

Microcontroladores

Semestre: 2021-0

Profesor: Kalun José Lau Gan

Semana 5: Lenguajes de alto nivel en microcontroladores

1

¿Preguntas previas?

- ¿Veremos lenguaje C?
 - Si, se atiende lenguaje de alto nivel en la segunda parte del semestre
- ¿Cuántas semanas tiene el semestre de verano?
 - Ocho semanas, la última semana no hay clases y se destina para la evaluación final, en nuestro caso la presentación de su trabajo final.
- Sobre el trabajo final:
 - Grupos máximo de 4 integrantes, agrupación depende de su afinidad, tema orientado a problemática de pandemia.

2

Agenda:

- Los lenguajes de alto nivel, en nuestro caso el XC8
- El MPLAB Xpress
- Desarrollo de un proyecto plantilla en el MPLAB X para el XC8

3

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

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

4

El compilador XC de Microchip

- Disponible versión gratis sin límite

5

El compilador XC de Microchip

- XC8, XC16 y XC32 son compiladores del XC y están separados...
- La versión XC8 Pro es una licencia con características de optimización de código.



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

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

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

[More Info >](#)

Standard Pricing:

Order Quantity

1+

USD per Unit
\$1,695.00

In Stock: 9
Delivery and scheduling options available in the cart [+>](#)

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

Lead Time For Additional Quantities [?>](#)

Additional quantities can ship by **11-Aug-2020**

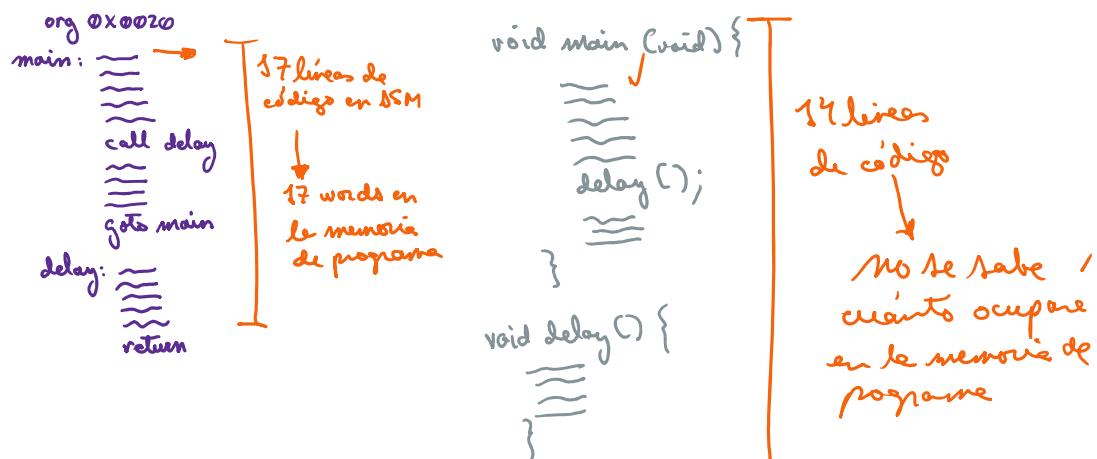
Quantity:



6

Eficiencia de código en Assembler vs C

¿Por qué Assembler y por qué C?



7

MPLAB X: Creación de un proyecto en XC8

- Link del manual de XC8:
 - <http://ww1.microchip.com/downloads/en/devicedoc/50002053g.pdf>
- Link de descarga del XC8 v2.20:
 - <https://www.microchip.com/mplabxc8windows>

8

Plantilla de código en XC8:

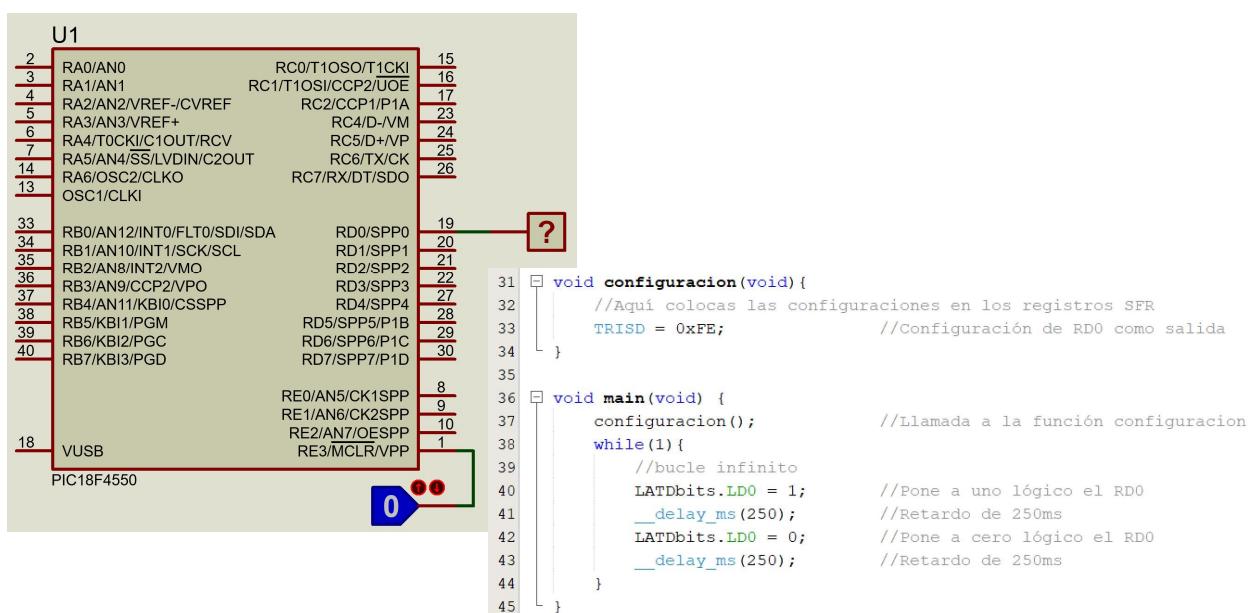
```

10  #pragma config PLLDIV = 1           // PLL Prescaler Selection bit
11  #pragma config CPUDIV = OSC1_PLL2 // System Clock Postscaler Selection
12  #pragma config FOSC = XTPLL_XT   // Oscillator Selection bits (XT)
13  #pragma config PWRT = ON          // Power-up Timer Enable bit
14  #pragma config BOR = OFF          // Brown-out Reset Enable bits
15  #pragma config BORV = 3           // Brown-out Reset Voltage bit
16  #pragma config WDT = OFF          // Watchdog Timer Enable bit
17  #pragma config WDTPS = 32768      // Watchdog Timer Postscale Selection
18  #pragma config CCP2MX = ON         // CCP2 MUX bit (CCP2 input/output)
19  #pragma config PBADEN = OFF        // PORTB A/D Enable bit (PORTB & PORTC)
20  #pragma config MCLRE = ON          // MCLR Pin Enable bit (MCLR pin internal pull-up)
21  #pragma config LVP = OFF           // Single-Supply ICSP Enable bit
22
23  #include <xc.h>
24
25  #define _XTAL_FREQ 48000000UL     //Frecuencia de trabajo 48MHz
26
27  void configuracion(void) {
28      //Aqui colocas las configuraciones iniciales
29  }
30
31  void main(void) {
32      configuracion();
33      while (1) {
34          //Tu programa de usuario
35      }
36  }

```

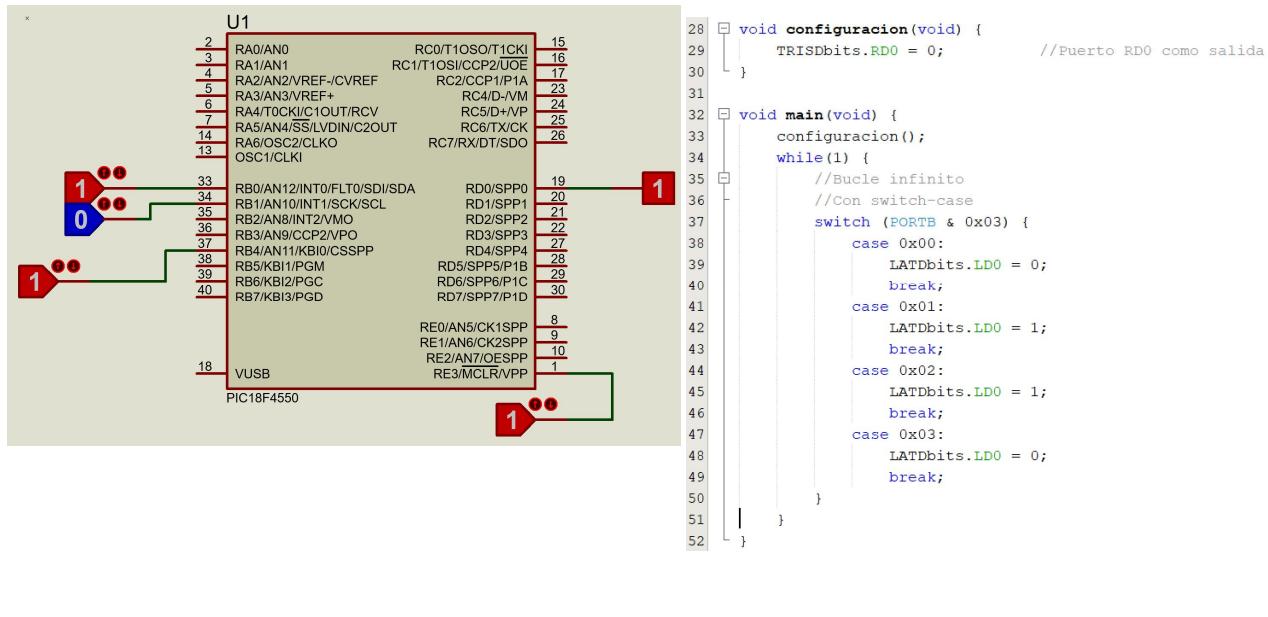
9

Ejemplo en XC8: Titileo de LED



10

Ejemplo en XC8: Compuerta XOR



11

Ejemplo:
Señal de
cruce de tren
con entrada
de activación

```

9 #pragma config PLLDIV = 1      //
10 #pragma config CPUDIV = OSC1_PLL2/
11 #pragma config FOSC = XTPLL_XT //
12 #pragma config FWRT = ON        //
13 #pragma config BOR = OFF       //
14 #pragma config BORV = 3        //
15 #pragma config WDT = OFF       //
16 #pragma config WDTPS = 32768   //
17 #pragma config CCP2MX = ON      //
18 #pragma config PBADEN = OFF    //
19 #pragma config LPT1OSC = OFF   //
20 #pragma config MCLRE = ON      //
21 #pragma config LVP = OFF       //
22
23 #include <xc.h>
24 #define _XTAL_FREQ 48000000UL
25
26 void configure(void){
27     // Aquí colocas las configuraciones
28     TRISD = 0xFC; // 
29 }
30
31 void main(void) {
32     configure();
33     while(1){
34         if(PORTBbits.RB0 == 1){
35             LATD = 0x01;
36             __delay_ms(250);
37             LATD = 0x02;
38             __delay_ms(250);
39         }
40         else{
41             LATD = 0x00;
42         }
43     }
44 }

```

12

Ejemplo: Señal de cruce de tren con entrada de activación en interrupción por cambio de valor en RB4:RB7

```

9  #pragma config PLLDIV = 1      // 
10 #pragma config CPUDIV = OSC1_PLL2/
11 #pragma config FOSC = XTPLL_XT // 
12 #pragma config PWRT = ON       // 
13 #pragma config BOR = OFF      // 
14 #pragma config BORV = 3        // 
15 #pragma config WDT = OFF      // 
16 #pragma config WDTPS = 32768   // 
17 #pragma config CCP2MX = ON     // 
18 #pragma config PBADEN = OFF    // 
19 #pragma config LPT1OSC = OFF   // 
20 #pragma config MCLRE = ON      // 
21 #pragma config LVF = OFF      // 
22 
23 #include <xc.h>
24 #define _XTAL_FREQ 48000000UL
25 
26 unsigned int estado = 0;
27 unsigned char cochinada = 0;
28 
29 void configure(void){
30     //Aquí colocas las configuraciones
31     TRISD = 0xFC;           //
32     INTCON = 0x80;          //
33 }
34 
35 void main(void) {
36     configure();
37     while(1){
38         if(estado == 1){
39             LATD = 0x01;
40             __delay_ms(250);
41             LATD = 0x02;
42             __delay_ms(250);
43         }
44         else{
45             LATD = 0x00;
46         }
47     }
48 }
49 
50 void __interrupt(high_priority) RB_Isr(void){
51     cochinada = PORTB;      //Para quitar la ;
52     if(estado == 1){
53         estado = 0;
54     }
55     else{
56         estado = 1;
57     }
58     INTCONbits.RBIF = 0;
59 }

```

13

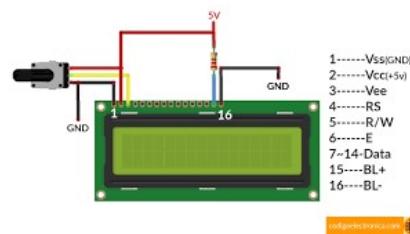
Agenda:

- El display alfanumérico LCD 2x16 HD44780, llamada de librerías en XC8
- El conversor A/D del microcontrolador PIC18F4550

14

El LCD alfanumérico HD44780

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



15

El LCD alfanumérico HD44780

- ROM de caracteres:
 - Muy similar al código ASCII en 7 bits
 - El símbolo de grado (°) en ASCII es Alt+0167, en el ROM de caracteres del HD44780 es 0xDF
 - El símbolo “ñ” en ASCII es Alt+164, en el ROM de caracteres del HD44780 es 0xEE
 - Capacidad de ocho caracteres personalizados (CGRAM 0x00-0x07)

Upper Row	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Lower Row	0x000000	0x000001	0x000010	0x000011												
	0	Q	P	^	P				-	タ	ミ	ア	。	。	。	。
(1)	!	1	A	Q	a	q			ア	チ	ム	エ	。	。	。	。
(2)	"	2	B	R	b	r			イ	ツ	メ	ゼ	。	。	。	。
(3)	#	3	C	S	c	s			ウ	テ	エ	ス	。	。	。	。
(4)	\$	4	D	T	d	t			エ	ト	ト	ム	。	。	。	。
(5)	%	5	E	U	u	e			オ	ナ	ニ	シ	。	。	。	。
(6)	&	6	F	V	f	v			カ	ニ	ヨ	ロ	Σ	Σ	Σ	Σ
(7)	'	7	G	W	g	w			ア	キ	フ	ラ	π	π	π	π
(8)	(8	H	X	h	x			イ	ク	ネ	リ	χ	χ	χ	χ
(1))	9	I	Y	i	y			タ	ル	ル	ル	γ	γ	γ	γ
(2)	*	:	J	Z	j	z			エ	コ	ハ	レ	じ	じ	じ	じ
(3)	+	;	K	L	k	l			オ	サ	ヒ	ロ	*	*	*	*
(4)	,	<	L	¥	1	l			シ	フ	ワ	フ	四	四	四	四
(5)	-	=	M]	m	}			ユ	ス	ヘ	ン	モ	モ	モ	モ
(6)	.	>	N	^	n	+			ヨ	セ	ホ	。	。	。	。	。
(7)	/	?	O	_	o	€			ウ	ソ	マ	ロ	ö	ö	ö	ö
(8)																

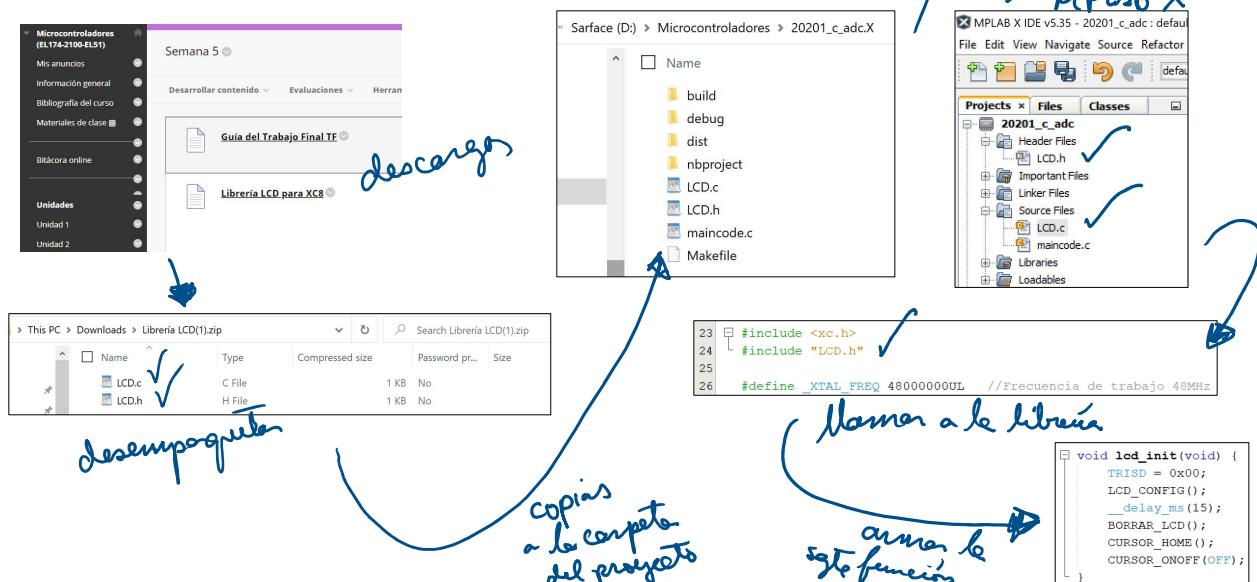
16

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 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
 - Tener en cuenta FOSC especificado dentro de la librería (48MHz)

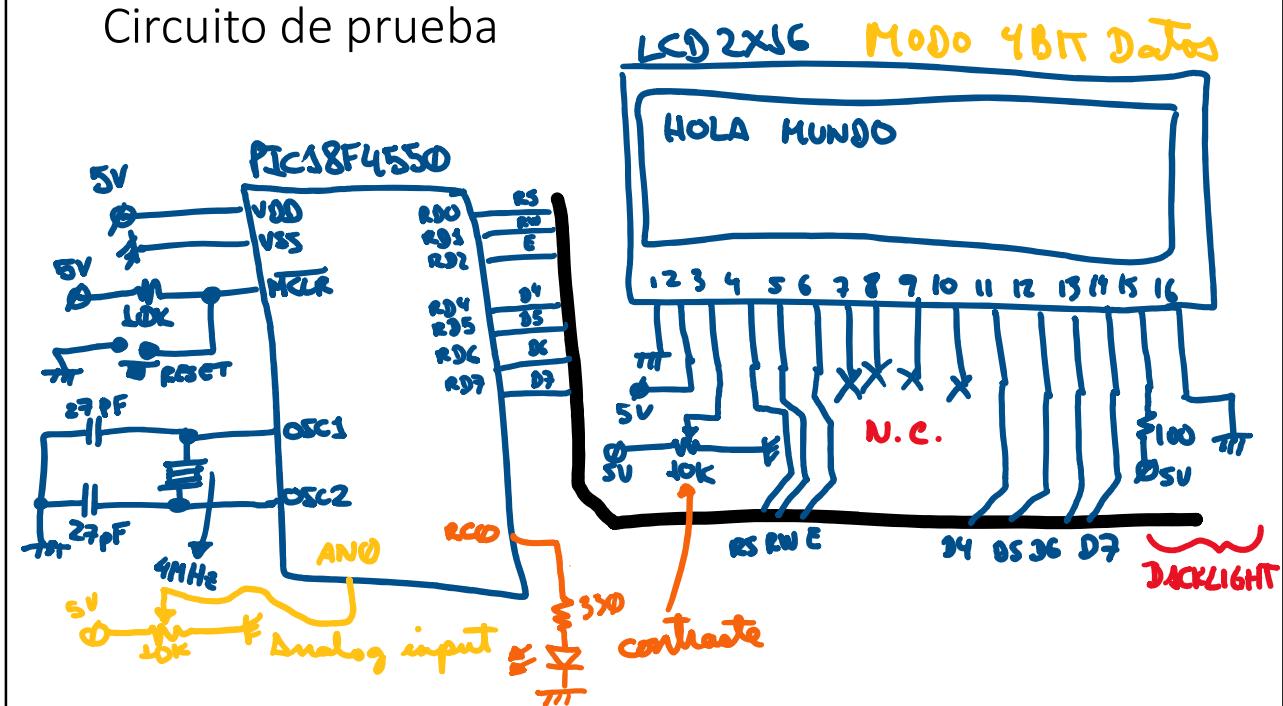
17

Uso del LCD en XC8 (librería S_SAL)



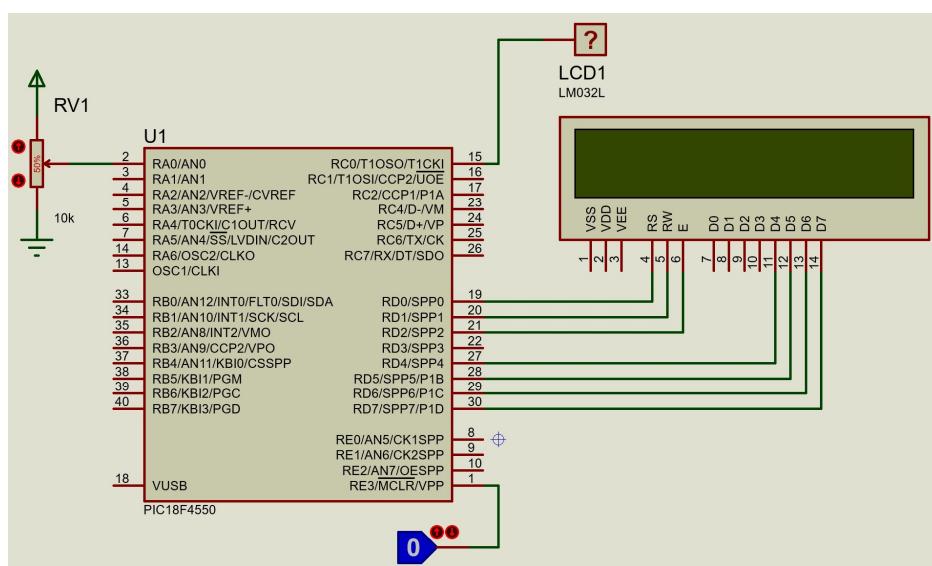
18

Circuito de prueba



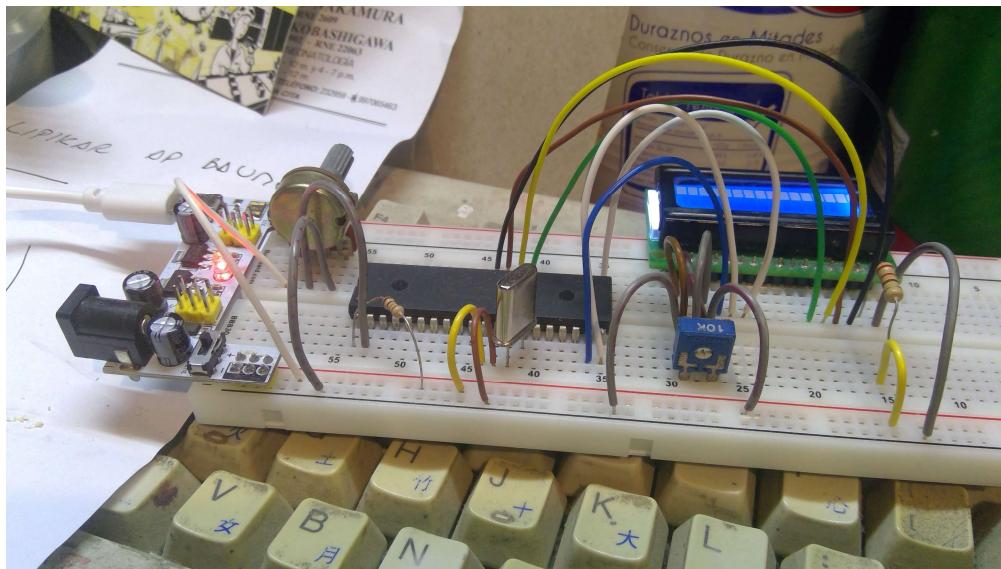
19

Circuito simulado en Proteus



20

Implementación física



21

Ejemplo de aplicación: Hola mundo con temperatura manual y carácter especial de grado

```

1  #pragma config PLLDIV = 1           // PLL Presc
2  #pragma config CPUDIV = OSC1_PLL2 // System C
3  #pragma config FOSC = XTPLL_XT // Oscillatc
4  #pragma config PWRT = ON          // Power-up
5  #pragma config BOR = OFF          // Brown-out
6  #pragma config WDT = OFF          // Watchdog
7  #pragma config CCP2MX = ON         // CCP2 MUX
8  #pragma config PBADEN = OFF        // PORTB A/I
9  #pragma config MCLE = ON          // MCLR Pin
10 #pragma config LVP = OFF          // Single-Su
11
12 #include <xc.h>
13 #include "LCD.h"
14 #define _XTAL_FREQ 48000000UL      //frecue
15
16 void lcd_init(void){
17     TRISD = 0x00;
18     LCD_CONFIG();
19     _delay_ms(15);
20     Borrar_LCD();
21     CURSOR_HOME();
22     CURSOR_ONOFF(OFF);
23 }
24
25 void main(void){
26     lcd_init();
27     ESCRIBE_MENSAJE("Hola mundo", 10);
28     POS_CURSOR(2,0);
29     ESCRIBE_MENSAJE("Temperatura:25", 14);
30     ENVIA_CHAR(0xDF);
31     ENVIA_CHAR('C');
32     while(1){
33     }
34 }
35

```

22

Ejemplo de aplicación: Hola mundo con contador 000-350 (con conversión para obtener dígitos de una variable

```

1 #pragma config PLLDIV = 1           // PLL Presca
2 #pragma config CPUDIV = OSC1_PLL2// System Cl
3 #pragma config FOSC = XTPLL_XT   // Oscillator
4 #pragma config PWRT = ON          // Power-up T
5 #pragma config BOR = OFF          // Brown-out
6 #pragma config WDT = OFF          // Watchdog T
7 #pragma config CCP2MX = ON         // CCP2 MUX b
8 #pragma config PBADEN = OFF        // PORTB A/D
9 #pragma config MCLE = ON          // MCLR Pin E
10 #pragma config IVE = OFF          // Single-Sup
11 #include <xc.h>
12 #include "LCD.h"
13 #define _XTAL_FREQ 48000000UL
14
15 //Declaracion de variables globales
16 unsigned int d_millar = 0;
17 unsigned int millar = 0;
18 unsigned int centena = 0;
19 unsigned int decena = 0;
20 unsigned int unidad = 0;
21 unsigned int cuenta = 0;
22
23
24 void lcd_init(void){
25     TRISD = 0x00;
26     LCD_CONFIG();
27     __delay_ms(15);
28     BORRAR_LCD();
29     CURSOR_HOME();
30     CURSOR_ONOFF(OFF);
31 }
32
33 void convierte(unsigned int numero){
34     d_millar = numero / 10000;
35     millar = (numero % 10000) / 1000;
36     centena = (numero % 1000) / 100;
37     decena = (numero % 100) / 10;
38     unidad = numero % 10;
39 }
40
41 void main(void){
42     lcd_init();
43     ESCRIBE_MENSAJE("Hola mundo", 10);
44     POS_CURSOR(2,0);
45     ESCRIBE_MENSAJE("Cuenta:", 7);
46     while(1){
47         convierte(cuenta);
48         POS_CURSOR(2,7);
49         ENVIA_CHAR(decena+0x30);
50         ENVIA_CHAR(unidad+0x30);
51         __delay_ms(50);
52         if(cuenta == 350){
53             cuenta = 0;
54         }
55         else{
56             cuenta++;
57         }
58     }
59 }
60 }
```

23

Ejemplo de aplicación: Hola mundo y mensaje con desplazamiento de derecha a izquierda

```

1 #pragma config PLLDIV = 1           // PLL Presca
2 #pragma config CPUDIV = OSC1_PLL2// System Cl
3 #pragma config FOSC = XTPLL_XT   // Oscillator
4 #pragma config PWRT = ON          // Power-up T
5 #pragma config BOR = OFF          // Brown-out
6 #pragma config WDT = OFF          // Watchdog T
7 #pragma config CCP2MX = ON         // CCP2 MUX b
8 #pragma config PBADEN = OFF        // PORTB A/D
9 #pragma config MCLE = ON          // MCLR Pin E
10 #pragma config IVE = OFF          // Single-Sup
11 #include <xc.h>
12 #include "LCD.h"
13 #define _XTAL_FREQ 48000000UL      //frecuen
14
15 unsigned char cadenon[] = "And
16
17 void lcd_init(void){
18     TRISD = 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     lcd_init();
28     ESCRIBE_MENSAJE("Hola mundo", 10);
29     POS_CURSOR(2,0);
30     while(1){
31         for(unsigned char i=0;i<57;i++){
32             for(unsigned char j=0;j<16;j++){
33                 ENVIA_CHAR(cadenon[j+i]);
34             }
35             __delay_ms(250);
36             POS_CURSOR(2,0);
37         }
38     }
39 }
```

24

Conversor A/D

Revisar Capítulo 21 de la hoja técnica del PIC18F4550

- 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?

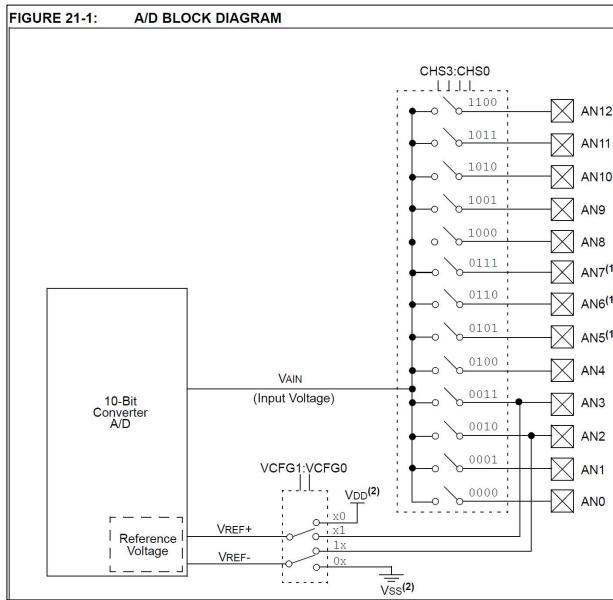
25

Conversor A/D

- Resolución: 10bits (ADRESH:ADRESL)
 - Posee un bit ADFM (justificación del resultado)
- Cantidad de canales analógicos: 13
- **Se lee un canal analógico a la vez**
- ¿Interviene el teorema de muestreo? Si.
- Tiempo de adquisición: se configura en reg. ADCON2
- Rango de voltaje de entrada: 0-5V? (Vref+ = VDD, Vref- = VSS)
- Proceso de adquisición de una señal analógica (ver datasheet)
- ¿Hay interrupciones? Si. ADIE, ADIF

26

Conversor A/D: Diagrama de bloques



27

Registros de configuración para el A/D:

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0							
U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							
bit 0							
Legend:							
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'					
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown				
bit 7-6	Unimplemented: Read as '0'						
bit 5-2	CHS3:CHS0: Analog Channel Select bits						
0000 = Channel 0 (AN0)	0001 = Channel 1 (AN1)	0010 = Channel 2 (AN2)	0011 = Channel 3 (AN3)	0100 = Channel 4 (AN4)	0101 = Channel 5 (AN5) ^(1,2)	0110 = Channel 6 (AN6) ^(1,2)	0111 = Channel 7 (AN7) ^(1,2)
1000 = Channel 8 (AN8)	1001 = Channel 9 (AN9)	1010 = Channel 10 (AN10)	1011 = Channel 11 (AN11)	1100 = Channel 12 (AN12)	1101 = Unimplemented ⁽²⁾	1110 = Unimplemented ⁽²⁾	1111 = Unimplemented ⁽²⁾
bit 1	GO/DONE: A/D Conversion Status bit						
When ADON = 1:	1 = A/D conversion in progress	0 = A/D Idle					
bit 0	ADON: A/D On bit						
1 = A/D converter module is enabled	0 = A/D converter module is disabled						

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1																
U-0	U-0	R/W-0	R/W-0	R/W-0 ⁽¹⁾	R/W ⁽¹⁾	R/W ⁽¹⁾	R/W ⁽¹⁾	bit 7	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
—	—	—	—	—	—	—	—	bit 0								
Legend:																
R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'																
-n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown																
bit 7-6																
Unimplemented: Read as '0'																
bit 5																
VCFG1: Voltage Reference Configuration bit (VREF+ source)																
1 = VREF+ (AN2) 0 = VSS																
bit 4																
VCFG0: Voltage Reference Configuration bit (VREF+ source)																
1 = VREF+ (AN3) 0 = VDD																
bit 3-0																
PCFG3:PCFG0: A/D Port Configuration Control bits:																
PCFG3: PCFG0																
AN12 AN11 AN10 AN9 AN8 AN7 ⁽²⁾ AN6 ⁽²⁾ AN5 ⁽²⁾ AN4 AN3 AN2 AN1 AN0																
A = Analog input D = Digital I/O																

28

Registros de configuración para el A/D:

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0
Legend:							
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'					
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown				
bit 7	ADFM: A/D Result Format Select bit						
	1 = Right justified						
	0 = Left justified						
bit 6	Unimplemented: Read as '0'						
bit 5-3	ACQT2:ACQT0: A/D Acquisition Time Select bits						
	111 = 20 TAD						
	110 = 16 TAD						
	101 = 12 TAD						
	100 = 8 TAD						
	011 = 6 TAD						
	010 = 4 TAD						
	001 = 2 TAD						
	000 = 0 TAD ⁽¹⁾						
bit 2-0	ADCS2:ADCS0: A/D Conversion Clock Select bits						
	111 = FRC (clock derived from A/D RC oscillator) ⁽¹⁾						
	110 = Fosc/64						
	101 = Fosc/16						
	100 = Fosc/4						
	011 = FRC (clock derived from A/D RC oscillator) ⁽¹⁾						
	010 = Fosc/32						
	001 = Fosc/8						
	000 = Fosc/2						

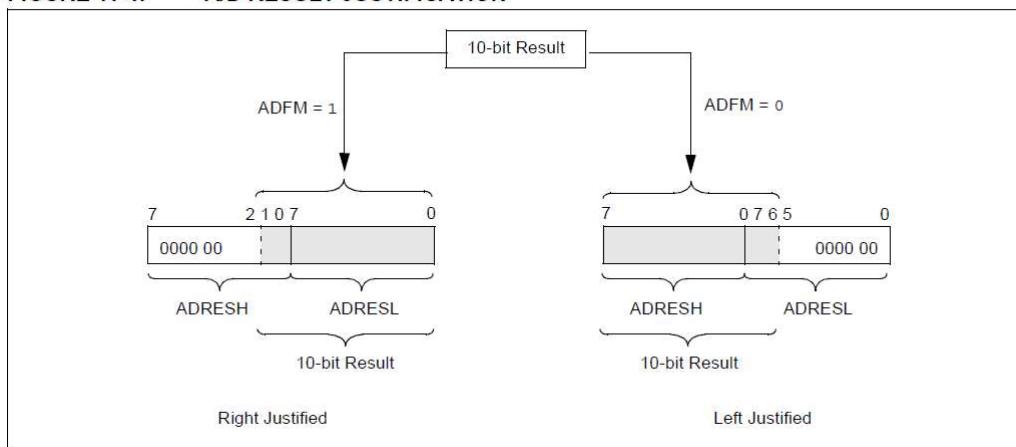
Procedimiento para adquirir una señal analógica:

1. Configure the A/D module:
 - Configure analog pins, voltage reference and digital I/O (ADCON1)
 - Select A/D input channel (ADCON0)
 - Select A/D acquisition time (ADCON2)
 - Select A/D conversion clock (ADCON2)
 - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
 - Clear ADIF bit
 - Set ADIE bit
 - Set GIE bit
3. Wait for the required acquisition time (if required).
4. Start conversion:
 - Set GO/DONE bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
 - Polling for the GO/DONE bit to be cleared
OR
 - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 3 TAD is required before the next acquisition starts.

29

Para obtener el resultado de la conversión A/D:

FIGURE 11-4: A/D RESULT JUSTIFICATION



30

Código ejemplo para configurar y leer AN0 en el A/D:

```

28     unsigned int res_ad = 0;
29
30     void configuracion(void) {
31         //Aqui colocas las configuraciones iniciales
32         ADCON2 = 0xA4;           //ADFM=0 (just derecha), 8TAD, Fosc/4
33         ADCON1 = 0x0E;          //Canal AN0 habilitado
34         ADCON0 = 0x01;          //Canal AN0 seleccionado y encendemos el A/D
35
36     void main(void) {
37         configuracion();
38
39         while (1) {
40             //Tu programa de usuario
41             ADCON0bits.GODONE = 1;           //Inicio una captura de muestra en AN0
42             while(ADCON0bits.GODONE == 1);   //Espero a que termine de convertir
43             res_ad = (ADRESH << 8) + ADRESL; //Grabar el resultado en la variable res_ad
44             convierte(res_ad);            //Sacar los dígitos
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

```

31

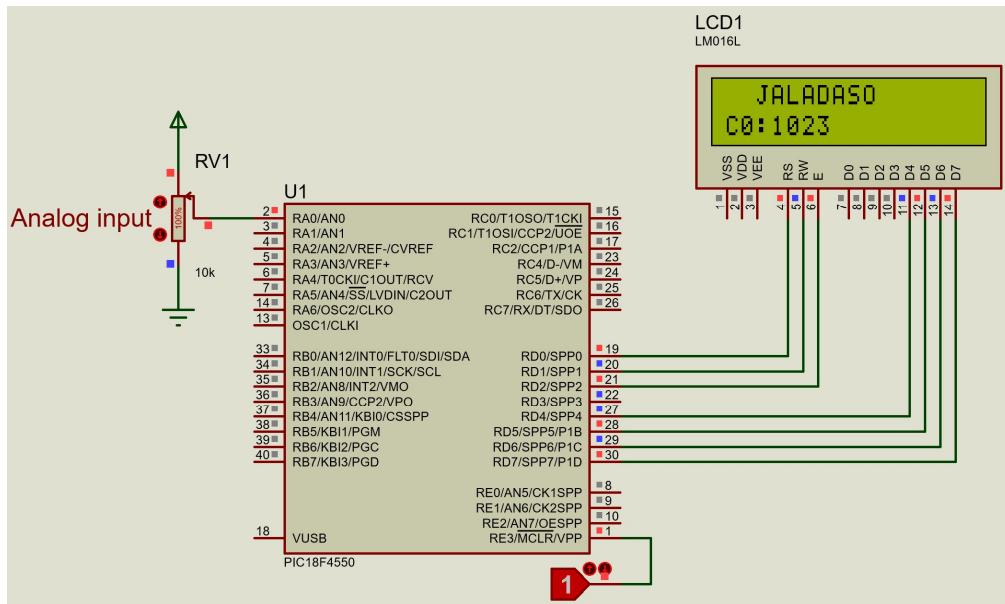
Código en XC8 del ejemplo inicial:

```

26 #define _XTAL_FREQ 48000000UL //Frecu
27
28 unsigned int res_ad = 0;
29 unsigned int millar = 0;
30 unsigned int centena = 0;
31 unsigned int decena = 0;
32 unsigned int unidad = 0;
33
34 void convierte(unsigned int numero){
35     millar = numero /1000;
36     centena = (numero % 1000) / 100;
37     decena = (numero % 100) / 10;
38     unidad = numero % 10;
39 }
40
41 void lcd_init(void) {
42     TRISD = 0x00;           //Puerto D
43     LCD_CONFIG();
44     __delay_ms(15);
45     BORRAR_LCD();
46     CURSOR_HOME();
47     CURSOR_ONOFF(OFF);
48 }
49
50 void configuracion(void) {
51     //Aqui colocas las configuraciones
52     ADCON2 = 0xA4;          //ADE
53     ADCON1 = 0x0E;          //Canal A
54     ADCON0 = 0x01;          //Canal A
55     lcd_init();
56 }
57
58 void main(void) {
59     configuracion();
60     ESCRIBE_MENSAJE("VIRTUAlASO",10);
61     while (1) {
62         //Tu programa de usuario
63         ADCON0bits.GODONE = 1;
64         while(ADCON0bits.GODONE == 1);
65         res_ad = (ADRESH << 8) + ADRESL;
66         convierte(res_ad);
67         POS_CURSOR(2,0);
68         ENVIA_CHAR(millar+0x30);
69         ENVIA_CHAR(centena+0x30);
70         ENVIA_CHAR(decena+0x30);
71         ENVIA_CHAR(unidad+0x30);
72     }
73 }
```

32

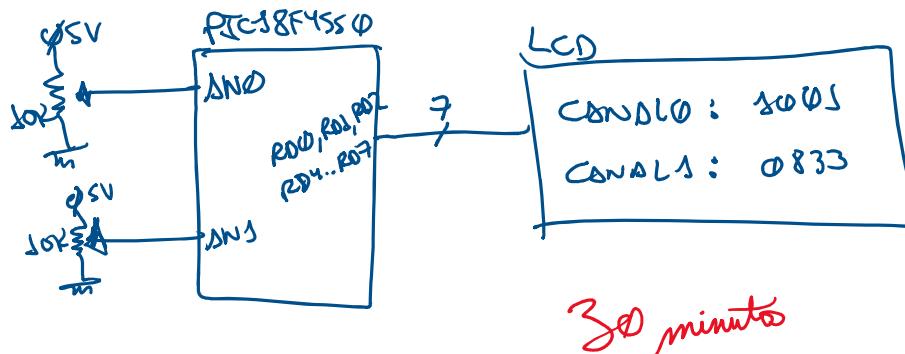
Simulación en Proteus:



33

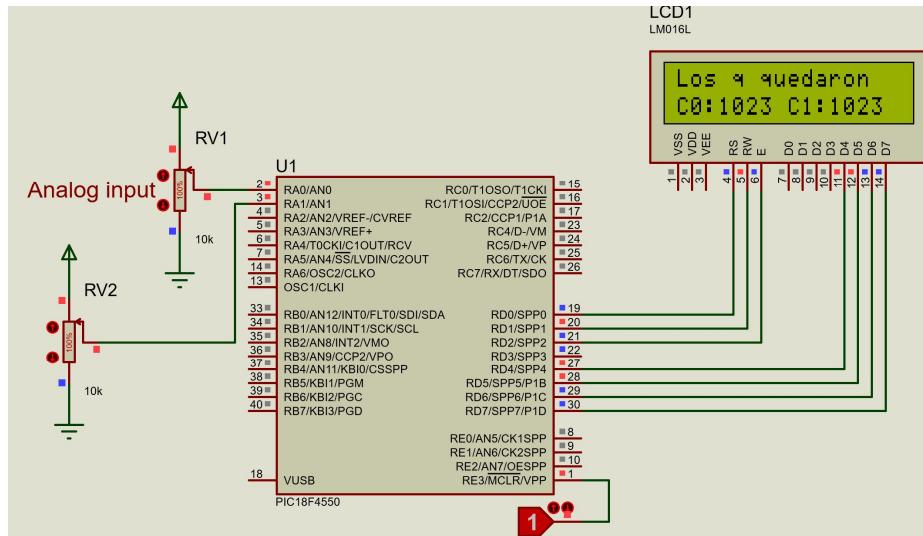
Asignación:

- Leer dos canales analógicos y mostrarlos en el LCD



34

Simulación:



35

Código en XC8:

```

27  #include <xc.h>
28  #include "LCD.h"
29  #define _XTAL_FREQ 48000000UL //Par
30
31  unsigned int res_ad0 = 0;
32  unsigned int res_ad1 = 0;
33
34  unsigned int millar = 0;
35  unsigned int centena = 0;
36  unsigned int decena = 0;
37  unsigned int unidad = 0;
38
39  void convierte(unsigned int numero){
40      millar = numero / 1000;
41      centena = (numero % 1000) / 100;
42      decena = (numero % 100) / 10;
43      unidad = numero % 10;
44  }
45
46  void lcd_init(void) {
47      TRISD = 0x00; //Pue
48      LCD_CONFIG();
49      _delay_ms(15);
50      BORRAR_LCD();
51      CURSOR_HOME();
52      CURSOR_ONOFF(OFF);
53  }
54
55  void configuracion(void) {
56      //Aqui colocas las configuraciones inic
57      ADCON2 = 0xA4; //ADFM=0 (
58      ADCON1 = 0x0D; //Canal AN0
59      ADCONbits.ADON = 1; //Encendemos
60      lcd_init();
61  }
62
63  void main(void) {
64      configuracion(); //Llamada a
65      ESCRIBE_MENSAJE("Los q quedaron",14);
66      while(1){
67          ADCON0 = 0x03; //ADCONbits.GODONE = 1;
68          while(ADCONbits.GODONE == 1); //res_ad0 = (ADRESH << 8) + ADRESL;
69          ADCON0 = 0x07; //res_ad1 = (ADRESH << 8) + ADRESL;
70          while(ADCONbits.GODONE == 1); //POS_CURSOR(2,0); //S
71          ESCRIBE_MENSAJE("00:",3);
72          convierte(res_ad0); //E
73          ENVIA_CHAR(millar+0x30); //E
74          ENVIA_CHAR(centena+0x30); //E
75          ENVIA_CHAR(decena+0x30); //E
76          ENVIA_CHAR(unidad+0x30); //E
77          ESCRIBE_MENSAJE(" Cl:",4);
78          convierte(res_ad1); //E
79          ENVIA_CHAR(millar+0x30); //E
80          ENVIA_CHAR(centena+0x30); //E
81          ENVIA_CHAR(decena+0x30); //E
82          ENVIA_CHAR(unidad+0x30); //E
83      }
84  }
85
86
87
88

```

36

Fin de la sesión!