

Microcontroladores

Profesor: Kalun José Lau Gan

Semestre 2024-2

Semana 9-10

1

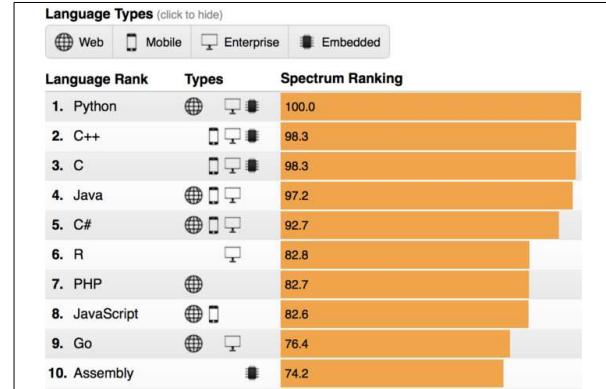
Coordinaciones previas:

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

2

¿Preguntas previas?

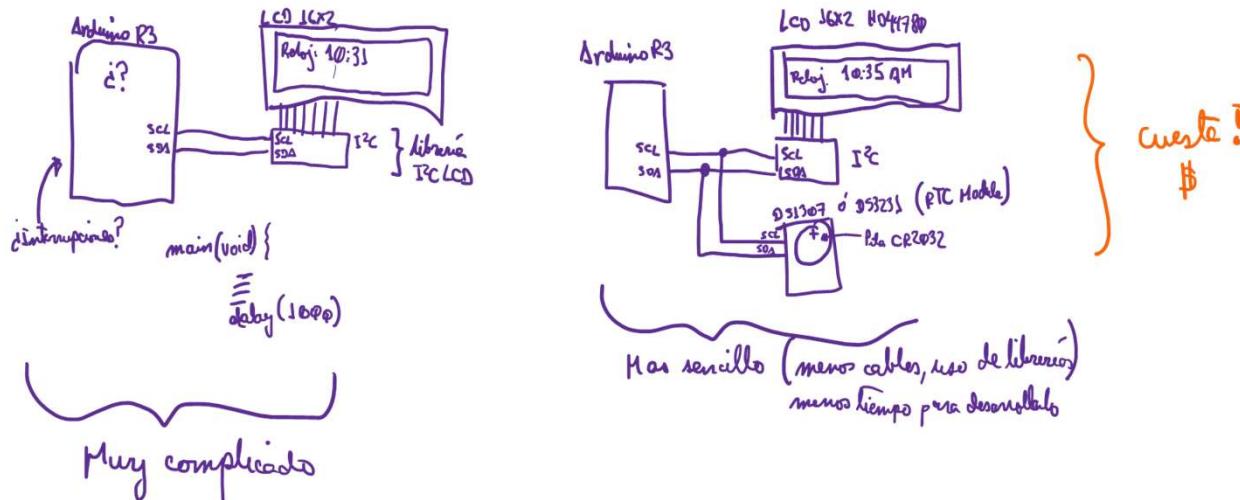
- ¿El hecho de usar C lo hace más fácil?
 - El proceso de diseño es el mismo tanto para lenguaje Assembler como en lenguaje C, se tiene que implementar primero el hardware, desarrollar el algoritmo y finalmente aterrizar el algoritmo en un lenguaje de programación.
 - El lenguaje XC va a permitir el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



3

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

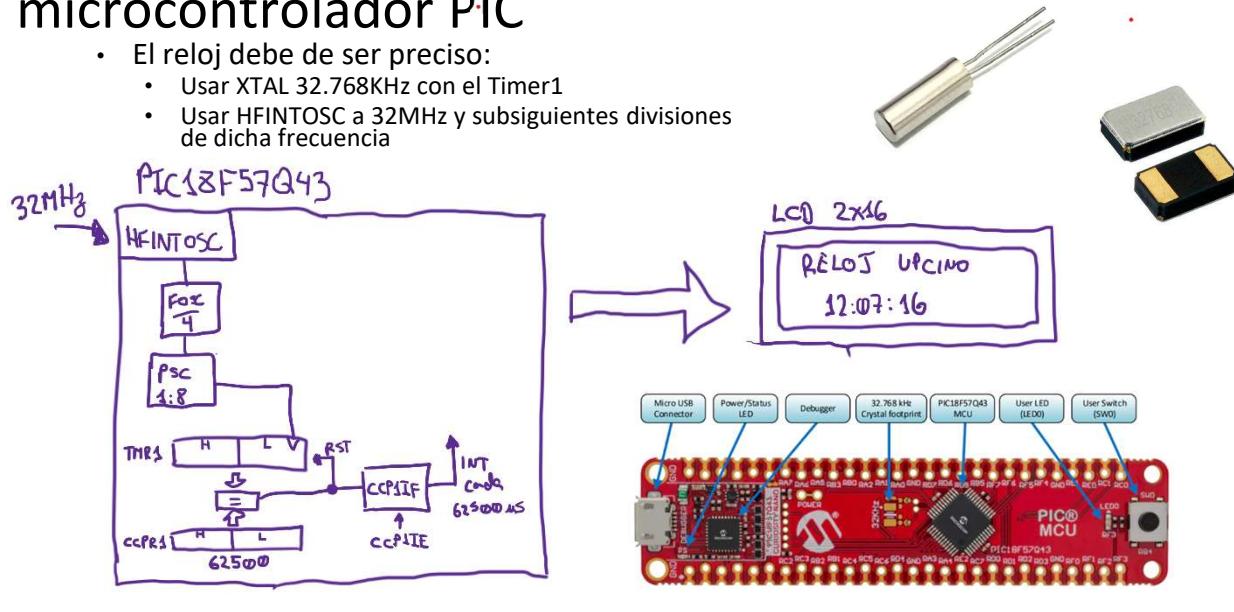
- El reloj debe de ser preciso



4

Ahora, vemos como hacer un reloj con microcontrolador PIC

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



5

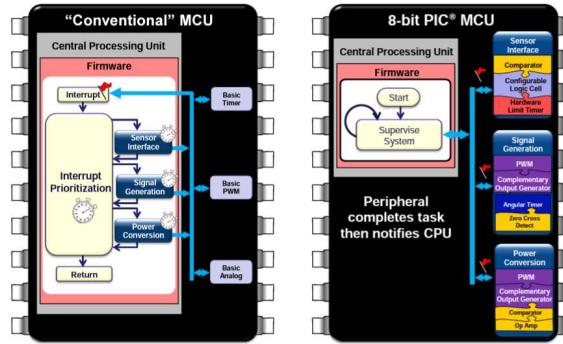
Agenda

- 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 (Microchip Code Configurator)
- Mi primer proyecto en lenguaje XC8 de alto nivel en el MPLAB X IDE
- Uso del LCD alfanumérico 16x2 HD44780
- El conversor ADC del PIC18F57Q43
- El conversor DAC del PIC18F57Q43

6

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

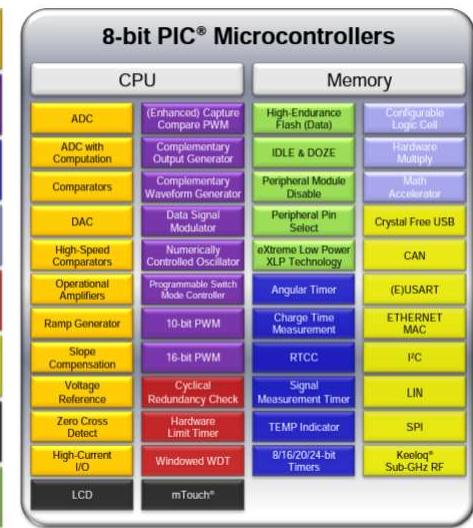
- Independizando el funcionamiento de los periféricos en el microcontrolador se logra un mejor rendimiento del CPU
- Los periféricos independientes administran sus tareas sin código ni supervisión del CPU para mantener su funcionamiento



7

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

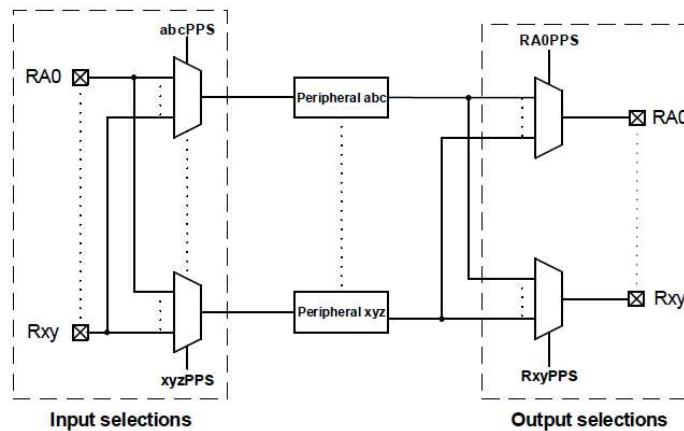
- Dependiendo del modelo y familia el microcontrolador tendrá un conjunto de periféricos CIP
- Los CIPs poseen conexiones adicionales con otros CIP para lograr la independencia del CPU
- Estos CIP se apoyan en el sistema de interconexión de pines (PPS)



8

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



9

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.

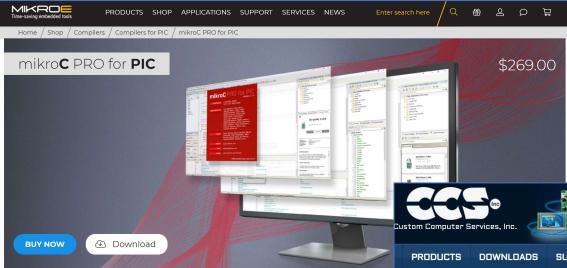
Problema de compatibilidad entre distintas plataformas de desarrollo aún empleando un mismo lenguaje.

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



10

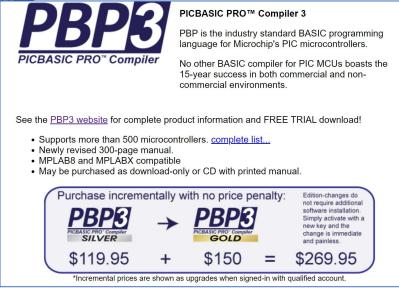
Panorama de los lenguajes de alto nivel para microcontroladores



mikroC PRO for PIC
\$269.00
BUY NOW Download



CCS Custom Computer Services, Inc.
PRODUCTS DOWNLOADS SUPPORT/RESOURCES PURCHASE DESIGN SERVICES
Buy - PCWHD IDE Compiler for Microchip PIC10/12/16/18/24/dsPIC Devices
SKU: 52202-588
PIC10, PIC12, 12-bit Instructions 8-bit PIC® MCUs
PIC16, PIC18, 16-bit Instructions 8-bit PIC® MCUs
PIC24, 24-bit Instructions 16-bit PIC® MCUs
POW PCWH PCWHD
In stock (ships immediately)
Download version available \$600.00 Add to Cart
Starting at \$100 more, buy a complete development kit

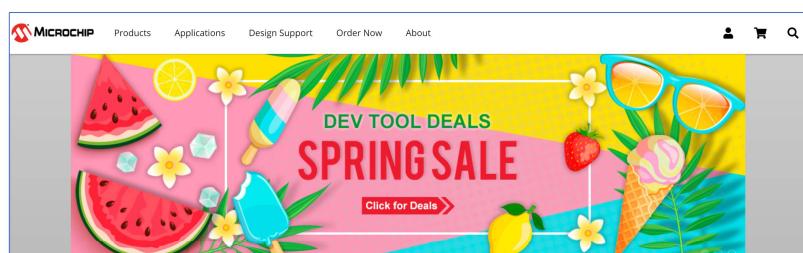


PBP3 PICBASIC PRO™ Compiler 3
PBP is the industry standard BASIC programming language for Microchip's PIC microcontrollers.
No other BASIC compiler for PIC MCUs boasts the 15-year success in both commercial and non-commercial environments.
See the [PBP3 website](#) for complete product information and 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:
PBP3 SILVER → **PBP3 GOLD**
\$119.95 + \$150 = \$269.95
*Incremental prices are shown as upgrades when signed-in with qualified account.

11

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



MICROCHIP Products Applications Design Support Order Now About
DEV TOOL DEALS SPRING SALE Click for Deals
Home / MPLAB / MPLAB® XC Compilers

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:

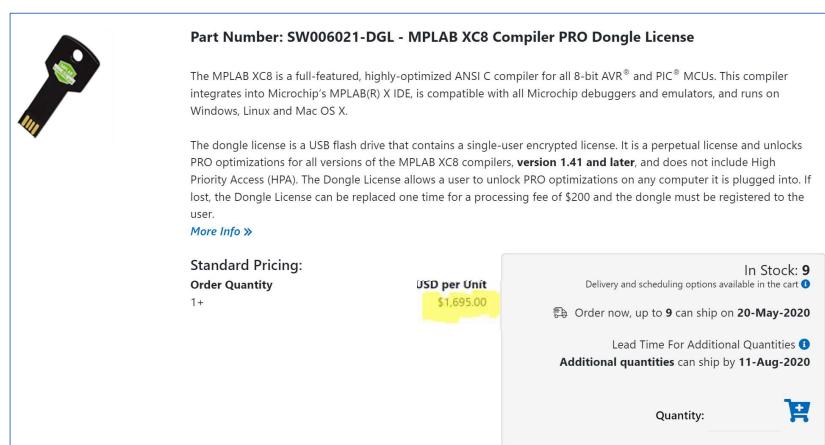
- MPLAB XC8 supports all 8-bit PIC® and AVR® microcontrollers (MCUs)
- MPLAB XC16 supports all 16-bit PIC MCUs and dsPIC® Digital Signal Controllers (DSCs)
- MPLAB XC32/32++ supports all 32-bit PIC and SAM MCUs and MPUs



12

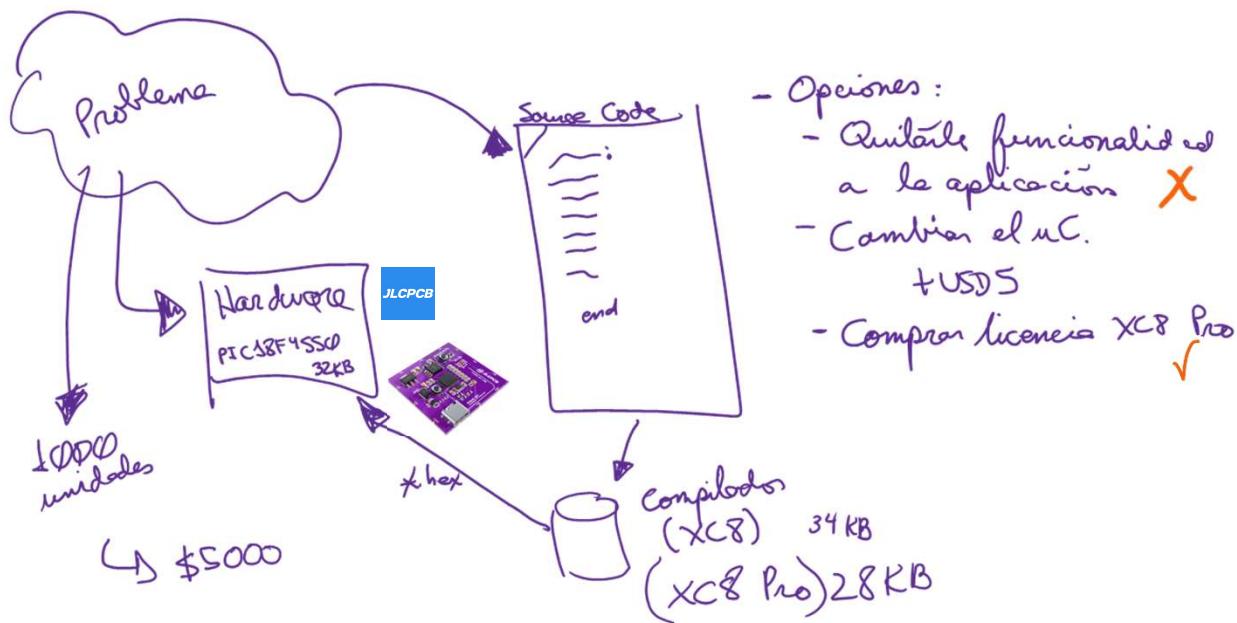
El compilador XC de Microchip

- 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.



13

Caso:



14

Optimización en el compilador XC

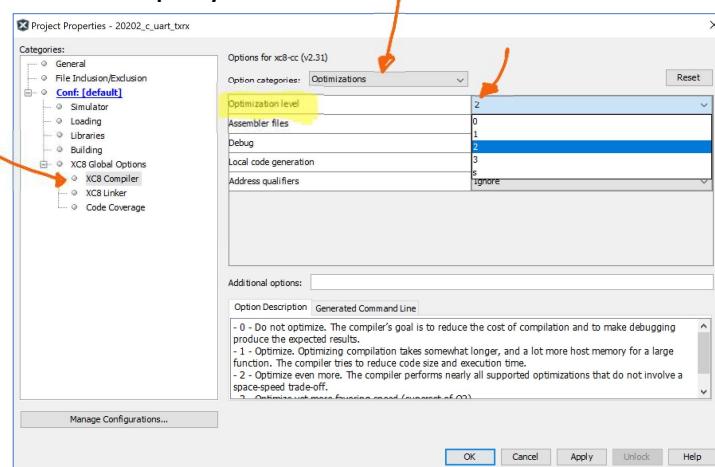
- **Optimization levels:**
 - Level 0: Fastest compilation time, minimal optimizations
 - Level 1: Optimizations with low debugging impact
 - Level 2: All optimizations that give the best balance between speed and size
 - Level 3: Program execution will be as fast as possible
 - Level s: Code size will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

15

Optimización en el compilador XC

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



16

Eficiencia de código en Assembler vs C

¿Por qué Assembler y por qué C?



17

El MPLAB Xpress

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



18

El MPLAB Xpress

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

```

12 // 'C' source line pragma statements
13 #pragma config PLLDIV = 1           // PLL Prescaler Selection bits (No prescale (a PLL 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 USOZDIV = 1          // USB Clock Selection bit (used in Full-Speed USB mode only; UCFQUSOZ = 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 CCPDPIX = ON         // CCP2 PUX 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; REE Input pin disabled)
25 #pragma config IUP = OFF             // Single-Supply ICEP Faultie Bit (Single-Supply ICEP disabled)
26
27 #include <xc8.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     //Configuracion de RDB como salida
33     TRISD = ~0xF0;
34 }
35
36 - void main(void){                //Llamada a la función configuración
37     configuration();               //Puede infinito
38     while(1){                      //Pone a uno Logico el RDO
39         LATDbits.LD0 = 1;           //Retardo de 250ms
40         delay_ms(250);            //Pone a cero Logico el RDO
41         LATDbits.LD0 = 0;           //Retardo de 250ms
42         delay_ms(250);
43     }
44 }
45
46 }

```

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

19

El Microchip Code Configurator (MCC)



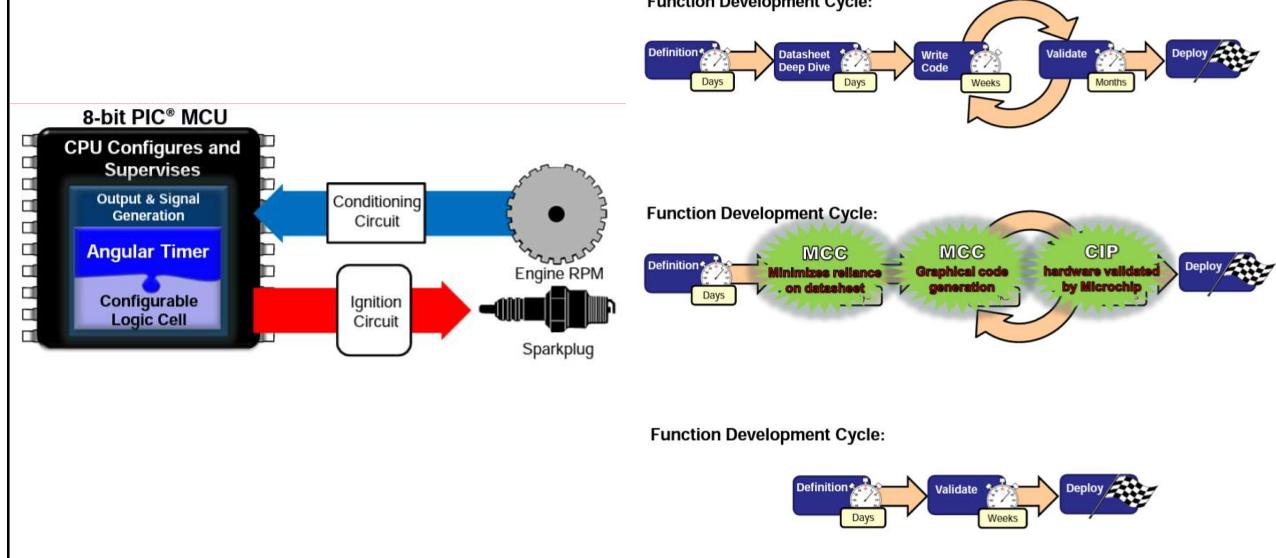
- Entorno de programación/configuración gráfico
- Interface intuitiva para desarrollos rápidos de prototipos
- Configuración automatizada para periféricos y funciones
 - **Minimiza el uso de la hoja técnica para configuraciones**
 - Reduce el esfuerzo y tiempo dedicado a la configuración
- Como desventaja: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

20

El Microchip Code Configurator (MCC)



- Caso de uso



21

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

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

22

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.

```

Arduino/Genuino Uno on /dev/cu.usbmodem1421

23

Plantilla de código en XC8:

```

#include <xc.h>

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

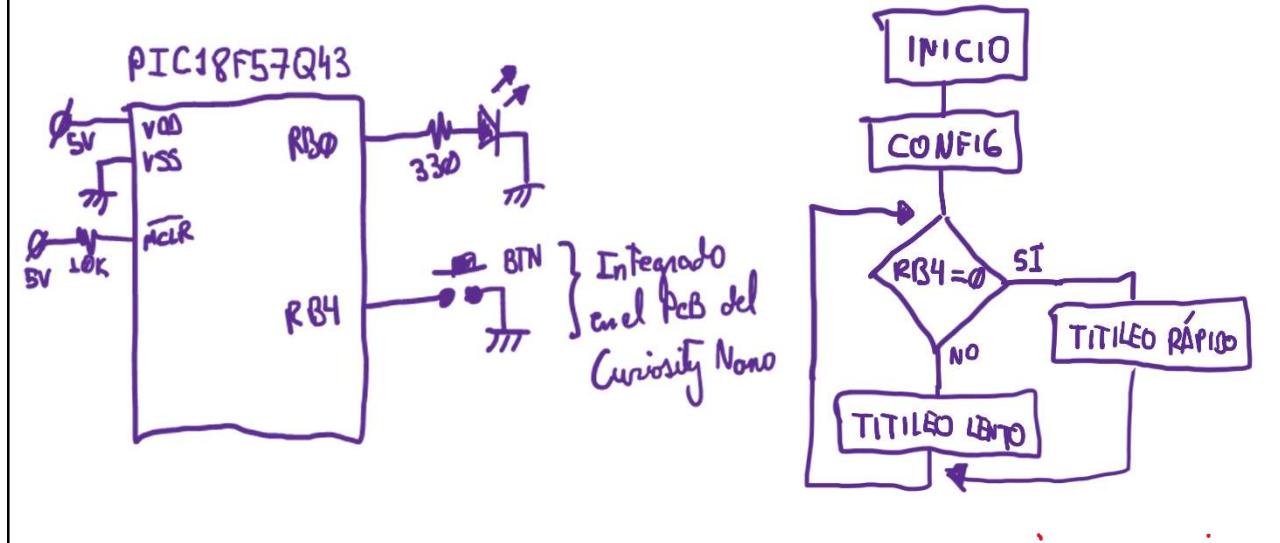
void configuración(void) {
  //Aqui colocas las configuraciones iniciales
}

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

```

24

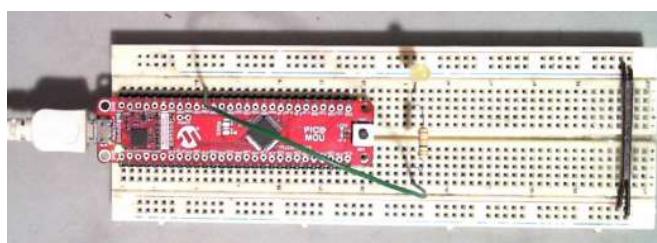
Ejemplo: Titileo de un LED en RB0 con opción de cambio de velocidad con RB4



25

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)
```

26

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

- Código en XC8 (con HFINTOSC a 48MHz)

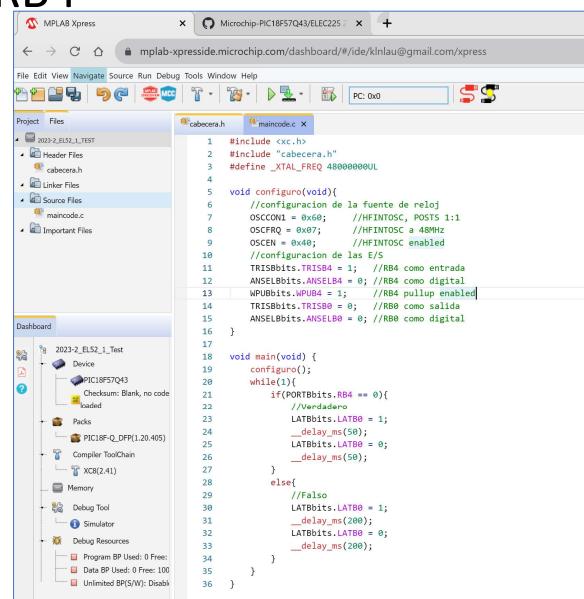
```

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 }
```

27

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

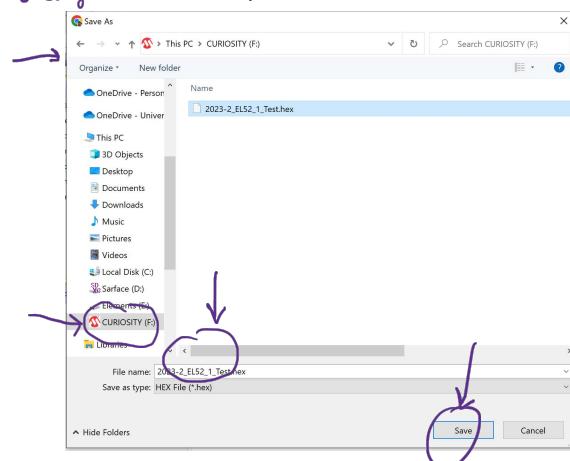
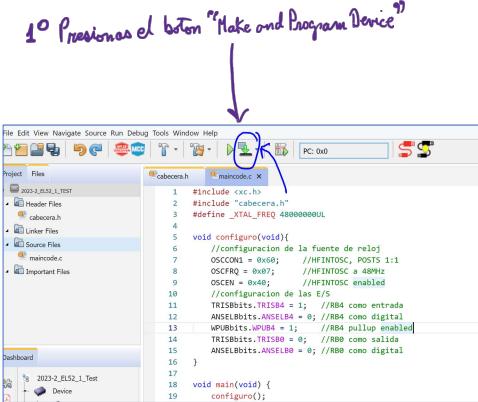
- Procedimiento para implementar este ejemplo utilizando el MPLAB Xpress
 - URL: <https://www.mplab-xpresside.microchip.com/>
 - Debes de registrarte para crear una cuenta dentro de dicha plataforma
 - Seguir las mismas indicaciones para crear un proyecto como en el MPLABX



28

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

- Procedimiento para implementar este ejemplo utilizando el MPLAB Xpress



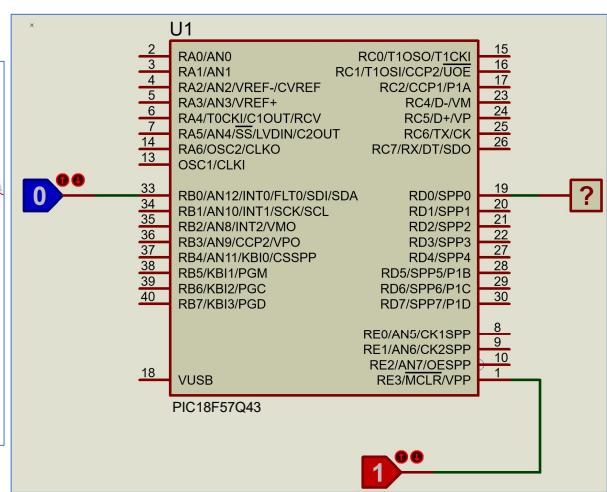
29

Ejemplo en XC8: Negador lógico de un bit

```

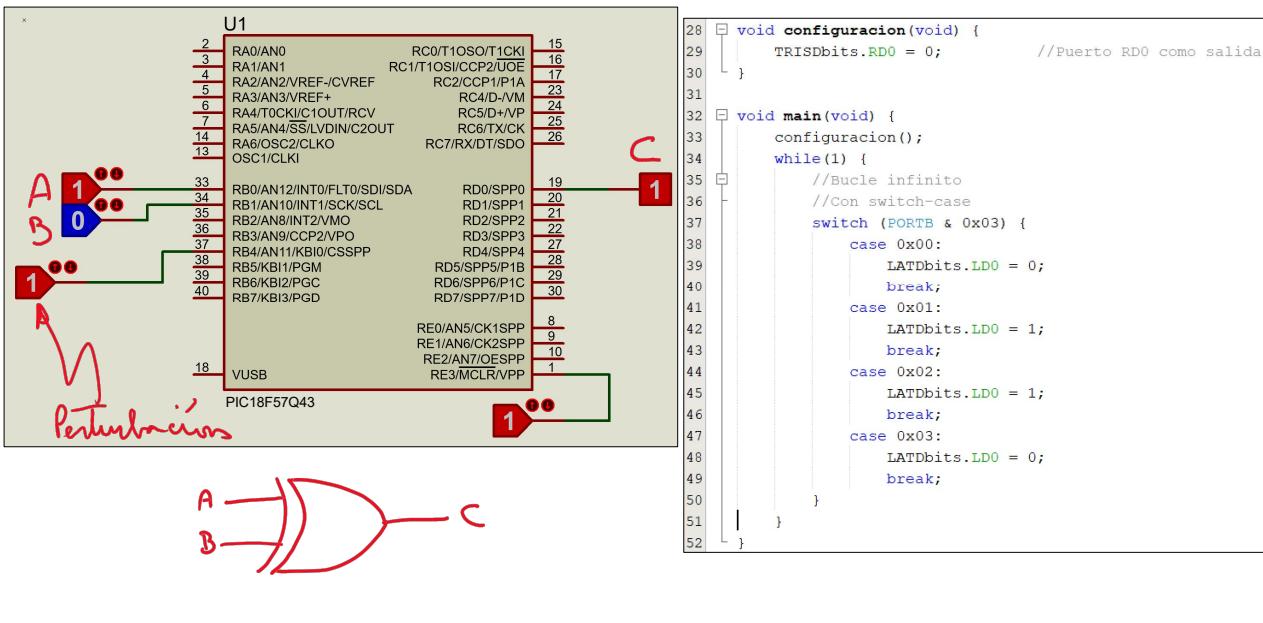
15 #include <xc.h>
16 #define _XTAL_FREQ 48000000UL
17
18 void init_conf(void) {
19     //Aca colocaremos las configuraciones iniciales de la aplicación
20     //TRISDbits.RD0 = 0;           // RD0 como salida
21     asm("bcf TRISD, 0");        // Escribiendo instrucciones en mpasm
22 }
23
24 void main(void) {
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 }

```



30

Ejemplo en XC8: Compuerta XOR



31

El LCD alfanuméricico 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



32

El LCD alfanumérico HD44780

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

Upper 4 Line	0000	0001	0010	0011	0100	0101	0110	0111	000	1001	1010	1011	1100	1101	1110	1111
CS Row (1)	0	1	P	P	-	タ	ミ	エ	0	ア	フ	ニ	エ	ス	0	1
xxxx0000	(2)	! 1A Q a q	・ ア フ ィ エ ク	q	q	q	q	q	q	q	q	q	q	q	q	q
xxxx0010	(3)	" 2B R b r	「 イ ツ エ ベ ベ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx0011	(4)	# 3C S c s	」 ウ テ ソ ソ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx0100	(5)	\$ 4D T d t	、 エ ベ ハ ム ベ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx0101	(6)	% 5E U e u	・ オ ナ ュ ゲ ュ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx0110	(7)	& 6F V f v	ヲ カ ニ ヨ パ サ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx0111	(8)	* 7G W g w	ア キ ヌ ラ ゴ パ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1000	(1)	(8H X h x	イ ク ネ リ バ ジ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1001	(2)) 9I Y i y	ウ ル ノ ル ビ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1010	(3)	* : J Z j z	エ コ ハ レ ジ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1011	(4)	+ : K L k l	オ サ ハ ロ *	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1100	(5)	, < L ¥ 1 1	ナ シ フ ワ フ ナ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1101	(6)	- = M] m }	ヌ ス ヘ ハ モ ハ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1110	(7)	. > N ^ n }	ミ セ ホ ハ ハ ハ	8	8	8	8	8	8	8	8	8	8	8	8	8
xxxx1111	(8)	/ ? O _ o {	リ リ リ リ リ リ リ	8	8	8	8	8	8	8	8	8	8	8	8	8

33

El LCD alfanumérico HD44780

- Tabla de caracteres ASCII de 7 bits
- Ref.
- <http://www.ascii-table.com>

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	NUL	(null)	32	20 040	4#32;	Space	64	40 100	4#64;	0	96	60 140	4#96;				
1	1 001	SOH	(start of heading)	33	21 041	4#33;	!	65	41 101	4#65;	A	97	61 141	4#97;	a			
2	2 002	STX	(start of text)	34	22 042	4#34;	"	66	42 102	4#66;	B	98	62 142	4#98;	b			
3	3 003	ETX	(end of text)	35	23 043	4#35;	#	67	43 103	4#67;	C	99	63 143	4#99;	c			
4	4 004	EOT	(end of transmission)	36	24 044	4#36;	\$	68	44 104	4#68;	D	100	64 144	4#100;	d			
5	5 005	ENQ	(enquiry)	37	25 045	4#37;	%	69	45 105	4#69;	E	101	65 145	4#101;	e			
6	6 006	ACK	(acknowledge)	38	26 046	4#38;	&	70	46 106	4#70;	F	102	66 146	4#102;	f			
7	7 007	BEL	(bell)	39	27 047	4#39;	:	71	47 107	4#71;	G	103	67 147	4#103;	g			
8	8 010	BS	(backspace)	40	28 050	4#40;	{	72	48 110	4#72;	H	104	68 150	4#104;	h			
9	9 011	TAB	(horizontal tab)	41	29 051	4#41;	}	73	49 111	4#73;	I	105	69 151	4#105;	i			
10	A 012	LF	(NL line feed, new line)	42	2A 052	4#42;	*	74	44 112	4#74;	J	106	6A 152	4#106;	j			
11	B 013	VT	(vertical tab)	43	2B 053	4#43;	+	75	4B 113	4#75;	K	107	6B 153	4#107;	k			
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			
13	D 015	CR	(carriage return)	45	2D 055	4#45;	-	77	4D 115	4#77;	M	109	6D 155	4#109;	m			
14	E 016	SO	(shift out)	46	2E 056	4#46;	.	78	4E 116	4#78;	N	110	6E 156	4#110;	n			
15	F 017	SI	(shift in)	47	2F 057	4#47;	/	79	4F 117	4#79;	O	111	6F 157	4#111;	o			
16	10 020	DLE	(data link escape)	48	30 060	4#48;	0	80	50 120	4#80;	P	112	70 160	4#112;	p			
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			
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			
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			
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			
21	15 025	NAK	(negative acknowledge)	53	35 065	4#53;	5	85	55 125	4#85;	U	117	75 165	4#117;	u			
22	16 026	SYN	(synchronous idle)	54	36 066	4#54;	6	86	56 126	4#86;	V	118	76 166	4#118;	v			
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			
24	18 030	CAN	(cancel)	56	38 076	4#56;	8	88	58 130	4#88;	X	120	78 170	4#120;	x			
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			
26	1A 032	SUB	(substitute)	58	3A 072	4#58;	:	90	5A 132	4#90;	Z	122	7A 172	4#122;	z			
27	1B 033	ESC	(escape)	59	3B 073	4#59;	;	91	5B 133	4#91;	[123	7B 173	4#123;	{			
28	1C 034	FS	(file separator)	60	3C 074	4#60;	<	92	5C 134	4#92;	\	124	7C 174	4#124;				
29	1D 035	GS	(group separator)	61	3D 075	4#61;	=	93	5D 135	4#93;]	125	7D 175	4#125;	}			
30	1E 036	RS	(record separator)	62	3E 076	4#62;	>	94	5E 136	4#94;	^	126	7E 176	4#126;	~			
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

34

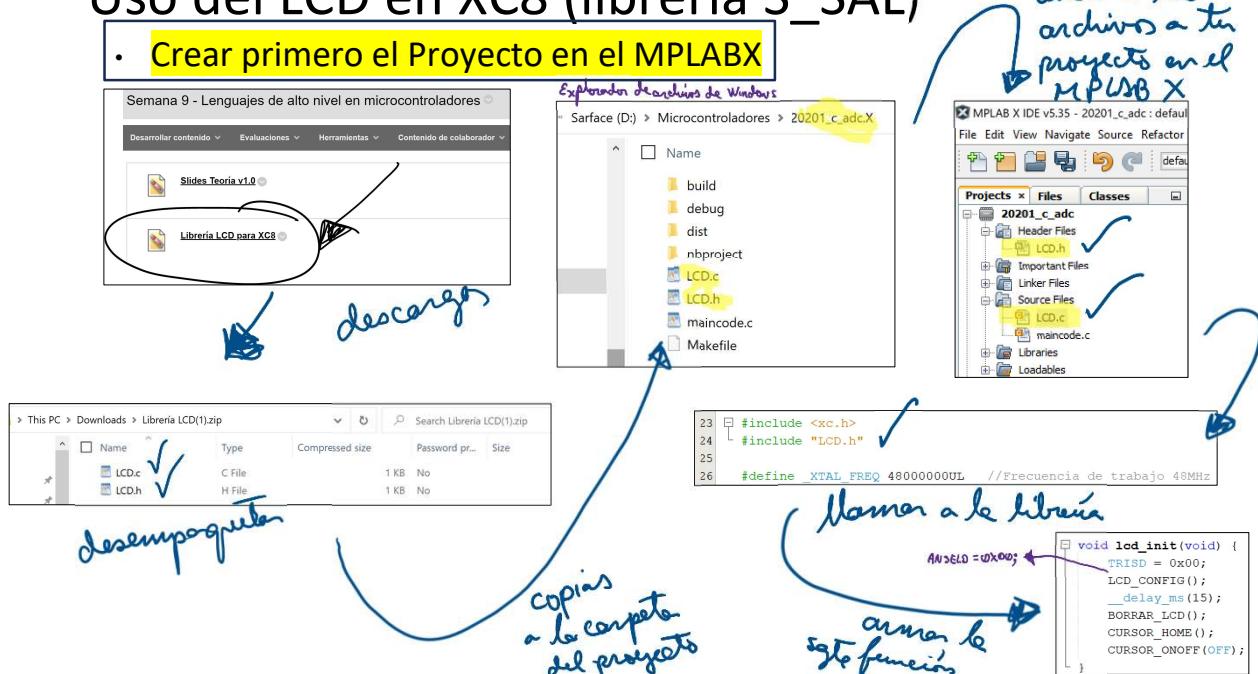
El LCD alfanumérico HD44780

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

35

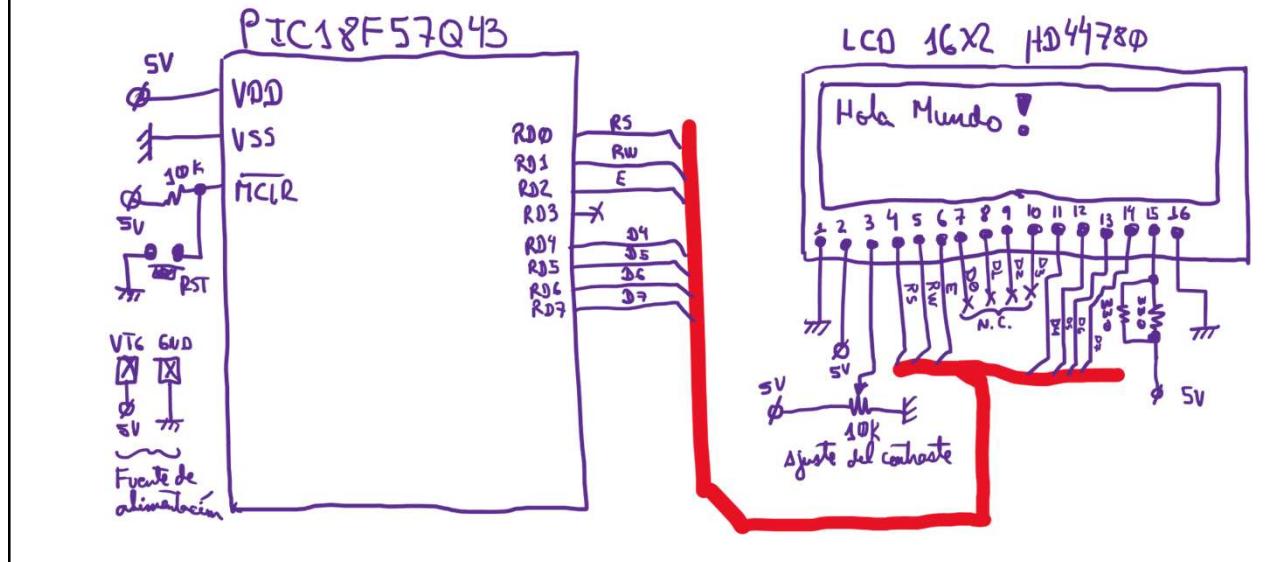
Uso del LCD en XC8 (librería S_SAL)

- Crear primero el Proyecto en el MPLABX



36

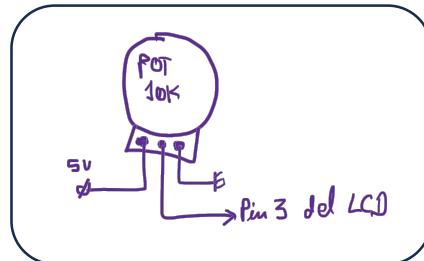
Circuito ejemplo de conexión del PIC18F57Q43 con el LCD



37

Recomendaciones en la interface del LCD con el microcontrolador

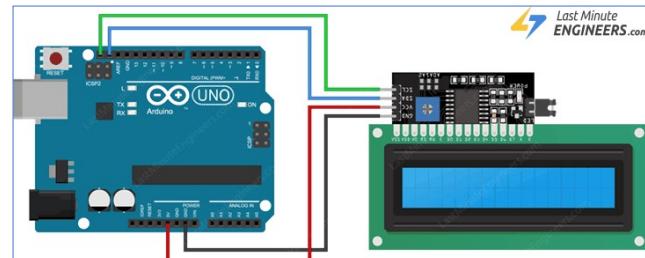
- Los pines deben de estar soldados correctamente!
- Seguir estrictamente el procedimiento para el llamado de las librerías.
- El potenciómetro debe de estar correctamente instalado



38

Interface del LCD 2x16 HD44780 en I2C

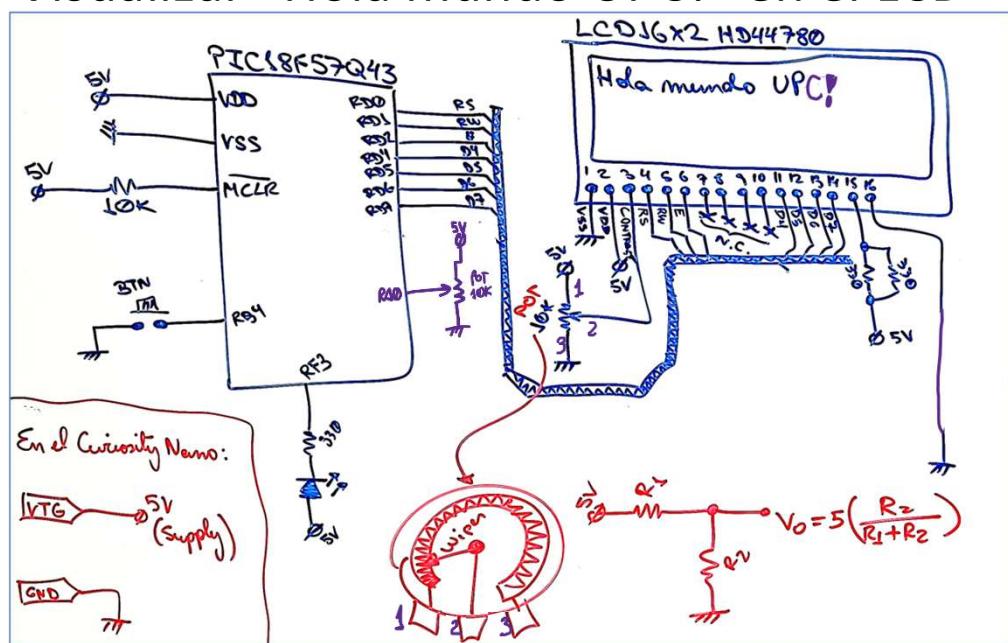
- Se le agrega un módulo adicional al LCD y permite la interface a dicho LCD usando I2C (comunicación serial), dicho módulo está basado en el circuito integrado PCF8574 (expansor de puertos vía I2C)
- Permite minimizar la cantidad de pines entre el LCD y el microcontrolador.
- Se requiere implementar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



39

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

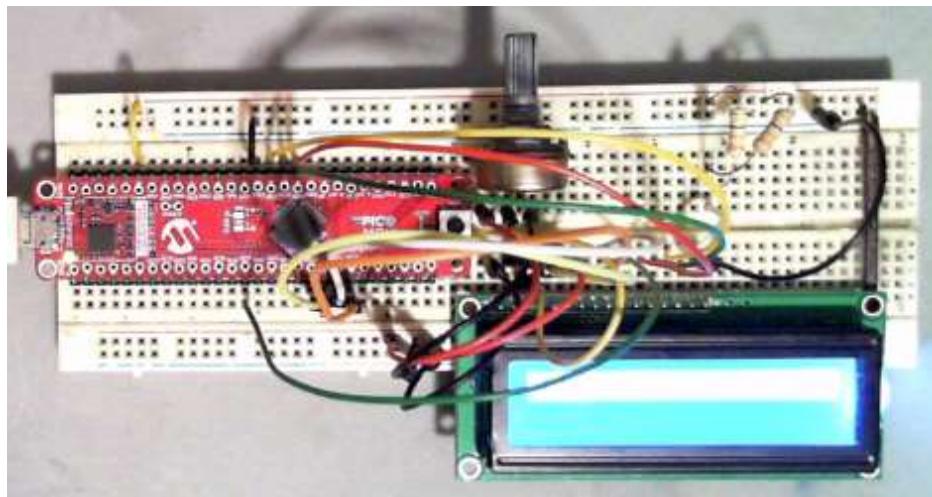
- Hardware



40

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

- Hardware



41

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 }
15
16 void lcd_init(void){
17     TRISD = 0x00;
18     ANSELD = 0x00;
19     LCD_CONFIG();
20     _delay_ms(15);
21     BORRAR_LCD();
22     CURSOR_HOME();
23     CURSOR_ONOFF(OFF);
24 }
25
26 void main(void){
27     configuro();
28     lcd_init();
29     POS_CURSOR(1,0);
30     ESCRIBE_MENSAJE("Hola mundo UPC!",15);
31     POS_CURSOR(2,0);
32     ESCRIBE_MENSAJE("Kalun Lau Gan",13);
33     while(1){
34         //aqui va el codigo de la aplicacion
35     }
36 }
```

42

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
14     //configuracion de las E/S
15     TRISFbits.TRISF3 = 0;   //RF3 como salida
16     ANSELFbits.ANSELF3 = 0; //RF3 como digital
17     TRISBbits.TRISB4 = 1;   //RB4 como entrada
18     ANSELBbits.ANSELB4 = 0; //RB4 como digital
19     WPUBbits.WPUB4 = 1;    //RB4 pullup activado
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("Kalon 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("Parpadeo rapido",15);
41             //parpadeo 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("Parpadeo lento ",15);
50             //parpadeo lento
51             LATFbits.LATF3 = 1;        //enciendo LED
52             _delay_ms(200);
53             LATFbits.LATF3 = 0;        //apago LED
54             _delay_ms(200);
55         }
56     }
57 }

```

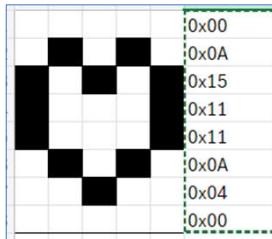
43

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/>

44

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[]={0x00,0xA,0x15,0x11,0x11,0xA,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

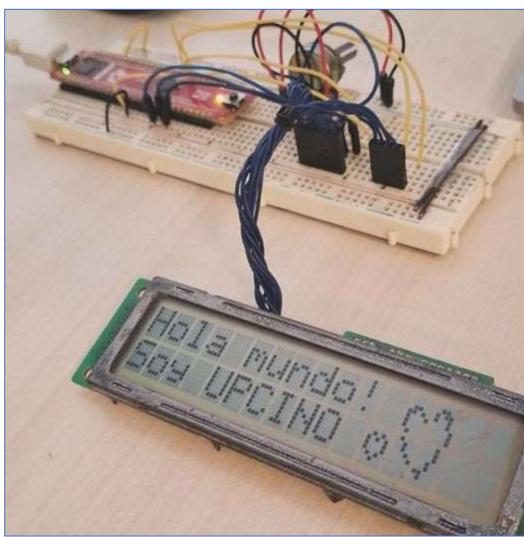
4. Para llamar al carácter personalizado, utilizar la función ENVIA_CHAR()

```
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);  
}
```

45

Caracteres personalizados

Visualización de una figura empleando múltiples caracteres personalizados



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

46

Conversor analógica a digital (A/D ó ADC)

Revisar Capítulo 40 del datasheet

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

47

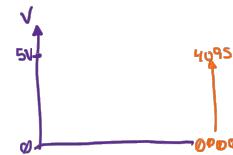
Conversor A/D

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

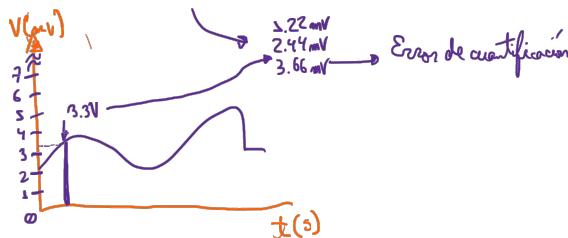
48

Sensibilidad del A/D del PIC18F57Q43

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



Ejemplo:



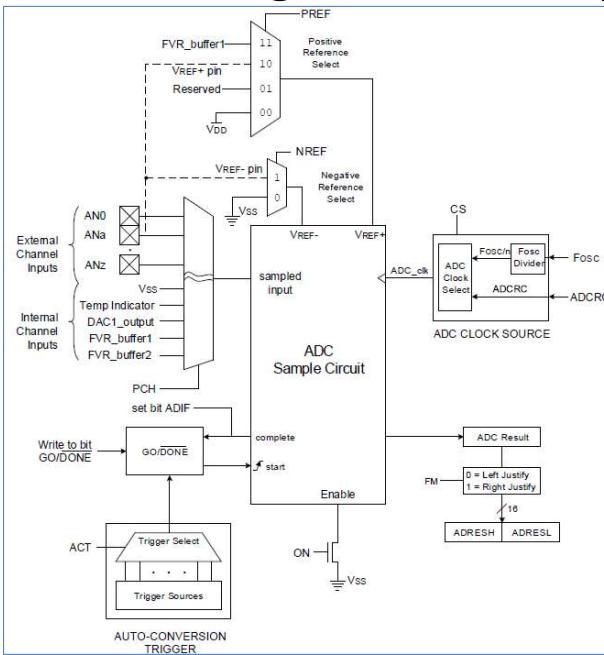
49

Alternativa de A/D de mayor resolución



50

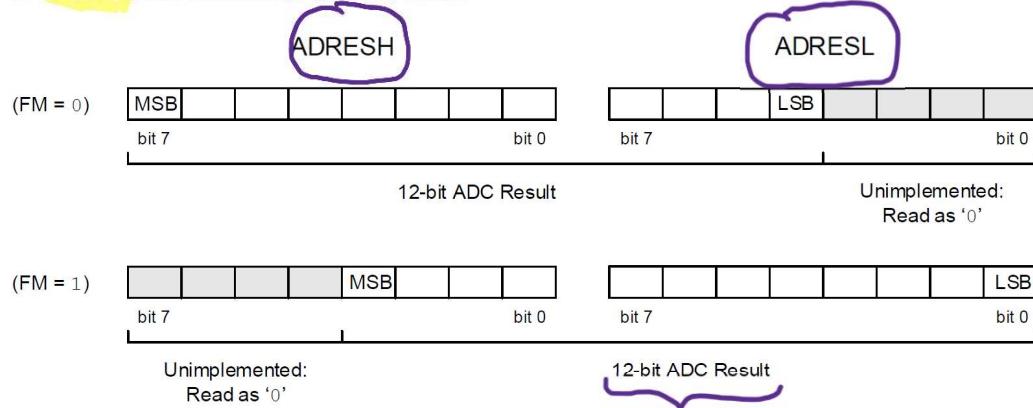
Conversor A/D: Diagrama de bloques



51

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
 $lectura = (ADRESH \ll 8) + ADRESL$

52

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]				
0x03DB	ADUTH	7:0				LTH[15:8]				
0x03DD	ADERR	7:0				UTH[7:0]				
0x03DF	ADSTPT	7:0				UTH[15:8]				
0x03E1	ADFLTR	7:0				ERR[7:0]				
0x03E1	ADFLTR	7:0				ERR[15:8]				
0x03E3	ADACC	7:0				STPT[7:0]				
0x03E3	ADACC	15:8				STPT[15:8]				
0x03E6	ADCN	7:0				FLTR[7:0]				
0x03E7	ADRPT	7:0				FLTR[15:8]				
0x03E8	ADPREV	7:0				ACC[7:0]				
0x03EA	ADRES	7:0				ACC[15:8]				
0x03EC	ADPCH	7:0				23:16				ACC[17:16]
0x03ED	Reserved									
0x03EE	ADACQ	7:0					CNT[7:0]			
0x03EE	ADACQ	15:8					RPT[7:0]			
0x03F0	ADCAP	7:0					PREV[7:0]			
0x03F1	ADPRE	7:0					PREV[15:8]			
0x03F3	ADCON0	7:0	ON	CONT			RES[7:0]			
0x03F4	ADCON1	7:0	PPOL	IPEN	GPOL		RES[15:8]			
0x03F5	ADCON2	7:0	PSIS		CRS[2:0]		RES[23:16]			
0x03F6	ADCON3	7:0			CALC[2:0]		RES[23:16]			
0x03F7	ADSTAT	7:0	AOV	UTHR	LTHR	MATH	SOI			
0x03F8	ADREF	7:0				NREF	MD[2:0]			
0x03F9	ADACT	7:0					TMD[2:0]			
0x03FA	ADCLK	7:0					STAT[2:0]			
0x03FA	ADCLK						PREF[1:0]			
0x03FA	ADCLK						CS[5:0]			

53

Conversor A/D: Registros

Name:	ADCON0							
Address:	0x3F3							
ADC Control Register 0								
Bit	7	6	5	4	3	2	1	0
Access	ON	CONT		CS		FM		GO
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 triggered 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 Fosc, 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							

54

Conversor A/D: Registros

Name: ADCON1
Address: 0x3F4

ADC Control Register 1

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

Bit 7 – PPOL Precharge Polarity

Action During 1st Precharge Stage

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

Bit 6 – IPEN A/D Inverted Precharge Enable

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

Bit 5 – GPOL Guard Ring Polarity Selection

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

Bit 0 – DSEN Double-Sample Enable

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

55

Conversor A/D: Registros

Name: ADCON2
Address: 0x3F5

ADC Control Register 2

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

Bit 7 – PSIS ADC Previous Sample Input Select

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

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

Value	Condition	Description
1 to e	MD = 'b100	Low-pass filter time constant is 2 ^{CRS} , filter gain is 1:1 ⁽²⁾
1 to e	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:

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

56

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																																																			
<table border="1"> <thead> <tr> <th colspan="3">ADERR</th> <th>Application</th> </tr> <tr> <th>CALC</th> <th>DSEN = 0 Single-Sample Mode</th> <th>DSEN = 1 CVD Double-Sample Mode⁽¹⁾</th> <th></th> </tr> </thead> <tbody> <tr> <td>111</td> <td>Reserved</td> <td>Reserved</td> <td>Reserved</td> </tr> <tr> <td>110</td> <td>Reserved</td> <td>Reserved</td> <td>Reserved</td> </tr> <tr> <td>101</td> <td>ADFLTR-ADSTPT</td> <td>ADFLTR-ADSTPT</td> <td>Average/filtered value vs. setpoint</td> </tr> <tr> <td>100</td> <td>ADPREV-ADFLTR</td> <td>ADPREV-ADFLTR</td> <td>First derivative of filtered value⁽³⁾ (negative)</td> </tr> <tr> <td>011</td> <td>Reserved</td> <td>Reserved</td> <td>Reserved</td> </tr> <tr> <td>010</td> <td>ADRES-ADFLTR</td> <td>(ADRES-ADPREV)-ADFLTR</td> <td>Actual result vs. averaged/filtered value</td> </tr> <tr> <td>001</td> <td>ADRES-ADSTPT</td> <td>(ADRES-ADPREV)-ADSTPT</td> <td>Actual result vs. setpoint</td> </tr> <tr> <td>000</td> <td>ADRES-ADPREV</td> <td>ADRES-ADPREV</td> <td>First derivative of single measurement⁽²⁾</td> </tr> <tr> <td></td> <td></td> <td></td> <td>Actual CVD result⁽²⁾</td> </tr> </tbody> </table>								ADERR			Application	CALC	DSEN = 0 Single-Sample Mode	DSEN = 1 CVD Double-Sample Mode ⁽¹⁾		111	Reserved	Reserved	Reserved	110	Reserved	Reserved	Reserved	101	ADFLTR-ADSTPT	ADFLTR-ADSTPT	Average/filtered value vs. setpoint	100	ADPREV-ADFLTR	ADPREV-ADFLTR	First derivative of filtered value ⁽³⁾ (negative)	011	Reserved	Reserved	Reserved	010	ADRES-ADFLTR	(ADRES-ADPREV)-ADFLTR	Actual result vs. averaged/filtered value	001	ADRES-ADSTPT	(ADRES-ADPREV)-ADSTPT	Actual result vs. setpoint	000	ADRES-ADPREV	ADRES-ADPREV	First derivative of single measurement ⁽²⁾				Actual CVD result ⁽²⁾
ADERR			Application																																																
CALC	DSEN = 0 Single-Sample Mode	DSEN = 1 CVD Double-Sample Mode ⁽¹⁾																																																	
111	Reserved	Reserved	Reserved																																																
110	Reserved	Reserved	Reserved																																																
101	ADFLTR-ADSTPT	ADFLTR-ADSTPT	Average/filtered value vs. setpoint																																																
100	ADPREV-ADFLTR	ADPREV-ADFLTR	First derivative of filtered value ⁽³⁾ (negative)																																																
011	Reserved	Reserved	Reserved																																																
010	ADRES-ADFLTR	(ADRES-ADPREV)-ADFLTR	Actual result vs. averaged/filtered value																																																
001	ADRES-ADSTPT	(ADRES-ADPREV)-ADSTPT	Actual result vs. setpoint																																																
000	ADRES-ADPREV	ADRES-ADPREV	First derivative of single measurement ⁽²⁾																																																
			Actual CVD result ⁽²⁾																																																
Notes:																																																			
<ol style="list-style-type: none"> When DSEN = 1 and PSIS = 0, ADERR is computed only after every second sample. When PSIS = 0. When PSIS = 1. 																																																			
Bit 3 – SOI ADC Stop-on-Interrupt																																																			
<table border="1"> <thead> <tr> <th>Value</th> <th>Condition</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>CONT = 0</td> <td>This bit is not used</td> </tr> <tr> <td>0</td> <td>CONT = 1</td> <td>GO is cleared when the threshold conditions are met, otherwise the conversion is retriggered</td> </tr> </tbody> </table>									Value	Condition	Description	1	CONT = 0	This bit is not used	0	CONT = 1	GO is cleared when the threshold conditions are met, otherwise the conversion is retriggered																																		
Value	Condition	Description																																																	
1	CONT = 0	This bit is not used																																																	
0	CONT = 1	GO is cleared when the threshold conditions are met, otherwise the conversion is retriggered																																																	
Bit 2:0 – TMD[2:0] Threshold Interrupt Mode Select																																																			
<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>111</td> <td>Interrupt regardless of threshold test results</td> </tr> <tr> <td>110</td> <td>Interrupt if ADERR > ADUTH</td> </tr> <tr> <td>101</td> <td>Interrupt if ADERR < ADUTH</td> </tr> <tr> <td>100</td> <td>Interrupt if ADERR < ADLTH or ADERR > ADUTH</td> </tr> <tr> <td>011</td> <td>Interrupt if ADERR > ADLTH and ADERR < ADUTH</td> </tr> <tr> <td>010</td> <td>Interrupt if ADERR ≥ ADLTH</td> </tr> <tr> <td>001</td> <td>Interrupt if ADERR < ADLTH</td> </tr> <tr> <td>000</td> <td>Never interrupt</td> </tr> </tbody> </table>									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																									
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																																																		

57

Conversor A/D: Registros

Name:	ADRES							
Address:	0x3EA							
ADC Result Register								
Bit	15	14	13	12	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
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bits 15:0 – RES[15:0] ADC Sample Result								
Notes: The individual bytes in this multibyte register can be accessed with the following register names:								
<ul style="list-style-type: none"> ADRESH: Accesses the high byte ADRES[15:8] ADRESL: Accesses the low byte ADRES[7:0] 								

58

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	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"> <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⁽⁴⁾</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>R2E/ANE2⁽⁴⁾</td></tr> <tr><td>100001</td><td>R1E/ANE1⁽⁴⁾</td></tr> <tr><td>100000</td><td>R0E/ANEO⁽⁴⁾</td></tr> <tr><td>011111</td><td>R0D/AND7⁽⁴⁾</td></tr> <tr><td>011110</td><td>R0B/AND6⁽⁴⁾</td></tr> <tr><td>011101</td><td>R0S/AND5⁽⁴⁾</td></tr> <tr><td>011100</td><td>R04/AND4⁽⁴⁾</td></tr> <tr><td>011011</td><td>R03/AND3⁽⁴⁾</td></tr> <tr><td>011010</td><td>R02/AND2⁽⁴⁾</td></tr> <tr><td>011001</td><td>R01/AND1⁽⁴⁾</td></tr> <tr><td>011000</td><td>R0D/AND0⁽⁴⁾</td></tr> <tr><td>010111</td><td>R07/ANC7</td></tr> <tr><td>010110</td><td>R06/ANC6</td></tr> <tr><td>010101</td><td>R05/ANC5</td></tr> <tr><td>010100</td><td>R04/ANC4</td></tr> <tr><td>010011</td><td>R03/ANC3</td></tr> <tr><td>010010</td><td>R02/ANC2</td></tr> <tr><td>010001</td><td>R01/ANC1</td></tr> <tr><td>010000</td><td>R00/ANCO</td></tr> <tr><td>001111</td><td>R07/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 ⁽⁴⁾	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	R2E/ANE2 ⁽⁴⁾	100001	R1E/ANE1 ⁽⁴⁾	100000	R0E/ANEO ⁽⁴⁾	011111	R0D/AND7 ⁽⁴⁾	011110	R0B/AND6 ⁽⁴⁾	011101	R0S/AND5 ⁽⁴⁾	011100	R04/AND4 ⁽⁴⁾	011011	R03/AND3 ⁽⁴⁾	011010	R02/AND2 ⁽⁴⁾	011001	R01/AND1 ⁽⁴⁾	011000	R0D/AND0 ⁽⁴⁾	010111	R07/ANC7	010110	R06/ANC6	010101	R05/ANC5	010100	R04/ANC4	010011	R03/ANC3	010010	R02/ANC2	010001	R01/ANC1	010000	R00/ANCO	001111	R07/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 ⁽⁴⁾																																																																								
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	R2E/ANE2 ⁽⁴⁾																																																																								
100001	R1E/ANE1 ⁽⁴⁾																																																																								
100000	R0E/ANEO ⁽⁴⁾																																																																								
011111	R0D/AND7 ⁽⁴⁾																																																																								
011110	R0B/AND6 ⁽⁴⁾																																																																								
011101	R0S/AND5 ⁽⁴⁾																																																																								
011100	R04/AND4 ⁽⁴⁾																																																																								
011011	R03/AND3 ⁽⁴⁾																																																																								
011010	R02/AND2 ⁽⁴⁾																																																																								
011001	R01/AND1 ⁽⁴⁾																																																																								
011000	R0D/AND0 ⁽⁴⁾																																																																								
010111	R07/ANC7																																																																								
010110	R06/ANC6																																																																								
010101	R05/ANC5																																																																								
010100	R04/ANC4																																																																								
010011	R03/ANC3																																																																								
010010	R02/ANC2																																																																								
010001	R01/ANC1																																																																								
010000	R00/ANCO																																																																								
001111	R07/ANB7																																																																								
<table border="1"> <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:																																																																									
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.																																																																									

59

Conversor A/D: Registros

Name:	ADREF										
Address:	0x3F8										
ADC Reference Selection Register											
Bit	7	6	5	4	3	PREF[1:0]	2	1	0		
Access	R/W	R/W	R/W	NREF	R/W	PREF[1:0]	R/W	R/W	R/W		
Reset	0	0	0		0		0	0	0		
Bit 4 – NREF ADC Negative Voltage Reference Selection											
<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>1</td><td>V_{REF^-} is connected to external V_{REF^-}</td></tr> <tr><td>0</td><td>V_{REF^-} is connected to AV_{SS}</td></tr> </tbody> </table>		Value	Description	1	V_{REF^-} is connected to external V_{REF^-}	0	V_{REF^-} is connected to AV_{SS}				
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											
<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>11</td><td>V_{REF^+} is connected to internal Fixed Voltage Reference (FVR) module</td></tr> <tr><td>10</td><td>V_{REF^+} is connected to external V_{REF^+}</td></tr> <tr><td>01</td><td>Reserved</td></tr> <tr><td>00</td><td>V_{REF^+} is connected to V_{DD}</td></tr> </tbody> </table>		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}
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}										

60

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 ADPCCH
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

61

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

Recorriendo:

$$\text{Circuito divisor de tensión: } V_O = V_I \left(\frac{R_2}{R_1+R_2} \right)$$

Con $R_1 = 1k$ y $R_2 = 1k$, se obtiene $V_O = 2.5V$

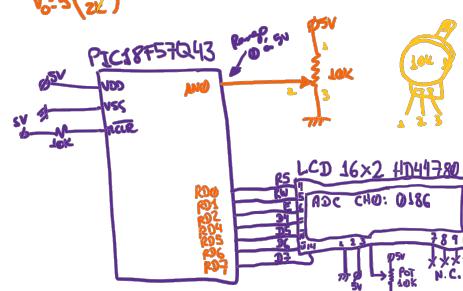
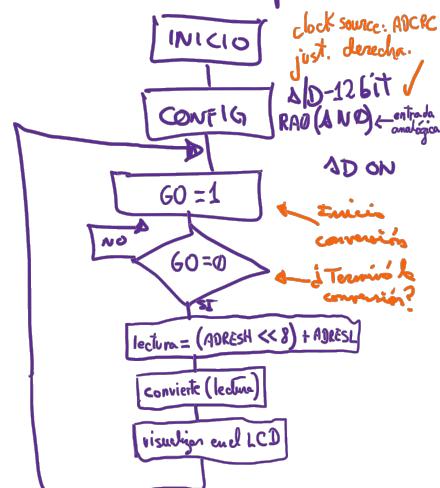


Diagrama de flujo:



62

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

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

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

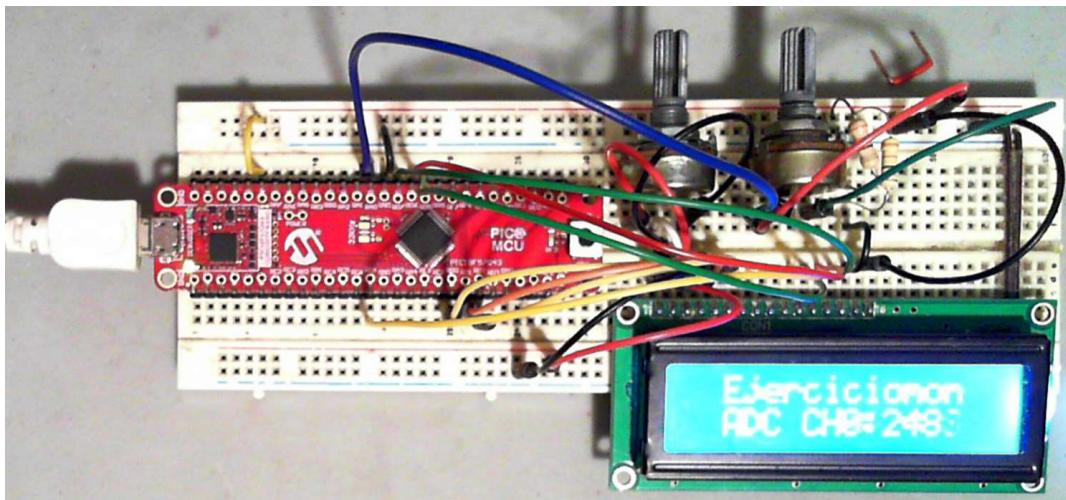
63

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 = 0x00; //HFINTOSC, POSTS 1:1
12     OSCFRO = 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.FN = 1; //right justify
24     ADCON0bits.CS = 1; //ADCRC Clock
25     ADCF = 0x00; //RA0 es 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
31 void lcd_init(void){
32     TRISD = 0x00; //RD salidas
33     ANSELD = 0x00; //RD digitales
34     LCD_CONFIG();
35     delay_ms(19);
36     BORRAR_LCD();
37     CURSOR_HOME();
38     CURSOR_ONOFF(OFF);
39 }
40
41 void convierte(unsigned int valor){
42     millar = valor / 1000;
43     centena = (valor % 1000) / 100;
44     decena = (valor % 100) / 10;
45     unidad = valor % 10;
46 }
47
48 void main(void) {
49     configuro();
50     lcd_init();
51     POS_CURSOR(1,2);
52     ESCRIBE_MENSAJE("Ejercicio01", 12);
53     while(1){
54         ADCON0bits.GO = 1; //Start conversion
55         while(ADCON0bits.GO); //Wait for conversion done
56         resultado = (ADRESH << 8) + ADRESL;
57         convierte(resultado);
58         POS_CURSOR(2,2);
59         ESCRIBE_MENSAJE("ADC CH0:", 8);
60         ENVIA_CHAR(millar+0x30);
61         ENVIA_CHAR(centena+0x30);
62         ENVIA_CHAR(decena+0x30);
63         ENVIA_CHAR(unidad+0x30);
64     }
65 }
```

64

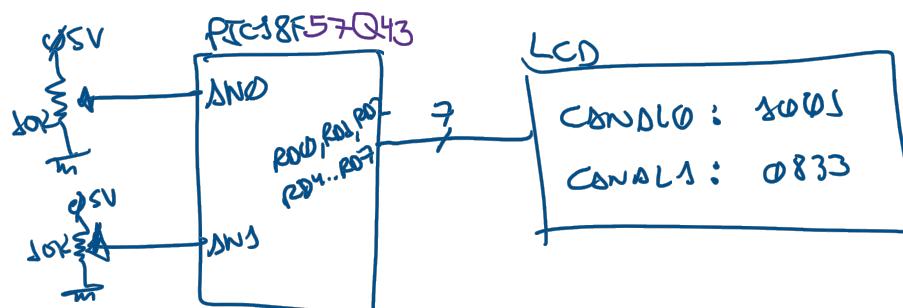
Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado (12 bits) emitirlo al LCD en formato decimal



65

Asignación:

- Leer dos canales analógicos y mostrarlos en el LCD



66

El módulo DAC del PIC18F57Q43

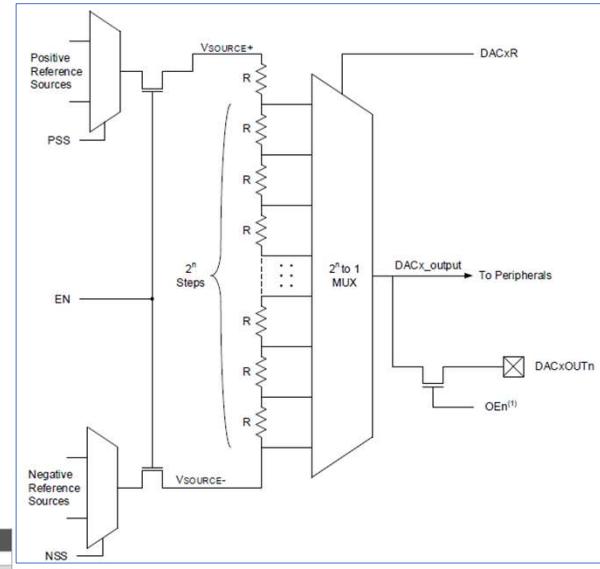
- Referencia: capítulo 41 del datasheet
- Resolución: 8bits
- Tipo arreglo R-2R
- Se puede usar el FVR como voltaje de referencia positivo (bits PSS)
- Fórmula:

Equation 41-1. DAC Output Equation

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

- Registros:

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



67

El módulo DAC del PIC18F57Q43

Digital-to-Analog Converter Control Register												
Bit	7	6	5	4	3	2	1	0				
Access	R/W			R/W	R/W	R/W	R/W	R/W				
Reset	0			0	0	0	0	0				
Bit 7 – EN DAC Enable												
Value	Description											
1	DAC is enabled											
0	DAC is disabled											
Bits 5:4 – OE[1:0] DAC Output Enable												
OE	DAC Outputs											
11	DACxOUT is disabled											
10	DACxOUT is enabled on pin RA2 only											
01	DACxOUT is enabled on pin RB7 only											
00	DACxOUT is disabled											
Bits 3:2 – PSS[1:0] DAC Positive Reference Selection												
PSS	DAC Positive Reference											
11	Reserved, do not use											
10	FVR Buffer 2											
01	V_{REF+}											
00	V_{DD}											
Bit 0 – NSS DAC Negative Reference Selection												
NSS	DAC Negative Reference											
1	V_{REF-}											
0	V_{SS}											

68

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

69

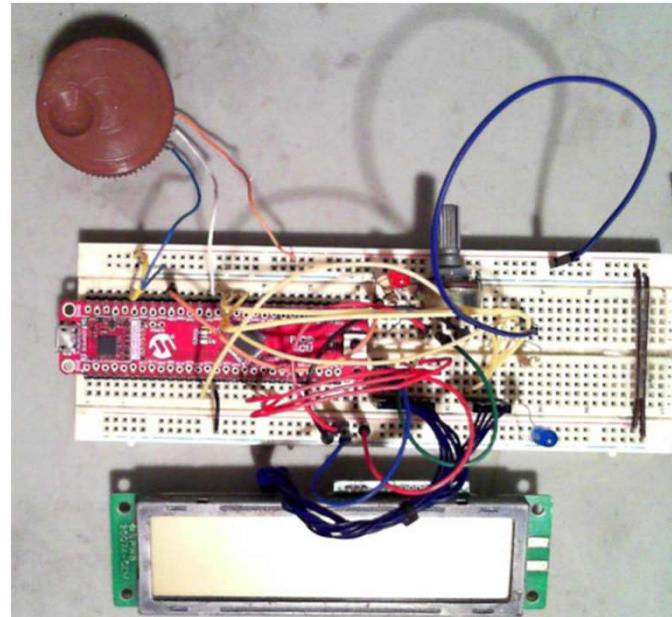
Evidencias de ejemplos 2024-1

70

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



71

Implementación 2024-1

Prueba inicial:

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

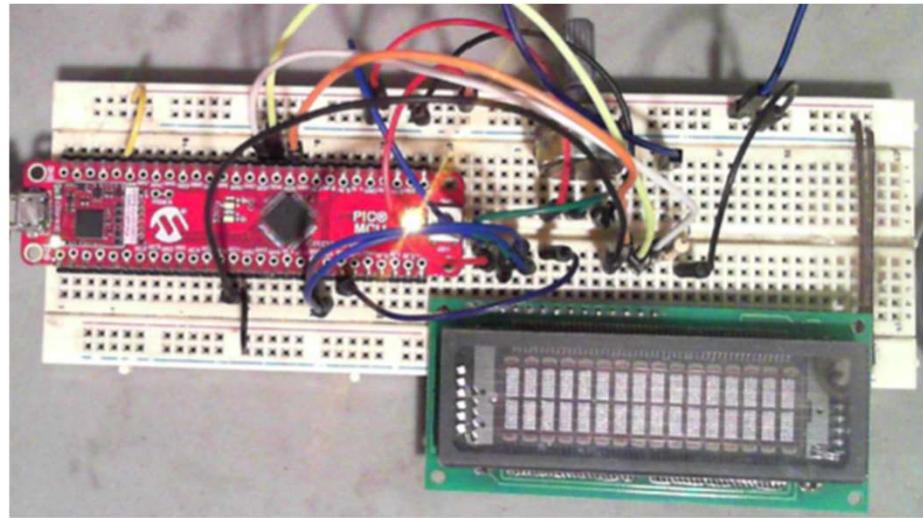
```

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 }
```

72

Implementación 2024-1

Evidencia de titileo de LED en RF3



73

Implementación 2024-1

Modificación del ejemplo inicial:

- Se agregó el pulsador integrado en RB4 para controlar el titileo del LED en RF3.
- Solo titilará el LED si se mantiene 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 NDIV1:1
8      OSCFRQ = 0x07;           //HFINTOSC 48MHz
9      OSCEN = 0x40;            //HFINTOSC enabled
10     TRISFbits.TRISF3 = 0;    //RF3 output
11     ANSELFbits.ANSELF3 = 0;  //RF3 digital
12     TRISBbits.TRISB4 = 1;    //RB4 input
13     ANSELBbits.ANSELB4 = 0;  //RB4 digital
14     WPUBbits.WPUB4 = 1;     //RB4 weak pullup enabled
15 }
16
17 void main(void) {
18     configuro();             //calling configuro function
19     while(1){                //do it forever
20         if(PORTBbits.RB4 == 0){
21             LATFbits.LATF3 = 0; //LED on
22             __delay_ms(100);   //delay of 100ms
23             LATFbits.LATF3 = 1; //LED off
24             __delay_ms(100);   //delay of 100ms
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 }
```

74

```

1 #include <xc.h>
2 #include "cabecera.h"
3 #define _XTAL_FREQ 48000000UL
4
5 void configuro(void){
6     OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
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
16 void main(void) {
17     configuro();
18     while(1){
19         if(PORTBbits.RB4 == 0){
20             LATFbits.LATF3 = 0; //LED on
21             _delay_ms(100);
22             LATFbits.LATF3 = 1; //LED off
23             _delay_ms(100);
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.

75

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; //detección falling edge
19    INT0PPS = 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 }

```

```

37 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void) {
38     PIRbits.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 rutear dicha interrupción hacia RB4 empleando el PPS (línea 19).
- Se estableció una FEM de dos estados en donde la acción de cambio es el evento de INT0, la variable indicador es el que determinará si parpadeará el LED o se mantiene apagado.

76

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
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     INTOPPS = 0x0C; //ruteando INT0 a RB4
18     INTCON0bits.INT0EDG = 0; //INT0 falling edge
19     PIE1bits.INT0IE = 1; //habilita INT0
20     INTCON0bits.GIE = 1; //int global enabled
21 }

```

```

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             }
42         }
43     }
44 }

```

```

46 void __interrupt(irq(IRQ_INTERRUPTO)) 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 titileo del LED.

77

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
5
6  #define apagado 0
7  #define parpadeando 1
8
9  unsigned char estado_LED = apagado;
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 INT0 a RB4
21     INTCON0bits.INT0EDG = 0; //INT0 falling edge
22     PIE1bits.INT0IE = 1; //habilita INT0
23     INTCON0bits.GIE = 1; //int global enabled
24 }

```

```

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
41
42 void __interrupt(irq(IRQ_INTERRUPTO)) INT0_ISR(void){
43     PIR1bits.INT0IF = 0; //abajamos bandera INT0IF
44     if(estado_LED == apagado){
45         estado_LED = parpadeando;
46     }
47     else{
48         estado_LED = apagado;
49     }
50 }
51
52 void __interrupt(irq(default)) default_ISR(void){
53 }

```

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

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

78

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    TRISFbits.TRISF3 = 0;
13    ANSELFBits.ANSELF3 = 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    PIEbits.INT0IE = 1;
23    PIE6bits.INT1IE = 1;
24    INTCON0bits.GIE = 1;
25    INTCON0bits.INT0EDG = 0;
26    INTCON0bits.INT1EDG = 0;
27    INT0PPS = 0x0C;
28    INT1PPS = 0x0B;
29 }
30
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 }
53
54 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
55     PIR6bits.INT1IF = 0; //bajamos bandera
56     if (indicador == 0){
57         indicador = 1;
58     }
59     else{
60         indicador = 0;
61     }
62 }
63
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)

79

Implementación 2024-1

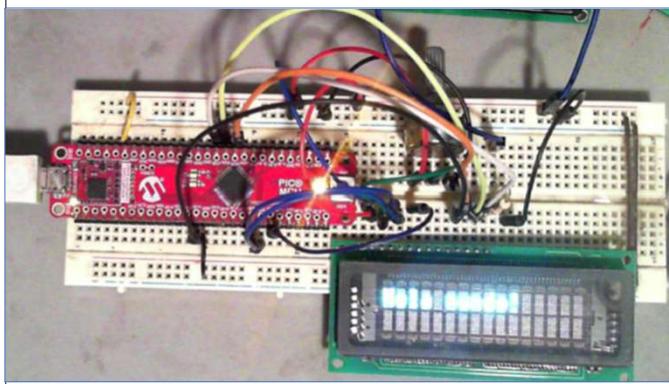
```

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    ANSELFBits.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 }

```

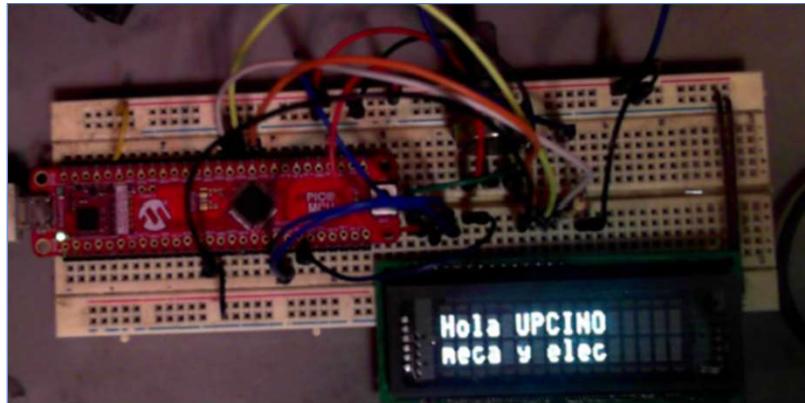
Segunda prueba: Visualizar "Hola mundo!" en el LCD:

- 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.



80

Implementación 2024-1



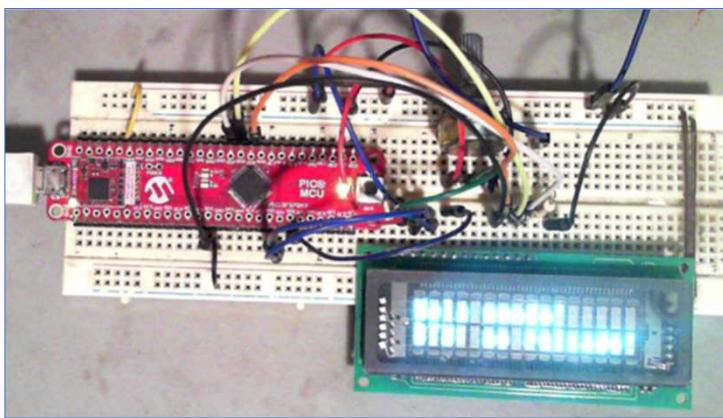
Ejemplo de visualización de mensajes en ambas líneas del LCD:

- 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

81

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     ANSELPbits.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 }

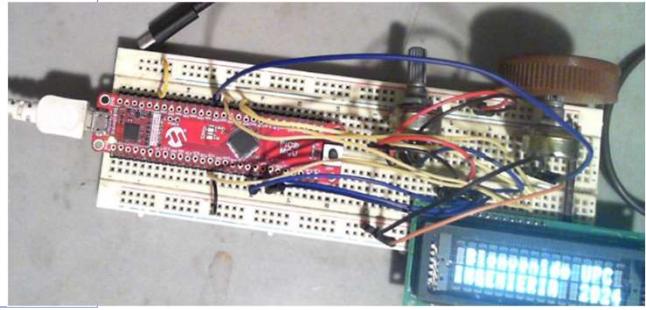
17
18 void main(void) {
19     configuro();
20     LCD_INIT();
21     POS_CURSOR(1,0);
22     ESCRIBE_MENSAJE2("Hola mundo!");
23     while(1){
24         if(PORTBbits.RB4 == 0){
25             POS_CURSOR(2,0);
26             ESCRIBE_MENSAJE2("BTN: pulsado    ");
27         }
28         else{
29             POS_CURSOR(2,0);
30             ESCRIBE_MENSAJE2("BTN: no pulsado!");
31         }
32     }
33 }
```

82

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     //configuracion del A/D
17     TRISAbits.TRISA0 = 1; //RA0 entrada
18     ANSELAbits.ANSELA0 = 1; //RA0 analogica
19     ADCON0 = 0x94; //ADC conv manual, ADCRC, just der, ADC on
20 }
21
22 void LCD_inicia(void){
23     TRISD = 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 }
33
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 }
46 }
```



83

**EL54 2024_1
Sem10**

Logo de UPC empleando seis caracteres personalizados

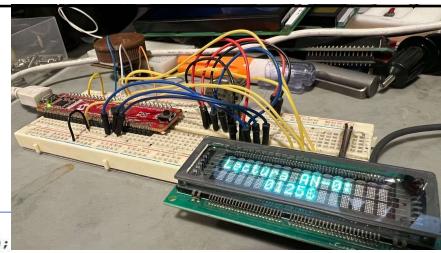
```

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=1:1
21     OSCFRQ = 0x06; //HFINTOSC a 32MHz
22     OSCEN = 0x40; //HFINTOSC enabled
23     TRISAbits.TRISA0 = 1; //RA0 entrada
24     ANSELAbits.ANSELA0 = 1; //RA0 analogica
25 }
26
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 }
44
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     }
60 }
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	0x01	0x11	0x10													
2	0x03	0x04	0x18													
3	0x06	0x06	0x0C													
4	0x04	0x0E	0x04													
5	0x0C	0x1C	0x04													
6	0x18	0x08	0x03													
7	0x18	0x06	0x03													
8	0x18	0x03	0x03													
9	0x1C	0x03	0x17													
10	0x0C	0x03	0x06													
11	0x07	0x04	0x1C													
12	0x07	0x1F	0x1C													
13	0x01	0x1F	0x10													
14	0x00	0xE	0x00													
15																
16																
17																

84

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;           //HFINTOSC a 32MHz
14     OSCFRQ = 0x06;           //HFINTOSC a 32MHz
15     OSCEN = 0x40;
16     //configuración del ADC
17     TRISAbits.TRISA0 = 1;    //RA0 entrada
18     ANSELAbits.ANSELAO = 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 u
25     while(ADCON0bits.GO == 1);
26     //el resultado está en ADRESH:ADRESL
27 }

```

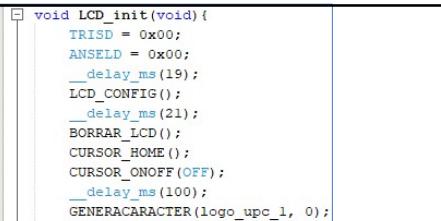
Primeras pruebas de uso del ADC para obtener una muestra de una señal analógica ingresada por RA0. La visualización en el LCD corresponde al resultado del ADC en formato de escalones en 12 bits

```

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 }

```

85



```

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[]={0x01,0x03,0x06,0x04,0x00,0x0C,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,0x04,0x06,0x03,0x03};
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     OSCFRQ = 0x06;           //HFINTOSC a 32MHz
22     OSCEN = 0x40;           //HFINTOSC enabled
23     //configuración del A/D
24     TRISAbits.TRISA0 = 1;    //RA0 entrada
25     ANSELAbits.ANSELAO = 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     TRISD = 0x00;
38     ANSEL0 = 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 sesis caracteres personalizados y el uso del ADC para tomar una muestra del canal RA0/ANA0, finalmente la visualización en steps 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 RAO:",12);
71         resultado = lectura_ADC_0();
72         POS_CURSOR(2,5);
73         ENVIA_CHAR((resultado / 10000) + 0x30); //vis diez millar
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 }

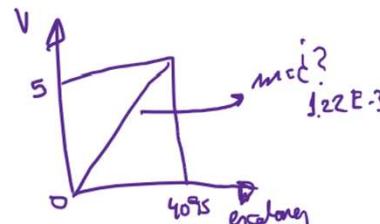
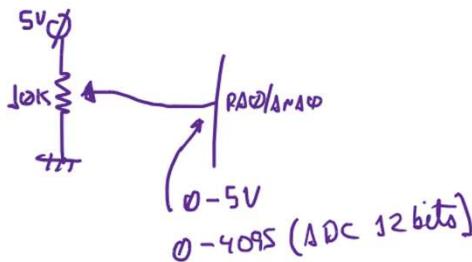
```

86

EL54 2024_1
Sem10

Implementación ADC 2024-1

Deseo visualizar el voltaje que recibe RA0/ANA0



87

EL54 2024_1 Sem10

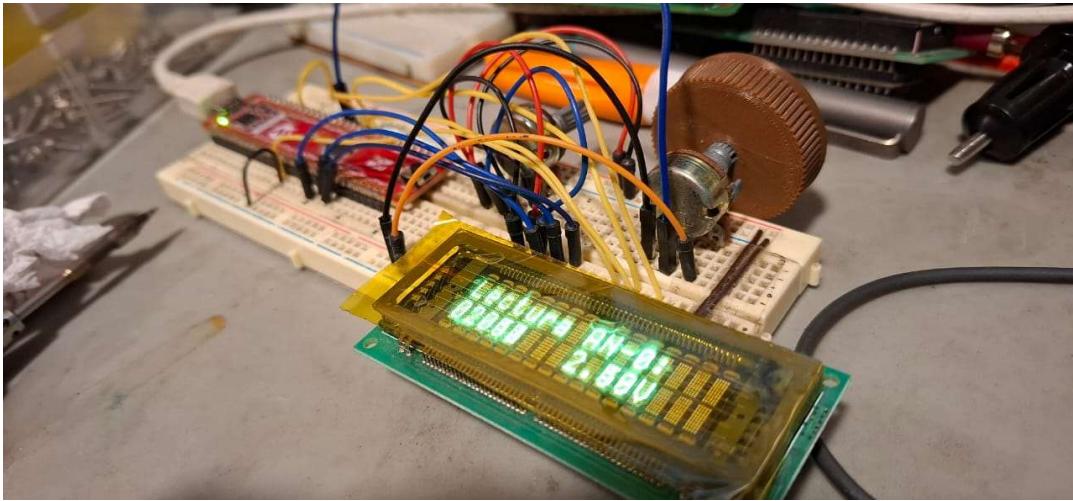
Modificación del main() para que se visualice el valor en voltaje

```

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

88

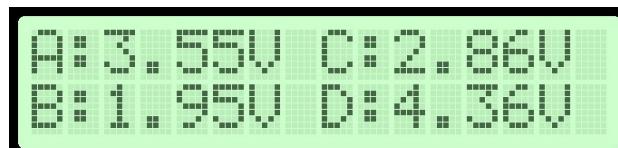
Implementación 2024-1



89

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.



A: 3.55V C: 2.86V
B: 1.95V D: 4.36V

- 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.

90

Asignación Semana 10 2024-1

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

Channel RA0:
Voltage: 3.87V

Channel RB4:
Voltage: 1.85V

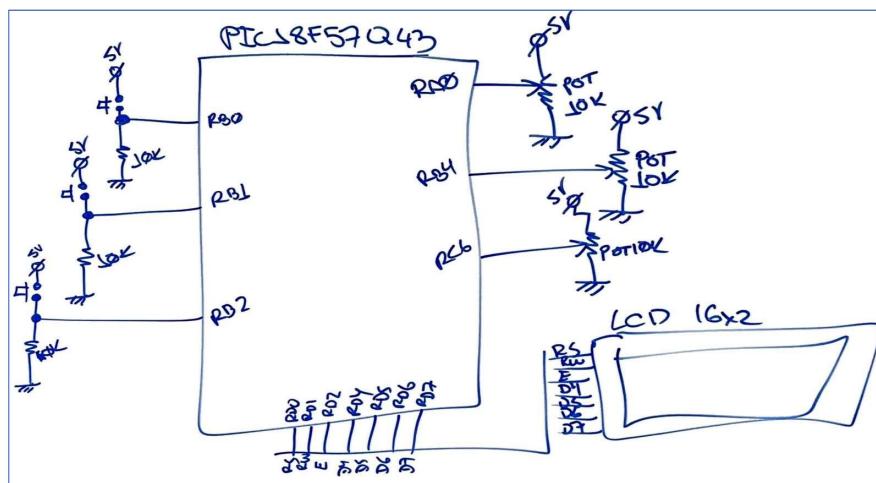
Channel RC6:
Voltage: 4.32V

- El intercambio de pantallas será al pulsar un botón activo en alto en INT0 para que se visualice la pantalla de RA0, al pulsar el botón activo en alto en INT1 para que se visualice la pantalla de RB4 y al pulsar el botón 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.

91

Asignación Semana 10 2024-1

- Hardware



```
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 = 0x60; //NOSC=HFINTOSC, NDIV=1:1
16     OSCFRC = 0x06; //HFINTOSC a 32MHz
17     OSCEN = 0x40; //HFINTOSC enabled
18     //config de las I/O
19     TRISAbits.TRISA0 = 1; //RA0 entrada
20     ANSELAbits.ANSELA0 = 1 //RA0 analógica
21     TRISB = 0xFF; //RB0 todas entradas
22     ANSELB = 0x18; //RB4 analógica, RB2 RB1 RB0 digitales
23     TRISBbits.TRISC6 = 1; //RC6 entrada
24     ANSELBbits.ANSELC6 = 1 //RC6 analógica
25     //configuración de ADC
26     ADCON0 = 0x94;
27     //config de las ints
28     PIE1bits.INT0IE = 1;
29     PIE1bits.INT1IE = 1;
30     PIE10bits.INT2IE = 1;
31     INTCONbits.GIE = 1;
32 }
33
34 unsigned int tomamuestra_ADC(unsigned char canal){
35     ADCPH = canal; //canal RA0
36     ADCCONbits.GO = 1; //inicio conversion
37     while(ADCCONbits.GO == 1); //espero a que termine de convertir
38     return((ADRESH & 0x08) + ADRESL);
39 }
40
41 void main(void){
42     unsigned int lectura;
43     float calculo;
44     unsigned int voltaje;
45     configuro();
46     LCD_inicializa();
47     splash_screen();
48     while(1){
49         switch(pantalla){
50             case 0:
51                 POS_CURSOR(1,0);
52                 ESCRIBE_MENSAJE("Channel RA0:",12);
53                 POS_CURSOR(2,0);
54                 ESCRIBE_MENSAJE("Voltage: ", 9);
55                 lectura = tomamuestra_ADC(0x00); //toma muestra de RA0
56                 calculo = lectura * 0.122;
57                 voltaje = calculo;
58                 ENVIA_CHAR((voltaje % 1000) / 100 + 0x30); //muestra centena
59                 ENVIA_CHAR('.');
60                 ENVIA_CHAR((lectura % 100) / 10 + 0x30); //muestra decena
61                 ENVIA_CHAR((lectura % 10) + 0x30); //muestra unidad
62                 ENVIA_CHAR('V');
63                 break;
64             case 1:
65                 POS_CURSOR(1,0);
66                 ESCRIBE_MENSAJE("Channel RB4:",12);
67                 POS_CURSOR(2,0);
68                 ESCRIBE_MENSAJE("Voltage: ", 0x0C);
69                 lectura = tomamuestra_ADC(0x00); //toma muestra de RB4
70                 calculo = lectura * 0.122;
71                 voltaje = calculo;
72                 ENVIA_CHAR((voltaje % 1000) / 100 + 0x30);
73                 ENVIA_CHAR('.');
74                 ENVIA_CHAR((lectura % 100) / 10 + 0x30);
75                 ENVIA_CHAR((lectura % 10) + 0x30);
76                 ENVIA_CHAR('V');
77                 break;
78         }
79     }
80 }
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140 }
```

Solución

93

Anexos: Ejemplos similares a los anteriores

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, 0x0A, 0x15, 0x11, 0x11, 0x0A, 0x04, 0x00};
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     INT0PPS = 0x0C; //ruteando INT0 a RB4
21     INTCON0bits.INT0EDG = 0; //INT0 falling edge
22     PIEbits.INT0IE = 1; //habilita INT0
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     GENERACARACTER(corazon, 0);
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     ENVIA_CHAR(0);
46
47     while(1){
48
49     }
50 }

```

```

52 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
53     PIR1bits.INT0IF = 0; //bajamos bandera INT0IF
54 }
55
56 void __interrupt(irq(default)) default_ISR(void){
57 }

```

95

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=1:1
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 enabled
20     INT0PPS = 0x0C; //redirigir señal
21     INTCON0bits.INT0EDG = 0; //INT0 detección
22     PIEbits.INT0IE = 1; //INT0 habilitada
23     INTCON0bits.GIE = 1; //switch global de ints habilitado
24 }

```

```

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 void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
41     PIR1bits.INT0IF = 0; //bajamos la bandera INT0IF
42     if(LED_estado == LED_apagado){
43         LED_estado = LED_parpadeo;
44     }
45     else{
46         LED_estado = LED_apagado;
47     }
48 }
49
50 void __interrupt(irq(default)) default_ISR(void){
51 }
52
53 }

```

96

```

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 configuracion(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.ANSELFB3 = 0; //RF3 digital
18     TRISBbits.TRISB4 = 1; //RB4 entrada
19     ANSELBBits.ANSELB4 = 0; //RB4 digital
20     WPUBbits.WPUB4 = 1; //RB4 pullup enabled
21     INTOPPS = 0x0C; //redirigir señal
22     INTCONOBits.INTOEDG = 0; //INTO detección
23     PIE1bits.INTOIE = 1; //INTO habilitado
24     INTCONOBits.GIE = 1; //switch global
25 }
26
27 void LCD_init(void){
28     TRISD = 0x00;
29     ANSELD = 0x00;
30     __delay_ms(10);
31     LCD_CONFIG();
32     __delay_ms(10);
33     BORRAR_LCD();
34     CURSOR_HOME();
35     CURSOR_ONOFF(OFF);
36 }
37
38 void main(void) {
39     configuracion();
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 }
57
58 void __interrupt(irq(IRQ_INTO)) INTO_ISR(void){
59     PIR1bits.INTOIF = 0; //abajamos la bandera INTOIF
60     if(LED_estado == LED_apagado){
61         LED_estado = LED_parpadeo;
62     }
63     else{
64         LED_estado = LED_apagado;
65     }
66 }
67
68 void __interrupt(irq(default)) default_ISR(void){
69 }
70
71

```

Implementación 2024-1

97

Ejemplo 2024-2

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

```

9 #include <xc.h>
10 #include "cabecera.h"
11
12 #define _XTAL_FREQ 32000000UL
13
14 void configuracion(void){
15     //config fuente de reloj
16     OSCCON1 = 0x60; //HFINTOSC POSTS 1:1
17     OSCFRQ = 0x06; //HFINTOSC 32MHz
18     OSCEN = 0x40; //HFINTOSC enabled
19     //config de las E/S
20     TRISBbits.TRISB4 = 1; //RB4 entrada
21     ANSELBBits.ANSELB4 = 0; //RB4 digital
22     WPUBbits.WPUB4 = 1; //RB4 pullup enabled
23     TRISBbits.TRISB0 = 0; //RB0 salida
24     ANSELBBits.ANSELB0 = 0; //RB0 digital
25 }
26
27 void main(void) {
28     configuracion();
29     while(1){
30         if(PORTBbits.RB4 == 0){
31             LATBbits.LATB0 = 1; //enciende el LED
32             __delay_ms(100);
33             LATBbits.LATB0 = 0; //apaga el LED
34             __delay_ms(100);
35         }
36         else{
37             LATBbits.LATB0 = 1; //enciende el LED
38             __delay_ms(300);
39             LATBbits.LATB0 = 0; //apaga el LED
40             __delay_ms(300);
41         }
42     }
43 }

```

98

Ejemplo 2024-2

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

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

99

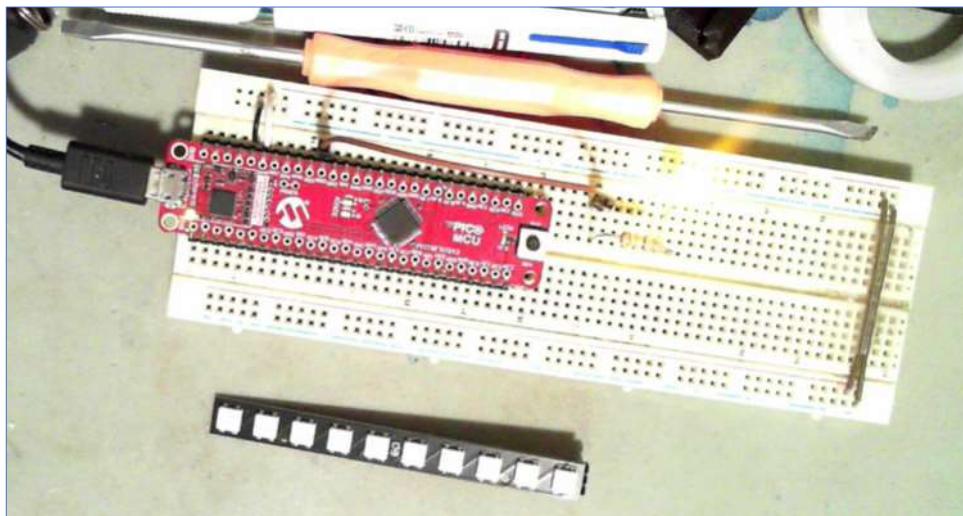
Ejemplo 2024-2

```

9  #include <xc.h>
10 #include "cabecera.h"
11
12 #define _XTAL_FREQ 32000000UL
13 #define LENTO 0
14 #define RAPIDO 1
15
16 unsigned char estado = LENTO; //variable que contiene el tiempo
17
18 void configuro(void){
19     //config fuente de reloj
20     OSCCON1 = 0x60; //HFINTOSC POSTS 1:1
21     OSCFRQ = 0x06; //HFINTOSC 32MHz
22     OSCEN = 0x40; //HFINTOSC enabled
23
24     //config de las E/S
25     TRISBbits.TRISB4 = 1; //RB4 entrada
26     ANSELBbits.ANSELB4 = 0; //RB4 digital
27     WPUBbits.WPUB4 = 1; //RB4 pullup enabled
28     TRISBbits.TRISB0 = 0; //RB0 salida
29     ANSELBbits.ANSELB0 = 0; //RB0 digital
30
31     //config de la INT0
32     INTOPPS = 0x0C; //la INT0 entra por RB4
33     INTCON0bits.INT0EDG = 0; //detección activo en alto
34     PIE1bits.INT0IE = 1; //habilitamos INT0
35     INTCON0bits.GIE = 1; //encendemos interrupciones
36
37     //sistema del RB4 en modo latch con interrupciones
38     void main(void) {
39         configuro();
40         while(1){
41             if(estado == RAPIDO){
42                 LATBbits.LATB0 = 1; //enciende el LED
43                 __delay_ms(100);
44                 LATBbits.LATB0 = 0; //apaga el LED
45                 __delay_ms(100);
46             }
47             else{
48                 LATBbits.LATB0 = 1; //enciende el LED
49                 __delay_ms(300);
50                 LATBbits.LATB0 = 0; //apaga el LED
51                 __delay_ms(300);
52             }
53         }
54     }
55
56     void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
57         PIR1bits.INT0IF = 0; //bajamos bandera de INT0
58         if(estado == LENTO){
59             estado = RAPIDO;
60         }
61         else{
62             estado = LENTO;
63         }
64     }
65 }
```

100

Ejemplo 2024-2



101

Ejemplo 2024-2

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

102

Ejemplo 2024-2

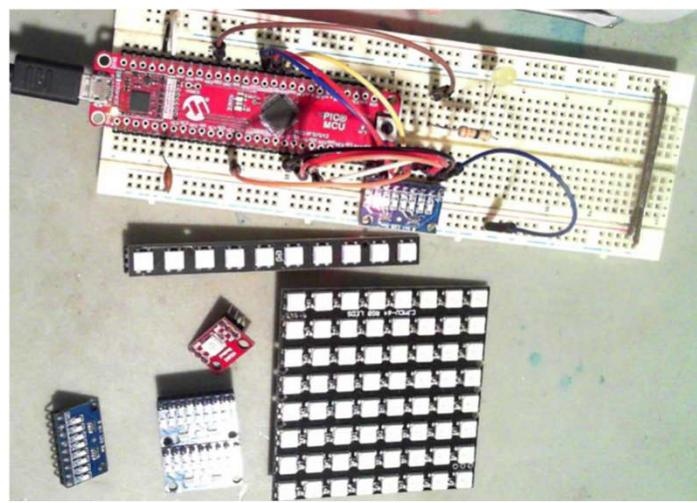
```

8  #include <xc.h>
9  #include "cabecera.h"
10 #define _XTAL_FREQ 32000000UL
11 #define E_LENTO 0
12 #define E_RAPIDO 1
13
14 unsigned char estado=E_LENTO;
15
16 void configuro(void){
17     //config de la fuente de reloj
18     OSCCON1 = 0x60;      //HFINTOSC POSTS 1:1
19     OSCFRQ = 0x06;       //HFINTOSC a 32MHz
20     OSCEN = 0x40;        //HFINTOSC enabled
21     //config de las E/S
22     TRISBbits.TRISB4 = 1; //RB4 como entrada
23     ANSELBbits.ANSELB4 = 0; //RB4 como digital
24     WPUBbits.WPUB4 = 1;   //RB4 con pullup
25     TRISBbits.TRISB0 = 0; //RB0 como salida
26     ANSELBbits.ANSELB0 = 0; //RB0 como digital
27     //config de la INT1
28     INT1PFS = 0x0C;      //RB4 sea la INT1
29     INTCON0bits.INT1EDG = 0; //INT1 en falling-edge
30     PIE6bits.INT1IE = 1;  //INT1 enabled
31     INTCON0bits.GIE = 1;  //GIE enabled
32 }
33
34 void condicion(void){
35     if(estado == E_LENTO){
36         __delay_ms(500);
37     }
38     else{
39         __delay_ms(100);
40     }
41 }
42
43 void main(void) {
44     configuro();
45     while(1){
46         LATBbits.LATB0 = 1;
47         condicion();
48         LATBbits.LATB0 = 0;
49         condicion();
50     }
51 }
52
53 void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
54     PIR6bits.INT1IF = 0; //abajamos la bandera
55     if(estado == E_LENTO){
56         estado = E_RAPIDO;
57     }
58     else{
59         estado = E_LENTO;
60     }
61 }
62
63 void __interrupt(irq(default)) DEFAULT_ISR(void){
64
65 }

```

103

Ejemplo 2024-2



104

Ejemplo 2024-2

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

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

105

Ejemplo 2024-2

```

8  #include <xc.h>
9  #include "cabecera.h"
10 #define _XTAL_FREQ 32000000UL
11 #define RAPIDISIMO 1
12 #define LENTISIMO 0
13
14 unsigned char ESTADO = LENTISIMO;
15
16 void configuro(void){
17     //config de la fuente de reloj
18     OSCCON1 = 0x60;      //NOSC=HFINTOSC POSTS 1:1
19     OSCFRQ = 0x06;       //HFINTOSC a 32MHz
20     OSCEN = 0x40;        //HFINTOSC enabled
21
22     //config las E/S
23     TRISBbits.TRISB4 = 1; //RB4 como entrada
24     ANSELBbits.ANSELB4 = 0; //RB4 como digital
25     WPUBbits.WPUB4 = 1;   //RB4 pullup enabled
26     TRISBbits.TRISB0 = 0; //RB0 como salida
27     ANSELBbits.ANSELB0 = 0; //RB0 como digital
28
29     //config de la INT1
30     INT1PPS = 0x0C;      //Asignando RB4 para INT1
31     INTCON0bits.INT1EDG = 0; //INT1 en flanco negativo
32     PIE6bits.INT1IE = 1;   //habilitando INT1
33     INTCON0bits.GIE = 1;   //habilitador GIE en uno
34     PIR6bits.INT1IF = 0;   //bajamos la bandera inicialmente
35
36     void main(void){
37         configuro();
38         while(1){
39             LATBbits.LATB0 = 1;
40             if(ESTADO == LENTISIMO){
41                 __delay_ms(500);
42             }
43             else if(ESTADO == RAPIDISIMO){
44                 __delay_ms(100);
45             }
46             LATBbits.LATB0 = 0;
47             if(ESTADO == LENTISIMO){
48                 __delay_ms(500);
49             }
50             else if(ESTADO == RAPIDISIMO){
51                 __delay_ms(100);
52             }
53         }
54     }
55     void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
56         PIR6bits.INT1IF = 0; //bajamos la bandera de INT1
57         if(ESTADO == RAPIDISIMO){
58             ESTADO = LENTISIMO;
59         }
60         else{
61             ESTADO = RAPIDISIMO;
62         }
63     }
64 }
```

106

Ejercicio 2024-2

```
8  #include <xc.h>
9  #include "cabecera.h"
10 #define _XTAL_FREQ 32000000UL
11 #define RAPIDO 1
12 #define LENTO 0
13
14 unsigned char ESTADO = LENTO;
15
16 void configuro(void){
17     //config la fuente de reloj
18     OSCCON1 = 0x60;      //NOSC=HFINTOSC POSTS=1:1
19     OSCFRQ = 0x06;       //HFINTOSC a 32MHz
20     OSCEN = 0x40;        //HFOSC enabled
21
22     //config de las E/S
23     TRISBbits.TRISB4 = 1; //RB4 como entrada
24     ANSELBbits.ANSELB4 = 0; //RB4 como digital
25     WPUBbits.WPUB4 = 1;   //RB4 pullup enabled
26     TRISBbits.TRISB0 = 0; //RB0 como salida
27     ANSELBbits.ANSELB0 = 0; //RB0 como digital
28
29     //config INT1
30     INT1PPS = 0x0C;      //asignar INT1 hacia RB4
31     INTCON0bits.INT1EDG = 0; //INT1 en falling_edge
32     PIE6bits.INT1IE = 1;   //INT1 enabled
33     INTCON0bits.GIE = 1;   //interrupciones habilitadas
34     PIR6bits.INT1IF = 0;   //bajamos bandera de INT1
35
36
37
38
39     else if(ESTADO == RAPIDO){
40         _delay_ms(100);
41     }
42 }
43
44 void main(void) {
45     configuro();
46     while(1){
47         LATBbits.LATB0 = 1; //enciende el LED
48         pregunta();
49         LATBbits.LATB0 = 0; //apago el LED
50         pregunta();
51     }
52 }
53
54 void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
55     PIR6bits.INT1IF = 0; //bajamos bandera INT1
56     if(ESTADO == LENTO){
57         ESTADO = RAPIDO;
58     }
59     else if(ESTADO == RAPIDO){
60         ESTADO = LENTO;
61     }
62 }
```

107

Ejercicio 2024-2 Mejora aplicando LATCH

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

108

Ejercicio mejorado 2024-2 EL56-1

```

8  #include <xc.h>
9  #include "cabecera.h"
10 #define _XTAL_FREQ 32000000UL
11 #define RAPIDO 1
12 #define LENTITO 0
13
14 unsigned char ESTADO = LENTITO;
15
16 void configuro(void){
17     //config la fuente de reloj
18     OSCCON1 = 0x60;      //NOSC=HFINTOSC POSTS=1:1
19     OSCFRQ = 0x06;       //HFINTOSC a 32MHz
20     OSCEN = 0x40;        //HFINTOSC enabled
21     //config de las E/S
22     TRISBbits.TRISB4 = 1; //RB4 es entrada
23     ANSELBbits.ANSELB4 = 0; //RB4 es digital
24     WPUBbits.WPUB4 = 1;   //RB4 pullup enabled
25     TRISBbits.TRISB0 = 0; //RB0 es salida
26     ANSELBbits.ANSELB0 = 0; //RB0 es digital
27     //config de la INT2
28     INT2PPS = 0x0C;      //asignando RB4 para INT2
29     INTCON0bits.INT2EDG = 0; //falling_edge para INT2
30     PIE10bits.INT2IE = 1; //activo INT2
31     INTCON0bits.GIE = 1;  //habilitador global de ints ON
32     PIR10bits.INT2IF = 0; //bajamos bandera INT2
33 }
34
35 void pregunta(void){
36     if(ESTADO == RAPIDO){
37         __delay_ms(100);
38     }
39     else if(ESTADO == LENTITO){
40         __delay_ms(500);
41     }
42 }
43
44 void main(void){
45     configuro();
46     while(1){
47         LATBbits.LATB0 = 1; //enciendo el LED
48         pregunta();
49         LATBbits.LATB0 = 0; //apagado el LED
50         pregunta();
51     }
52 }
53
54 void __interrupt(irq(IRQ_INT2)) INT2_ISR(void){
55     PIR10bits.INT2IF = 0; //bajamos bandera de INT2
56     if(ESTADO == LENTITO){
57         ESTADO = RAPIDO;
58     }
59     else if(ESTADO == RAPIDO){
60         ESTADO = LENTITO;
61     }
62 }

```

109

Ejercicio mejorado 2024-2 EL56-2

```

8  #include <xc.h>
9  #include "cabecera.h"
10 #define _XTAL_FREQ 32000000UL
11 #define LENTASO 0
12 #define RAPIDASO 1
13
14 unsigned char ESTADO = LENTASO;
15
16 void configuro(void){
17     //config la fuente reloj
18     OSCCON1 = 0x60;      //NOSC=HFINTOSC NDIV=1:1
19     OSCFRQ = 0x06;       //HFINTOSC a 32MHz
20     OSCEN = 0x40;        //HFINTOSC enabled
21     //config las E/S
22     TRISBbits.TRISB0 = 0; //RB0 sea salida
23     ANSELBbits.ANSELB0 = 0; //RB0 sea digital
24     TRISBbits.TRISB4 = 1; //RB4 como entrada
25     ANSELBbits.ANSELB4 = 0; //RB4 como digital
26     WPUBbits.WPUB4 = 1;   //RB$ pull-up enabled
27     //config de la INT2
28     INT2PPS = 0x0C;      //INT2 tenga asignado el RB4
29     INTCON0bits.INT2EDG = 0; //INT2 en flanko negativo
30     PIE10bits.INT2IE = 1; //INT2 habilitada
31     INTCON0bits.GIE = 1;  //habilitador global encendido
32     PIR10bits.INT2IF = 0; //bajamos la bandera de INT2
33 }
34
35 void pregunta_delay(void){
36     if(ESTADO == RAPIDASO){
37         __delay_ms(100);
38     }
39     if(ESTADO == LENTASO){
40         __delay_ms(500);
41     }
42 }
43
44 void main(void) {
45     configuro();
46     while(1){
47         LATBbits.LATB0 = 1; //enciendo LED
48         pregunta_delay();
49         LATBbits.LATB0 = 0; //apagar LED
50         pregunta_delay();
51     }
52 }
53
54 void __interrupt(irq(IRQ_INT2)) INT2_ISR(void){
55     PIR10bits.INT2IF = 0; //bajamos la bandera de INT2
56     if(ESTADO == LENTASO){
57         ESTADO = RAPIDASO;
58     }
59     else if(ESTADO == RAPIDASO){
60         ESTADO = LENTASO;
61     }
62 }
63
64 void __interrupt(irq(default)) default_ISR(void){
65 }
66

```

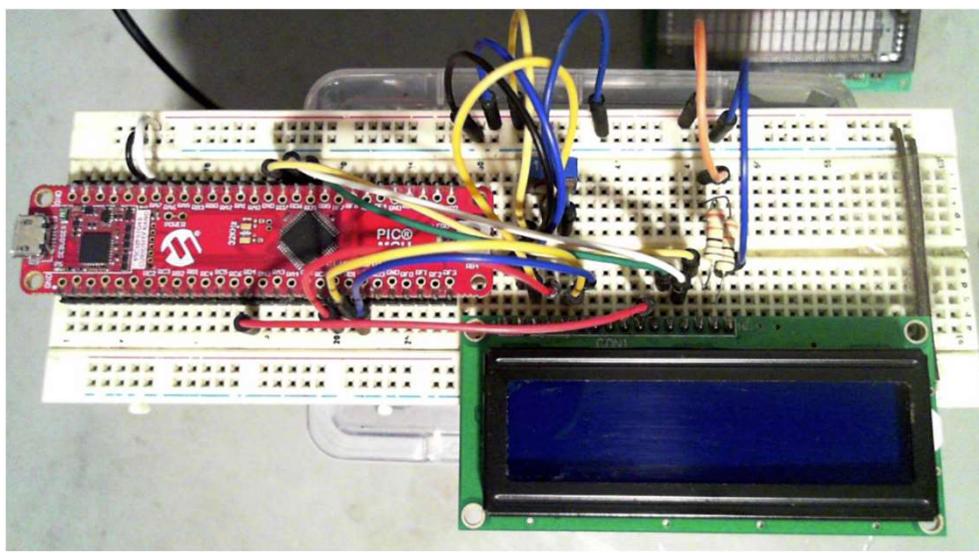
110

Ejercicios 2024-2 Semana 10

111

Ejercicio 2024-2

- Circuito implementado



112

Ejercicio 2024-2

- Funcionamiento:



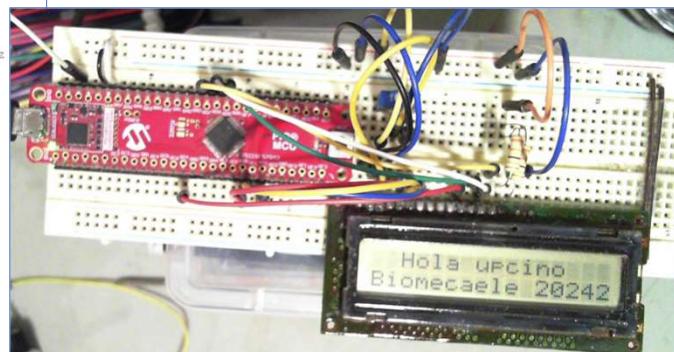
113

Ejercicio 2024-2

- Ejemplo:

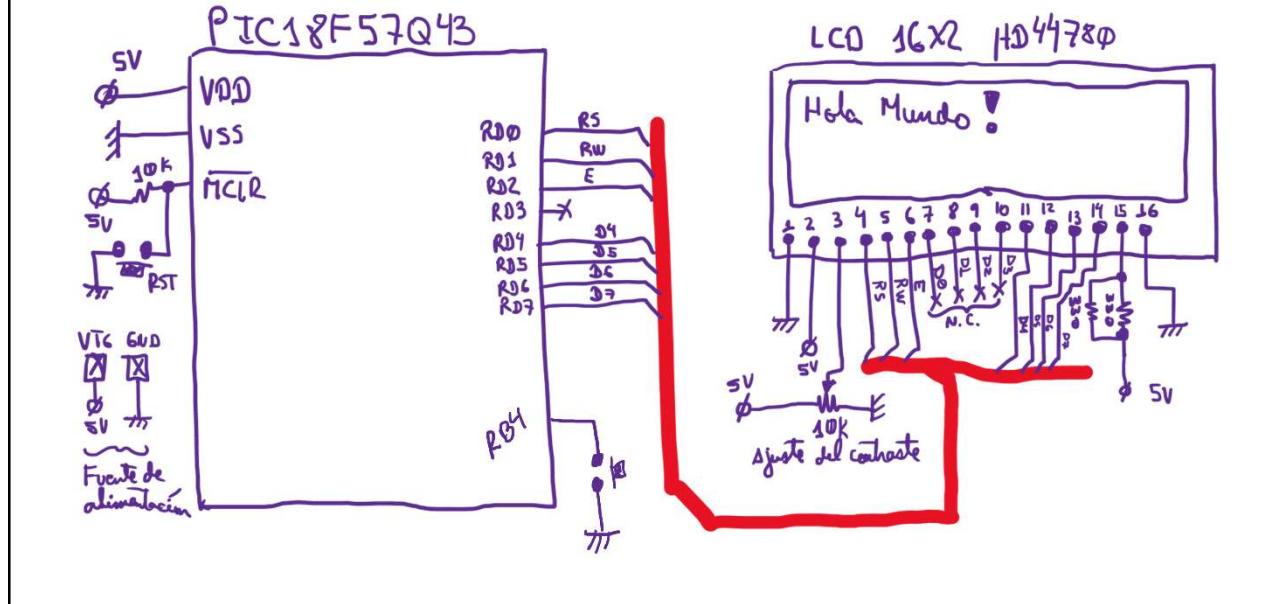
```

8  #include <xc.h>
9  #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 void configuro(void){
14     //config de la fuente de reloj
15     OSCCON1 = 0x60;
16     OSCFRQ = 0x06; //HFINTOSC a 32MHz
17     OSCEN = 0x40;
18 }
19
20 void lcd_init(void){
21     TRISD = 0x00;
22     ANSEL0 = 0x00;
23     LCD_CONFIG();
24     _delay_ms(17);
25     BORRAR_LCD();
26     CURSOR_HOME();
27     CURSOR_ONOFF(OFF);
28 }
29
30 void main(void) {
31     configuro();
32     lcd_init();
33     POS_CURSOR(1,2);
34     ESCRIBE_MENSAJE("Hola upcino",11);
35     POS_CURSOR(2,0);
36     ESCRIBE_MENSAJE("Biomecaele 20242",16);
37     while(1){
38     }
39 }
```



114

Ej. 2024-2 Agregando la función de un pulsador



115

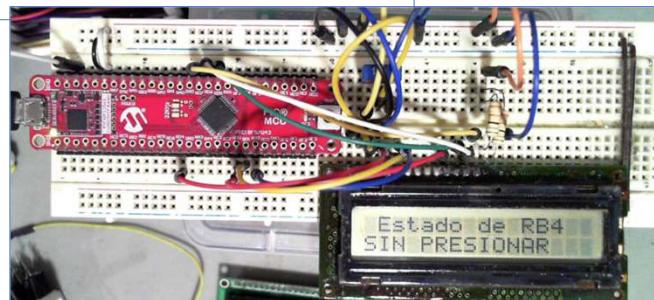
Ej. 2024-2 Agregando la función de un pulsador

```

8  #include <xc.h>
9  #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 void configuro(void){
14     //config de la fuente de reloj
15     OSCCON1 = 0x60;
16     OSCFRQ = 0x06; //HFINTOSC a 32MHz
17     OSCEN = 0x40;
18     //config de las E/S
19     TRISBbits.TRISB4 = 1; //RB4 input
20     ANSELBbits.ANSELB4 = 0; //RB4 digital
21     WPUBBbits.WPUB4 = 1; //pullup enabled
22 }
23
24 void lcd_init(void){
25     TRISD = 0x00;
26     ANSELD = 0x00;
27     LCD_CONFIG();
28     _delay_ms(17);
29     BORRAR_LCD();
30     CURSOR_HOME();
31     CURSOR_ONOFF(OFF);
32 }
```

```

34 void main(void) {
35     configuro();
36     lcd_init();
37     POS_CURSOR(1,1);
38     ESCRIBE_MENSAJE("Estado de RB4",13);
39
40     while(1){
41         if(PORTBbits.RB4 == 0){
42             POS_CURSOR(2,0);
43             ESCRIBE_MENSAJE("PRESIONADO ", 13);
44         }
45         else{
46             POS_CURSOR(2,0);
47             ESCRIBE_MENSAJE("SIN PRESIONAR", 13);
48         }
49     }
}
```



116

Ejercicio 2024-2

```

8  #include <xc.h>
9  #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 unsigned int resultado_ADC;
14 unsigned char millar, centena, decena, unidad;
15
16 void configuro(void){
17     //config fuente de reloj
18     OSCCON1 = 0x60;
19     OSCFRQ = 0x06;      //HFINTOSC a 32MHz
20     OSCEN = 0x40;
21     //config de las E/S
22     TRISBbits.TRISB4 = 1; //RB4 entrada
23     ANSELBbits.ANSELB4 = 0; //RB4 digital
24     WPUBbits.WPUB4 = 1;   //RB4 pullup activado
25     TRISAbits.TRISA0 = 1; //RA0 entrada
26     ANSELAbits.ANSEL0A = 1; //RA0 analogica
27     //config del ADC
28     ADCON0bits.FM = 1;    //just a la derecha
29     ADCON0bits.CS = 1;    //fuente de reloj ADCRC
30     ADPC = 0x00;          //canal RA0/ANAO seleccionado
31     ADREF = 0x00;
32     ADCON0bits.ADON = 1; //encendemos el ADC
33 }
34
35 void LCD_INIT(void){
36     TRISD = 0x00;
37     ANSELD = 0x00;
38     LCD_CONFIG();
39     delay_ms(16);

```

• Conversión ADC: Canal RA0/ANAO

```

40     BORRAR_LCD();
41     CURSOR_HOME();
42     CURSOR_ONOFF(OFF);
43 }
44
45 unsigned int leer_ADC(void){
46     ADCON0bits.GO = 1; //inicia la toma de una muestra
47     while(ADCON0bits.GO == 1); //espera a que termine de convertir
48     //ya tenemos el resultado de la conversion ADRESH:ADRESL
49     return (ADRESH << 8) + ADRESL;
50 }
51
52 void conversion(unsigned int dato){
53     millar = dato / 1000;
54     centena = (dato % 1000) / 100;
55     decena = (dato % 100) / 10;
56     unidad = dato % 10;
57 }
58
59 void main(void) {
60     configuro();
61     LCD_INIT();
62     //me voy a la primera linea del LCD
63     POS_CURSOR(1,2);
64     ESCRIBE_MENSAJE("Ucino 2024-2", 13);
65     while(1){
66         resultado_ADC = leer_ADC();
67         conversion(resultado_ADC);
68         POS_CURSOR(2,0);
69         ESCRIBE_MENSAJE("Canal ANAO:", 11);
70         ENVIA_CHAR(millar+0x30);
71         ENVIA_CHAR(centena+0x30);
72         ENVIA_CHAR(decena+0x30);
73         ENVIA_CHAR(unidad+0x30);
74     }
75 }

```

117

```

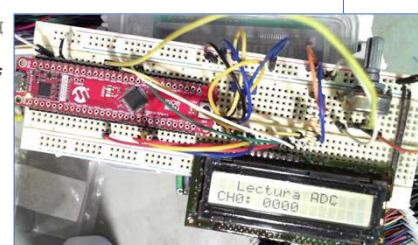
8  #include <xc.h>
9  #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 unsigned char millar, centena, decena, unidad;
14
15 void configuro(){
16     //config de la fuente de reloj
17     OSCCON1 = 0x60;
18     OSCFRQ = 0x06;      //HFINTOSC a 32MHz
19     OSCEN = 0x40;
20     //config de las E/S
21     TRISBbits.TRISB4 = 1; //RB4 input
22     ANSELBbits.ANSELB4 = 0; //RB4 digital
23     WPUBbits.WPUB4 = 1;   //RB4 pullup enabled
24     //config del ADC
25     TRISAbits.TRISA0 = 1; //RA0 input
26     ANSELAbits.ANSEL0A = 1; //RA0 analog
27     ADCON0bits.FM = 1;    //just a la derecha
28     ADCON0bits.CS = 1;    //fuente de reloj ADCRC
29     ADCON0bits.ADON = 1; //ADC encendido
30 }
31
32 void lcd_init(void){
33     TRISD = 0x00;
34     ANSELD = 0x00;
35     LCD_CONFIG();
36     delay_ms(17);
37     BORRAR_LCD();
38     CURSOR_HOME();
39     CURSOR_ONOFF(OFF);
40 }

```

```

42 unsigned int toma_muestra(void){
43     ADPC = 0x00;
44     ADCON0bits.GO = 1; //toma una muestra
45     while(ADCON0bits.GO == 1); //va a esperar a que termine de convertir
46     return ((ADRESH << 8) + ADRESL);
47 }
48
49 void convierte(unsigned int dato){
50     millar = dato / 1000;
51     centena = (dato % 1000) / 100;
52     decena = (dato % 100) / 10;
53     unidad = dato % 10;
54 }
55
56 void main(void) {
57     configuro();
58     lcd_init();
59     POS_CURSOR(1,2);
60     ESCRIBE_MENSAJE("Lectura ADC", 11);
61     while(1){
62         convierte(toma_muestra());
63         POS_CURSOR(2,0);
64         ESCRIBE_MENSAJE("CH0: ",5);
65         ENVIA_CHAR(millar+0x30);
66         ENVIA_CHAR(centena+0x30);
67         ENVIA_CHAR(decena+0x30);
68         ENVIA_CHAR(unidad+0x30);
69         delay_ms(50);
70     }
71 }

```



• Conversión ADC: Canal RA0/ANAO

118

```

8  #include <xc.h>
9  #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 unsigned char millar,centena,decena,unidad;
14
15 void configuro(void){
16     //config fuente de reloj
17     OSCCON1 = 0x60; //NOSC=HFINTOSC NDIV=1:1
18     OSCFRQ = 0x06; //HFINTOSC a 32MHz
19     OSCEN = 0x40; //HFINTOSC enabled
20     //config del RB4
21     TRISBbits.TRISB4 = 1; //RB4 como entrada
22     ANSELBbits.ANSELB4 = 0; //RB4 como digital
23     WPUBBbits.WPUB4 = 1; //RB4 con pullup
24     //config el ADC
25     TRISAbits.TRISA0 = 1; //RA0 es entrada
26     ANSELAbits.ANSELA0 = 1; //RA0 es analogica
27     ADCON0bits.FM = 1; //resultado justo derecho
28     ADCON0bits.CS = 1; //fuente de reloj del ADC en ADCRC
29     ADCON0bits.ADON = 1; //enciendo el ADC
30 }
31
32 unsigned int tomamuestra_ADC(void){
33     ADPCH = 0; //selecciono el canal RA0/ANAO
34     ADCON0bits.GO = 1; //procede a tomar una muestra
35     while(ADCON0bits.GO == 1); //espera a que termine
36     return ((ADRESH << 8) + ADRESL);
37 }
38
39 void convierte(unsigned int dato){
40     millar = dato / 1000;
41     centena = (dato % 1000) / 100;
42     decena = (dato % 100) / 10;
43     unidad = dato % 10;
44 }
45
46 void lcd_init(void){
47     TRISD = 0x00;
48     ANSELD = 0x00;
49     LCD_CONFIG();
50     __delay_ms(21);
51     BORRAR_LCD();
52     CURSOR_HOME();
53     CURSOR_ONOFF(OFF);
54 }
55
56 void main(void) {
57     configuro();
58     lcd_init();
59     POS_CURSOR(1,0);
60     ESCRIBE_MENSAJE("Tercer ejemplo:",15);
61     while(1){
62         convierte(tomamuestra_ADC());
63         POS_CURSOR(2,0);
64         ESCRIBE_MENSAJE("CHO:",4);
65         ENVIA_CHAR(millar+0x30);
66         ENVIA_CHAR(centena+0x30);
67         ENVIA_CHAR(decena+0x30);
68         ENVIA_CHAR(unidad+0x30);
69     }
70 }

```

119

```

8  #include <xc.h>
9  #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 void configuro(void){
14     //conf la fuente de reloj
15     OSCCON1 = 0x60; //NOSC=HFINTOSC, NDIV=1:1
16     OSCFRQ = 0x06; //HFINTOSC a 32MHz
17     OSCEN = 0x40; //HFINTOSC enabled
18 }
19
20 void lcd_init(void){
21     TRISD = 0x00;
22     ANSELD = 0x00;
23     LCD_CONFIG();
24     __delay_ms(23);
25     BORRAR_LCD();
26     CURSOR_HOME();
27     CURSOR_ONOFF(OFF);
28     __delay_ms(50);
29 }
30
31 void main(void) {
32     configuro();
33     lcd_init();
34     POS_CURSOR(1,0);
35     ESCRIBE_MENSAJE("Hola mundo UPC!",15);
36     POS_CURSOR(2,0);
37     ESCRIBE_MENSAJE("Microcontrolador",16);
38     while(1){
39     }
40 }

```

- Visualización de mensajes en el LCD con la librería LCD

120

```

8 #include <xc.h>
9 #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 unsigned char millar, centena, decena, unidad;
14
15 void configuro(void){
16     //config la fuente de reloj
17     OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
18     OSCFRC = 0x06;       //HFINTOSC a 32MHz
19     OSCEN = 0x40;        //HFINTOSC enabled
20
21     //config el ADC
22     TRISAbits.TRISA0 = 1; //RA0 como entrada
23     ANSELAbits.ANSELAO = 1; //RA0 como analógica
24     ADCON0bits.FM = 1;    //resultado just a la derecha
25     ADCON0bits.CS = 1;    //fuente de reloj del ADC = ADCRC
26     ADCON0bits.ADON = 1;  //encendemos el adc
27 }
28
29 unsigned int tomamuestra_ADC(void){
30     ADPCH = 0; //seleccionamos RA0
31     ADCON0bits.GO = 1; //que toma una muestra
32     while(ADCON0bits.GO == 1); //espero a que termine de tomar la muestra
33     return ((ADRESH << 8) + ADRESL);
34 }
35
36 void conviertemon(unsigned int dato){
37     millar = dato / 1000;
38     centena = (dato % 1000) / 100;
39     decena = (dato % 100) / 10;
40     unidad = dato % 10;
41 }
42
43 void lcd_init(void){
44     TRISD = 0x00;
45     ANSELD = 0x00;
46     LCD_CONFIG();
47     __delay_ms(23);
48     BORRAR_LCD();
49     CURSOR_HOME();
50     CURSOR_ONOFF(OFF);
51 }
52
53 void main(void){
54     configuro();
55     lcd_init();
56     POS_CURSOR(1,0);
57     ESCRIBE_MENSAJE("Ejemplo #3",10);
58     while(1){
59         conviertemon(tomamuestra_ADC());
60         POS_CURSOR(2,0);
61         ESCRIBE_MENSAJE("RA0: ",5);
62         ENVIA_CHAR(millar+0x30);
63         ENVIA_CHAR(centena+0x30);
64         ENVIA_CHAR(decena+0x30);
65         ENVIA_CHAR(unidad+0x30);
66     }
67 }

```

- Conversión ADC: Canal RA0/ANAO

121

```

8 #include <xc.h>
9 #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 void configuro(void){
14     //config fuente de reloj
15     OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
16     OSCFRC = 0x06;       //HFINTOSC a 32MHz
17     OSCEN = 0x40;        //HFINTOSC enabled
18
19     //config del boton
20     TRISBbits.TRISB4 = 1; //RB4 como entrada
21     ANSELBbits.ANSELB4 = 0; //RB4 como digital
22     WPUBbits.WPUB4 = 1;   //RB4 con pullup activo
23 }
24
25 void lcd_init(void){
26     TRISD = 0x00;
27     ANSELD = 0x00;
28     LCD_CONFIG();
29     __delay_ms(26);
30     BORRAR_LCD();
31     CURSOR_HOME();
32     CURSOR_ONOFF(OFF);
33 }
34
35 /*void retardado(unsigned int valor){
36     unsigned int temporal;
37     for(temporal=0;temporal<valor;temporal++){
38         __delay_us(1);
39     }
40 } */
41
42 void main(void) {
43     configuro();
44     lcd_init();
45     POS_CURSOR(1,0);
46     ESCRIBE_MENSAJE("Ejemplo #2",10);
47     while(1){
48         if(PORTBbits.RB4 == 0){
49             //aqui cuando es verdad
50             POS_CURSOR(2,0);
51             ESCRIBE_MENSAJE("Machucado ",12);
52         }
53         else{
54             //aqui cuando es falso
55             POS_CURSOR(2,0);
56             ESCRIBE_MENSAJE("Sin machucar",12);
57         }
58     }
59 }

```

- Visualización del estado del pulsador en RB4

122

```

8 #include <xc.h>
9 #include "cabecera.h"
10 #include "LCD.h"
11 #define _XTAL_FREQ 32000000UL
12
13 unsigned char millar, centena, decena, unidad;
14
15 void configuro(void){
16     //config fuente de reloj
17     OSCCON1 = 0x60;      //NOSC=HFINTOSC, NDIV=1:1
18     OSCFRQ = 0x06;       //HFINTOSC a 32MHz
19     OSCEN = 0x40;        //HFINTOSC enabled
20
21     //config del boton
22     TRISBbits.TRISB4 = 1; //RB4 como entrada
23     ANSELBbits.ANSELB4 = 0; //RB4 como digital
24     WPUBbits.WPUB4 = 1;   //RB4 con pullup activo
25
26     //config ADC
27     TRISAbits.TRISA0 = 1; //RA0 como entrada
28     ANSELAbits.ANSELAO = 1; //RA0 como analogico
29     ADCON0bits.FM = 1;    //resultado justa derecha
30     ADCON0bits.CS = 1;   //fuente de reloj del ADC en ADCRC
31     ADCON0bits.ADON = 1; //encendemos el ADC
32
33     unsigned int tomamuestra_ADC(void){
34         ADPCH = 0;           //canal seleccionado RA0
35         ADCON0bits.GO = 1;  //toma una muestra
36         while(ADCON0bits.GO == 1); //espero a que termine la toma de muestra
37         return ((ADRESH << 8) + ADRESL);
38     }
39
40     conviertemon(unsigned int dato){
41         millar = dato / 1000;
42         centena = (dato % 1000) / 100;
43         decena = (dato % 100) / 10;
44         unidad = dato % 10;
45
46     void lcd_init(void){...9 lines}
47
48     void main(void) {
49         configuro();
50         lcd_init();
51         POS_CURSOR(1,0);
52         ESCRIBE_MENSAJE("Ejemplo #3",10);
53         while(1){
54             conviertemon(tomamuestra_ADC());
55             POS_CURSOR(2,0);
56             ESCRIBE_MENSAJE("Canal RA0:",10);
57             ENVIA_CHAR(millar+0x30);
58             ENVIA_CHAR(centena+0x30);
59             ENVIA_CHAR(decena+0x30);
60             ENVIA_CHAR(unidad+0x30);
61         }
62     }
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
764
765
766
766
767
767
768
768
769
769
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
158
```

Asignación Semana 10:

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

125

Asignación (Minilaboratorio) Semana 10

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

126

Asignación (Minilaboratorio) Semana 10

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

127

Fin de la sesión

128