS_CURSOR(2,0);
113                ESCRIBE_MENSAJE("Voltage: ", 9);
114                lectura = tomamuestra_ADC(0x16); //toma muestra de RC6
115                calculo = lectura * 0.122;
116                voltaje = calculo;
117                ENVIA_CHAR((voltaje % 1000) / 100) + 0x30); //muestra centena
118                ENVIA_CHAR('.');
119                ENVIA_CHAR((lectura % 100) / 10) + 0x30); //muestra decena
120                ENVIA_CHAR((lectura % 10) + 0x30); //muestra unidad
121                ENVIA_CHAR('V');
122                break;
123        }
124    }
125 }
126
127 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void)
128 {
129     PIR1bits.INT0IF = 0;
130     pantalla = 0;
131 }
132
133 void __interrupt(irq(IRQ_INT1)) INT1_ISR(void)
134 {
135     PIR6bits.INT1IF = 0;
136     pantalla = 1;
137 }
138
139 void __interrupt(irq(IRQ_INT2)) INT2_ISR(void)
140 {
141     PIR10bits.INT2IF = 0;
142     pantalla = 2;
143 }
144

```

Solución

88

Anexos: Ejemplos similares a los anteriores

89

Implementación 2024-1

```

1  #include <xc.h>
2  #include "cabecera.h"
3  #include "LCD.h"
4
5  #define _XTAL_FREQ 48000000UL /*para que el compilador sepa
6  | la frq que esta trabajando el MCU*/
7
8  unsigned char estado_LED = 0;
9  unsigned char corazon[]={0x00,0xA,0x15,0x11,0xA,0x4,0x0};
10
11 void configuro(void){
12     OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
13     OSCFRQ = 0x07;      //HFINTOSC 48MHz
14     OSCEN = 0x40;       //HFINTOSC enabled
15     TRISBbits.TRISB4 = 1; //RB4 entrada
16     ANSELBbits.ANSELB4 = 0; //RB4 digital
17     WPUBbits.WPUB4 = 1; //RB4 pullup enabled
18     TRISFbits.TRISF3 = 0; //RF3 salida
19     ANSELFbits.ANSELF3 = 0; //RF3 digital
20     INTOPPS = 0x0C; //ruteando INTO a RB4
21     INTCON0bits.INTOEDG = 0; //INTO falling edge
22     PIE1bits.INTOIE = 1; //habilita INTO
23     INTCON0bits.GIE = 1; //int global enabled
24 }

```

```

26 void LCD_init(void){
27     TRISD = 0x00;
28     ANSELD = 0x00;
29     __delay_ms(21);
30     LCD_CONFIG();
31     __delay_ms(23);
32     BORRAR_LCD();
33     CURSOR_HOME();
34     CURSOR_ONOFF(OFF);
35     GENERARCARACTER(corazon,0);
36 }
37
38 void main(void) {
39     configuro();
40     LCD_init();
41     POS_CURSOR(1,0);
42     ESCRIBIR_MENSAJE("Hola mundo!", 11);
43     POS_CURSOR(2,0);
44     ESCRIBIR_MENSAJE("Soy UPCINO ", 11);
45     ENVIA_CHAR(0);
46
47     while(1){
48
49     }
50 }
51
52 void __interrupt(irq(IRQ_INTERRUPT)) INTO_ISR(void){
53     PIR1bits.INTOIF = 0; //bajamos bandera INTOIF
54 }
55
56 void __interrupt(irq(default)) default_ISR(void){
57 }

```

90

Implementación 2024-1

```

1 #include <xc.h>
2 #include "cabecera.h"
3 #define _XTAL_FREQ 48000000UL
4
5 #define LED_integrado LATFbits.LATF3
6 #define LED_apagado 0
7 #define LED_parpadeo 1
8
9 unsigned char LED_estado = LED_apagado;
10
11 void configuro(void){
12     OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1
13     OSCFRQ = 0x07;      //HFINTOSC 48MHz
14     OSCEN = 0x40;       //HFINTOSC enabled
15     TRISFbits.TRISF3 = 0; //RF3 salida
16     ANSELFbits.ANSELF3 = 0; //RF3 digital
17     TRISBbits.TRISB4 = 1; //RB4 entrada
18     ANSELBbits.ANSELB4 = 0; //RB4 digital
19     WPUBbits.WPUB4 = 1; //RB4 pullup enable
20     INTOPPS = 0x0C; //redirigir señal
21     INTCON0bits.INT0EDG = 0; //INT0 detección
22     PIE1bits.INT0IE = 1; //INT0 habilitada
23     INTCON0bits.GIE = 1; //switch global de ints habilitado
24 }
25
26 void main(void) {
27     configuro();
28     while(1){
29         if(LED_estado == LED_parpadeo){ //pregunto si presione BTN
30             LED_integrado = 0; //LED on
31             __delay_ms(400); //retardo de 400ms
32             LED_integrado = 1; //LED off
33             __delay_ms(400); //retardo de 400ms
34         }
35         else if(LED_estado == LED_apagado){
36             LED_integrado = 1; //LED off
37         }
38     }
39 }
40
41 void __interrupt(irq(IRQ_INTO)) INTO_ISR(void){
42     PIR1bits.INT0IF = 0; //abajamos la bandera INT0IF
43     if(LED_estado == LED_apagado){
44         LED_estado = LED_parpadeo;
45     }
46     else{
47         LED_estado = LED_apagado;
48     }
49 }
50
51 void __interrupt(irq(default)) default_ISR(void){
52
53 }

```

91

```

1 #include <xc.h>
2 #include "cabecera.h"
3 #include "LCD.h"
4 #define _XTAL_FREQ 48000000UL
5
6 #define LED_integrado LATFbits.LATF3
7 #define LED_apagado 0
8 #define LED_parpadeo 1
9
10 unsigned char LED_estado = LED_apagado;
11
12 void configuro(void){
13     OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1
14     OSCFRQ = 0x07;      //HFINTOSC 48MHz
15     OSCEN = 0x40;       //HFINTOSC enabled
16     TRISFbits.TRISF3 = 0; //RF3 salida
17     ANSELFbits.ANSELF3 = 0; //RF3 digital
18     TRISBbits.TRISB4 = 1; //RB4 entrada
19     ANSELBbits.ANSELB4 = 0; //RB4 digital
20     WPUBbits.WPUB4 = 1; //RB4 pullup enable
21     INTOPPS = 0x0C; //redirigir señal
22     INTCON0bits.INT0EDG = 0; //INT0 detección
23     PIE1bits.INT0IE = 1; //INT0 habilitada
24     INTCON0bits.GIE = 1; //switch global de ints habilitado
25 }
26
27 void LCD_init(void){
28     TRISD = 0x00;
29     ANSELD = 0x00;
30     __delay_ms(18);
31     LCD_CONFIG();
32     __delay_ms(19);
33     BORRAR_LCD();
34     CURSOR_HOME();
35     CURSOR_ONOFF(OFF);
36 }
37
38 void main(void) {
39     configuro();
40     LCD_init();
41     POS_CURSOR(1,0);
42     ESCRIBE_MENSAJE("Hola mundo!", 11);
43     POS_CURSOR(2,0);
44     ESCRIBE_MENSAJE("Soy UPCINO!", 11);
45     while(1){
46         if(LED_estado == LED_parpadeo){ //pregunto si presione BTN
47             LED_integrado = 0; //LED on
48             __delay_ms(400); //retardo de 400ms
49             LED_integrado = 1; //LED off
50             __delay_ms(400); //retardo de 400ms
51         }
52         else if(LED_estado == LED_apagado){
53             LED_integrado = 1; //LED off
54         }
55     }
56 }

```

Implementación 2024-1

92

Fin de la sesión