Taller de Microcontroladores y Placas de Desarrollo

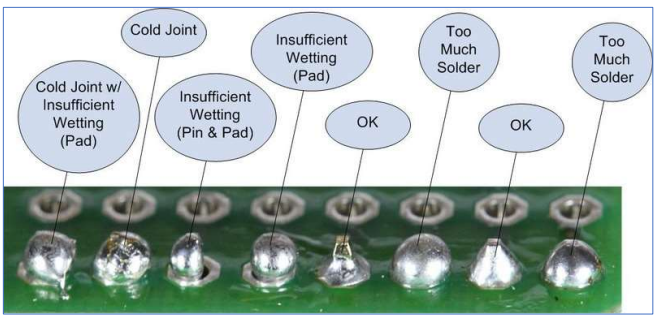
Profesor: Kalun José Lau Gan

Semestre 2023-2

Sesión 4

Preguntas previas:

- ¿Cómo soldar bien?
  - Tener buenas herramientas: cautín con control de temperatura, pinzas, pelacables, desarmadores, **pasta para soldar (flux)**, soldadura de buena calidad, esponja humedecida con agua, alcohol isopropílico.
  - Practicar y desarrollar la habilidad (mirar videos instruccionales)
  - Verificar que lo soldado tenga una forma cónica, si esta en circunferencia significa que es un soldado frío.
- ¿Empresas que desarrollan PCBs?
  - Jlcpcb.com
  - Pcbway.com



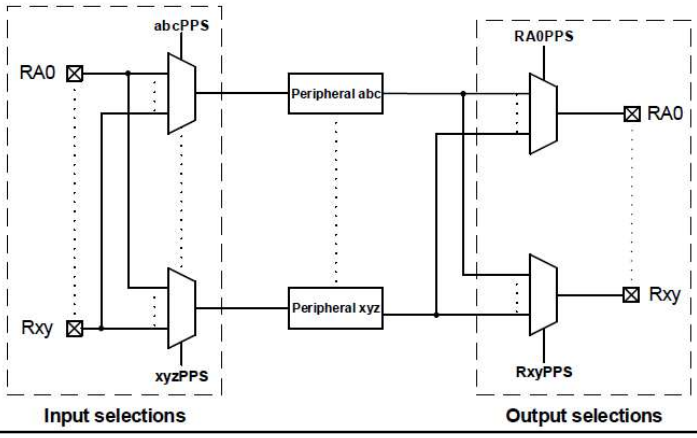
Preguntas previas:

Agenda:

- El PPS en el PIC18F57Q43
- El módulo LCD 16x2 HD44780
- Módulos de temporización en el PIC18F57Q43
  - El Timer0
- El módulo ADC del PIC18F57Q43

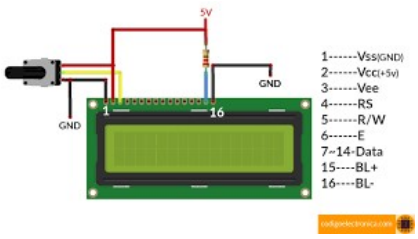
El PPS en el PIC18F57Q43

- 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



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



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)

El LCD alfanumérico HD44780

- Tabla de caracteres ASCII de 7 bits

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

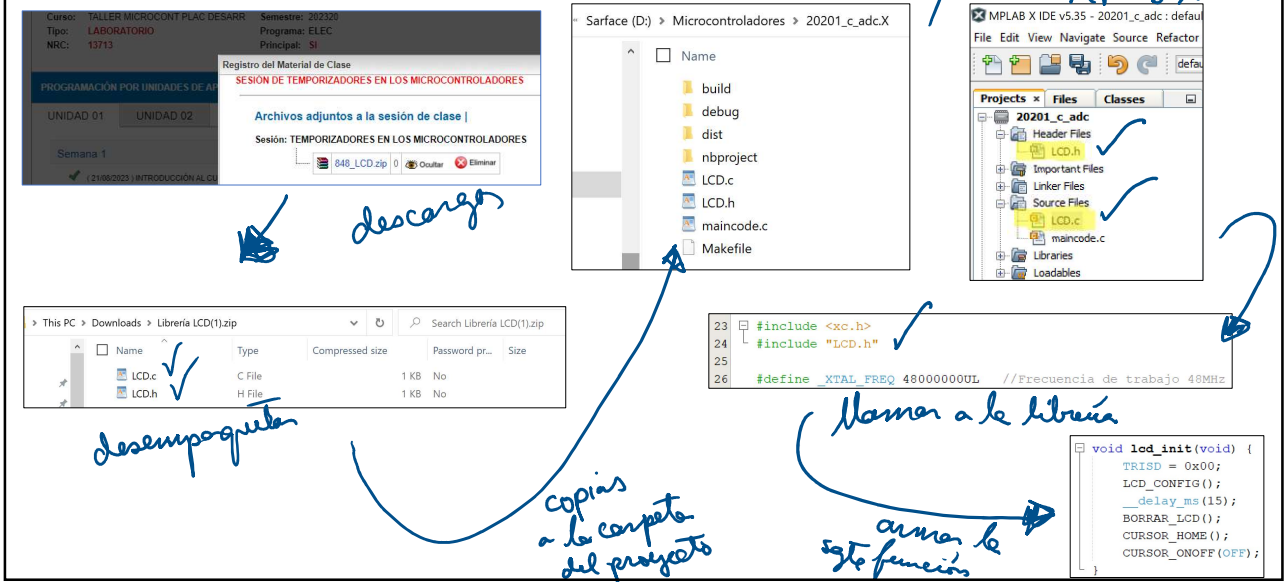
Source: www.LookupTables.com

El LCD alfanumérico HD44780

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

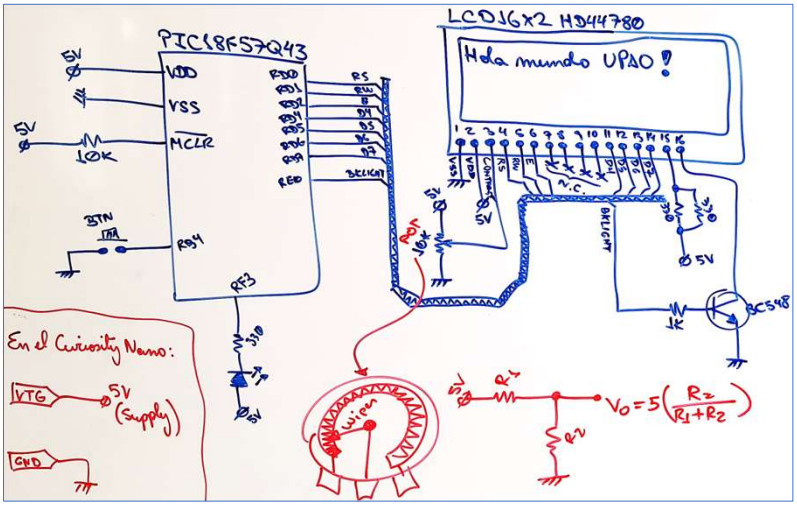
Uso del LCD en XC8 (librería S_SAL)

- Crear primero el Proyecto en el MPLABX



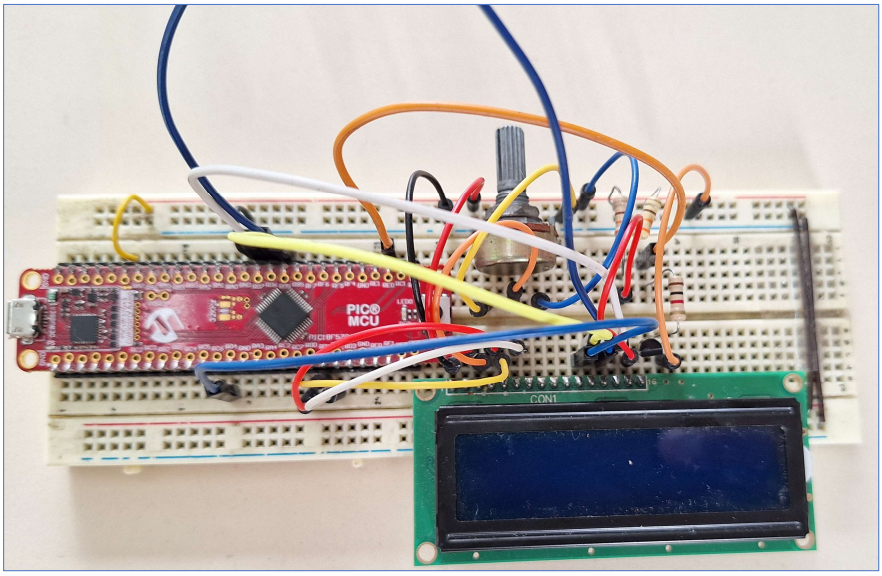
Ejemplo: Visualizar “Hola mundo UPAO!” en el LCD

- Hardware



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

- Hardware



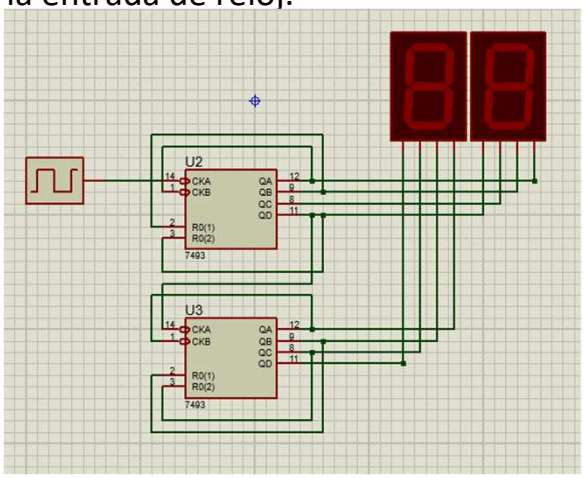
Ejemplo: Visualizar “Hola mundo UPAO!” en el LCD

- Código ejemplo en XC8
- Se esta empleando el botón integrado en el Curiosity para la función de RESET.
- Esta función de RESET esta implementado con la INT0 redirigido al RB4 mediante el PSS y empleando la instrucción en XC8 PIC Assembler “RESET” (reset por software)

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

void configuro(void){
    OSCCON1 = 0x60;      //HFINTOSC selected
    OSCFRQ = 0x02;       //HFINTOSC a 4MHz
    OSCEN = 0x40;        //HFINTOSC enabled
    TRISEbits.TRISE0 = 0;   //RE0 como salida
    ANSELEbits.ANSELE0 = 0; //RE0 como digital
    LATEbits.LATE0 = 1;     //LCD backlight ON
    //Para usar el boton integrado como funcion de RESET al módulo
    TRISBbits.TRISB4 = 1;   //RB4 como entrada
    ANSELBbits.ANSELB4 = 0; //RB4 como digital
    WPUBbits.WPUB4 = 1;     //RB4 con pullup activada
    INT0PPS = 0x0C;         //Redireccion de INT0 hacia RB4
    INTCON0 = 0x80;         //GIE=1, INT0EDG=0
    PIE1 = 0x01;            //INT0IE=1
}

void main(void) {
    configuro();            //llamado a funcion configuro
    LCD_INIT();             //Inicializacion del LCD
    POS_CURSOR(1,0);        //Posicion del curso arriba a la izquierda
    ESCRIBE_MENSAJE2("Hola mundo UPAO!");
    while(1){
        unsigned char x_var;
        for(x_var=0;x_var<16;x_var++){
            POS_CURSOR(2,x_var);
            ENVIA_CHAR(0xDB);
            __delay_ms(100);
        }
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE2("                ");
    }
}

//Funcion de RESET usando interrupcion externa INT0 e instruccion assembler
void __interrupt(irq(IRQ_INT0)) INT0_ISR(void){
    PIR1bits.INT0IF = 0;
    asm("RESET");
}
```

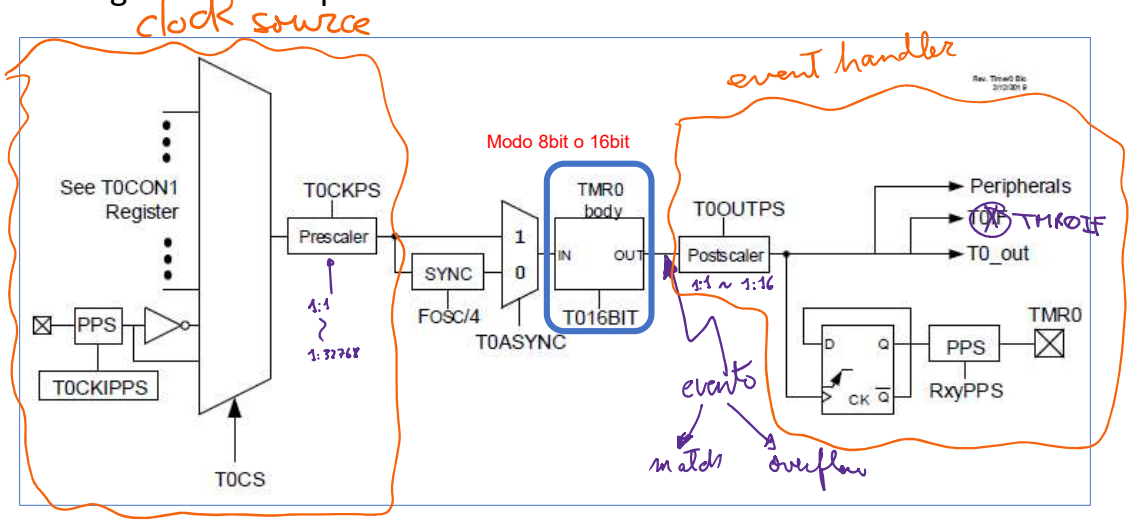

La previa: contadores digitales

- Sistemas secuenciales, poseen reloj, un registro de cuenta, dependiendo del su función, puede incrementar o puede decrementar la cuenta según la entrada de reloj.



El módulo Timer 0:

- Diagrama de bloques



15

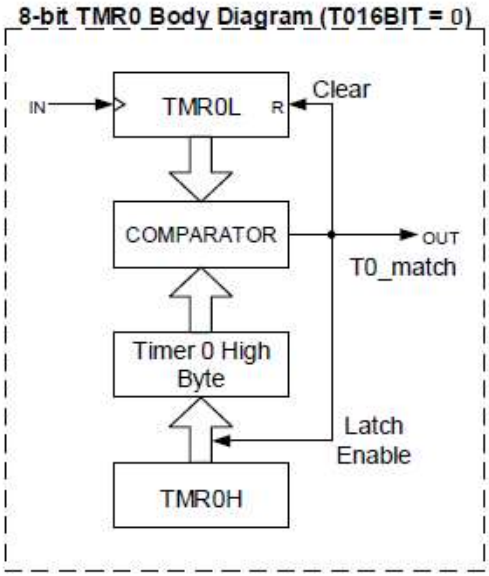
El módulo Timer 0

- (Ref. Item 24 de la hoja técnica del microcontrolador PIC18F57Q43)
- Temporizador de cuenta ascendente
- Resolución 8 bits (0-254) ó 16 bits (0-65535)
- Las cuentas del Timer0 se alojan en:
  - TMR0H:TMR0L (16 bits)
  - TMR0L (8 bits)
- Diversas fuentes de reloj (revisar T0CON1 y T0CS)
- Divisor de frecuencia al reloj de entrada PRESCALER (1:1 – 1:32768)
- POSTSCALER de 1:1 a 1:16 (incrementos de uno en uno)
- Al activarse TMR0IF=1 ó T0IF=1 se debe de bajar manualmente la bandera para que se pueda detectar un nuevo desborde ó evento de match (simplemente haciendo “bcf PIR3, 7”; siendo el bit 7 el TMR0IF).
- En un evento de match o al desbordarse puede emitir interrupción al CPU si TMR0IE=1), revisar capítulo 11 de la hoja técnica.

16

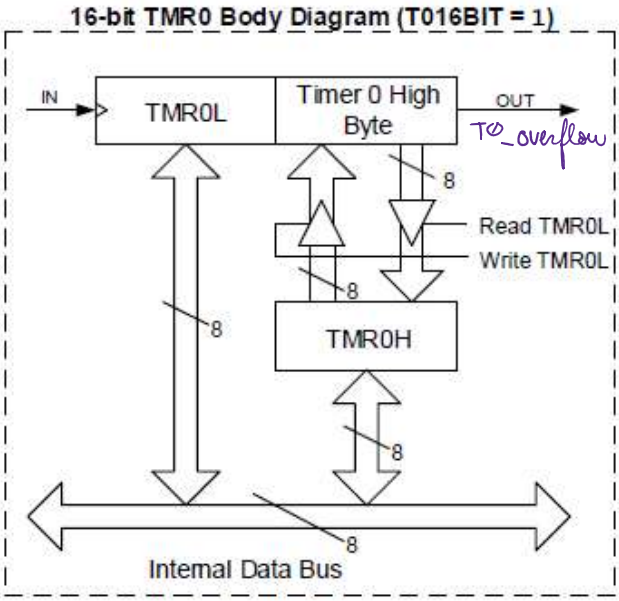
El módulo Timer 0:

- Modo de trabajo en 8 bits:
  - Mejorado con respecto al Timer0 presente en el PIC18F4550 ó PIC18F45K50
  - El TMR0L se usa como el registro de cuenta
  - El TMR0H se utiliza como valor de comparación
  - Cuenta máxima es cuando TMR0H es 255, haciendo que se mande a cero la cuenta actual, quiere decir que tendrás un rango de 0-254
  - Cuando TMR0L es igual a TMR0H se produce un evento de "match" el cual limpia la cuenta y actualiza el valor de comparación.
  - Nunca se desborda



El módulo Timer 0:

- Modo de trabajo en 16 bits:
- Tener en consideración el procedimiento estricto sobre el proceso de carga de un valor en la cuenta en modo 16 bits: Primero cargar en TMR0H y luego en TMR0L.
- El evento de desborde se produce cuando la cuenta esta en el valor mas alto (65535) y se recibe un pulso de reloj, ocasionando que la cuenta pase a 0 y levantándose la bandera de desborde (TMR0IF=1 ó T0IF=1) siempre y cuando POSTSCALER 1:1



El módulo Timer 0:

- Temporización máxima si HFINTOSC = 4MHz
  - Empleando Opción FOSC/4 = 1MHz (1µs) como fuente de reloj al TMR0
  - Prescaler = 1:32768 (32768 µs por cuenta)
  - Modo = 16bits (65536 cuentas)
  - En el evento de desborde: 2147483648 µs (35 minutos)
  - Postscaler = 1:16
  - Nos sale al final 34,359,738,368 µs de temporización máxima! (aprox 9.5 horas)

El módulo Timer 0

- Registros implicados en la operación del Timer0:
  - Registros de cuenta
    - TMR0H:TMR0L (16bits)
    - TMR0H es valor de comparación y TMR0L es el registro de cuenta (8bits)
  - Registros de configuración T0CON0 y T0CON1
  - Registros PIE3 (habilitadores), PIR3 (banderas) y IPR3 (prioridades) ubicando en la VIC (vectored interrupt controller cap 11)

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0318	TMR0L	7:0	TMR0L[7:0]							
0x0319	TMR0H	7:0	TMR0H[7:0]							
0x031A	T0CON0	7:0	EN		OUT	MD16	OUTPS[3:0]			
0x031B	T0CON1	7:0	CS[2:0]			ASYNC	CKPS[3:0]			
0x04A1	PIE3	7:0	TMR0IE	CCP1IE	TMR1GIE	TMR1IE	TMR2IE	SPI1IE	SPI1TXIE	SPI1RXIE
0x04B1	PIR3	7:0	TMR0IF	CCP1IF	TMR1GIF	TMR1IF	TMR2IF	SPI1IF	SPI1TXIF	SPI1RXIF
0x0365	IPR3	7:0	TMR0IP	CCP1IP	TMR1GIP	TMR1IP	TMR2IP	SPI1IP	SPI1TXIP	SPI1RXIP

El módulo Timer 0

- Registro T0CON0:
  - Habilitador del módulo
  - Señal OUT
  - Modo (8 ó 16 bits)
  - Postscaler

Name: T0CON0
Address: 0x31A

Timer0 Control Register 0

Bit	7	6	5	4	3	2	1	0
	EN		OUT	MD16	OUTPS[3:0]			
Access	R/W		R	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit 7 – EN TMR0 Enable

Value	Description
1	The module is enabled and operating
0	The module is disabled

Bit 5 – OUT TMR0 Output

Bit 4 – MD16 16-Bit Timer Operation Select

Value	Description
1	TMR0 is a 16-bit timer
0	TMR0 is an 8-bit timer

Bits 3:0 – OUTPS[3:0] TMR0 Output Postscaler (Divider) Select

Value	Description
1111	1:16 Postscaler
1110	1:15 Postscaler
1101	1:14 Postscaler
1100	1:13 Postscaler
1011	1:12 Postscaler
1010	1:11 Postscaler
1001	1:10 Postscaler
1000	1:9 Postscaler
0111	1:8 Postscaler
0110	1:7 Postscaler
0101	1:6 Postscaler
0100	1:5 Postscaler
0011	1:4 Postscaler
0010	1:3 Postscaler
0001	1:2 Postscaler
0000	1:1 Postscaler

El módulo Timer 0

- Registro T0CON1:
  - Fuente de reloj
  - Sincronismo
  - Prescaler

Name: T0CON1
Address: 0x31B

Timer0 Control Register 1

Bit	7	6	5	4	3	2	1	0
	CS[2:0]			ASYNC	CKPS[3:0]			
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 7:5 – CS[2:0] Timer0 Clock Source Select

Value	Description
111	CLC1_OUT
110	SOSC
101	MFINTOSC (500 kHz)
100	LFINTOSC
011	HFINTOSC
010	$F_{OSC}/4$
001	Pin selected by T0CKIPPS (Inverted)
000	Pin selected by T0CKIPPS (Non-inverted)

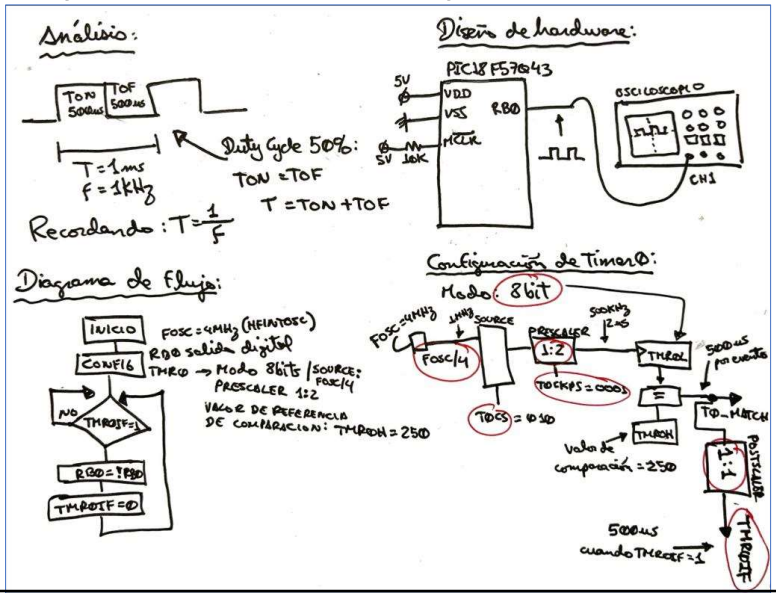
Bit 4 – ASYNC TMR0 Input Asynchronization Enable

Value	Description
1	The input to the TMR0 counter is not synchronized to system clocks
0	The input to the TMR0 counter is synchronized to Fosc/4

Bits 3:0 – CKPS[3:0] Prescaler Rate Select

Value	Description
1111	1:32768
1110	1:16384
1101	1:8192
1100	1:4096
1011	1:2048
1010	1:1024
1001	1:512
1000	1:256
0111	1:128
0110	1:64
0101	1:32
0100	1:16
0011	1:8
0010	1:4
0001	1:2
0000	1:1

Ejemplo: Emisión de una señal cuadrada de 1KHz y 50% de DC empleando el Timer0



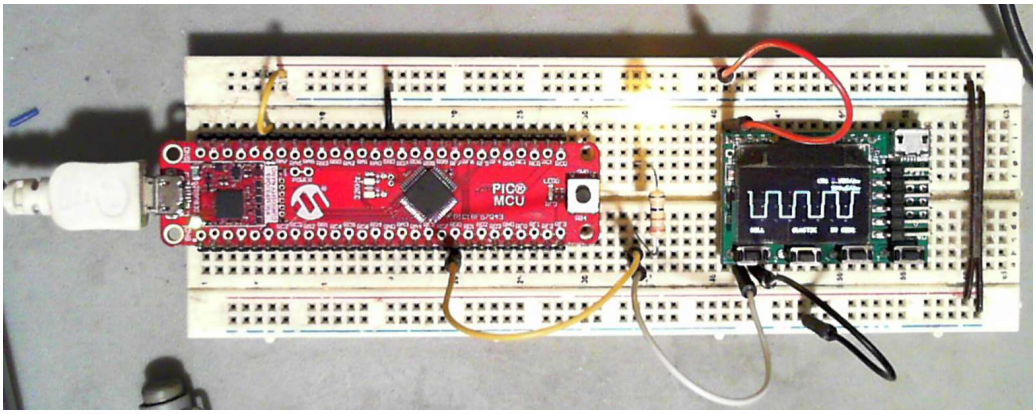
Ejemplo: Emisión de una señal cuadrada de 1KHz y 50% de DC empleando el Timer0

- Código fuente en XC8

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

void configuro(void){
    OSCCON1 = 0x60;                 //HFINTOSC y Divisor en 1:1
    OSCFRQ = 0x02;                  //HFINTOSC a 4MHz
    OSCEN = 0x40;                   //HTINTOSC habilitado
    TRISBbits.TRISB0 = 0;           //RB0 como salida
    ANSELBbits.ANSELB0 = 0;         //RB0 como digital
    T0CON0 = 0x80;                  //Tmr0 ON, postscaler 1:1
    T0CON1 = 0x41;                  //Source FOSC/4, prescaler 1:2
    TMR0H = 250;                    //Valor de referencia de comparacion
}

void main(void) {
    configuro();                            //llama a función configuro
    while(1){                               //bucle repetitivo
        while(PIR3bits.TMR0IF == 0);        //pregunto si TMR0IF=1
        LATDbits.LATD0 = ~LATDbits.LATD0;   //basculo RB0
        PIR3bits.TMR0IF = 0;                //bajo banderda TMR0IF
    }
}
```

Ejemplo: Emisión de una señal cuadrada de 1KHz y 50% de DC empleando el Timer0

- Validación del ejemplo en hardware



25

Ejemplo: Emisión de una señal cuadrada de 1KHz y 50% de DC empleando el Timer0

- Añadiendo el LCD para visualizar mensajes al ejemplo anterior:

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

void configuro(void){
    OSCCON1 = 0x60;                 //HFINTOSC y Divisor en 1:1
    OSCFRQ = 0x02;                  //HFINTOSC a 4MHz
    OSCEN = 0x40;                   //HTINTOSC habilitado
    TRISBbits.TRISB0 = 0;           //RB0 como salida
    ANSELBbits.ANSELB0 = 0;         //RB0 como digital
    T0CON0 = 0x80;                  //Tmr0 ON, postscaler 1:1
    T0CON1 = 0x41;                  //Source FOSC/4, prescaler 1:2
    TMR0H = 250;                    //Valor de referencia de comparacion
    LCD_INIT();                     //inicializacion del LCD
}

void main(void) {
    configuro();                                //llama a función configuro
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE2("Generador SQRWAV");
    while(1){                                   //bucle repetitivo
        POS_CURSOR(2,0);
        ESCRIBE_MENSAJE2("f:1.00KHz DC:50%");
        while(PIR3bits.TMR0IF == 0);            //pregunto si TMR0IF=1
        LATDbits.LATD0 = ~LATDbits.LATD0;       //basculo RB0
        PIR3bits.TMR0IF = 0;                    //bajo banderda TMR0IF
    }
}
```

26

Ejemplo: Emisión de una señal cuadrada de 1KHz con 20% y 70% de DC

- Se tiene que calcular cuatro temporizaciones en el Timer0 (200µs, 800µs, 700µs y 300µs)
- Tener en cuenta que TMR0H solo acepta valores en un rango de 8 bits (0-255) por lo que el PRESCALER se modificó a 1:4
- Se esta empleando el pulsador en RB4 integrado en el Curiosity Nano para seleccionar entre 20% y 70% de duty cycle de la señal de 1KHz en RB0

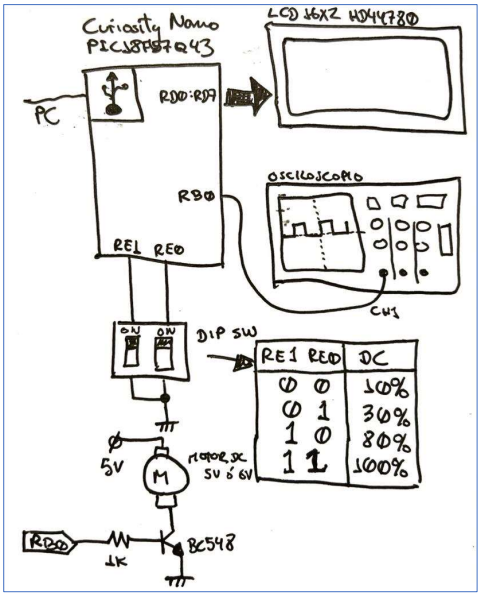
```c
#include <xc.h>
#include "cabecera.h"
#include "LCD.h"
#define _XTAL_FREQ 4000000UL

void configuro(void){
    OSCCON1 = 0x60;              //HFINTOSC y Divisor en 1:1
    OSCFRQ = 0x02;               //HFINTOSC a 4MHz
    OSCEN = 0x40;                //HTINTOSC habilitado
    TRISBbits.TRISB0 = 0;        //RB0 como salida
    ANSELBbits.ANSELB0 = 0;      //RB0 como digital
    TRISBbits.TRISB4 = 1;        //RB4 como entrada
    ANSELBbits.ANSELB4 = 0;      //RB4 como digital
    WPUBbits.WPUB4 = 1;          //RB$ con pullup activado
    T0CON0 = 0x80;               //Tmr0 ON, postscaler 1:1
    T0CON1 = 0x42;               //Source FOSC/4, prescaler 1:4
    TMR0H = 250;                 //Valor de referencia de comparacion
    LCD_INIT();                  //inicializacion del LCD
}
```

```c
void main(void) {
    configuro();                          //llama a función configuro
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE2("Generador SQRWAV");
    while(1){                             //bucle repetitivo
        if(PORTBbits.RB4 == 0){           //pregunto si presione RB4
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE2("f:1.00KHz DC:20%");
            LATBbits.LATB0 = 1;           //RB4 en uno (TON)
            TMR0H = 50;                  //TMR0 temporiza 200uS
            while(PIR3bits.TMR0IF == 0);  //pregunto si terminó de temporizar (TMR0IF=1)
            PIR3bits.TMR0IF = 0;          //bajo banderda TMR0IF
            LATBbits.LATB0 = 0;           //RB4 en cero (TOF)
            TMR0H = 200;                  //TMR0 temporiza 800uS
            while(PIR3bits.TMR0IF == 0);  //pregunto si terminó de temporizar (TMR0IF=1)
            PIR3bits.TMR0IF = 0;          //bajo banderda TMR0IF
        }
        else{
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE2("f:1.00KHz DC:70%");
            LATBbits.LATB0 = 1;           //RB4 en uno (TON)
            TMR0H = 175;                  //TMR0 temporiza 700uS
            while(PIR3bits.TMR0IF == 0);  //pregunto si terminó de temporizar (TMR0IF=1)
            PIR3bits.TMR0IF = 0;          //bajo banderda TMR0IF
            LATBbits.LATB0 = 0;           //RB4 en cero (TOF)
            TMR0H = 75;                  //TMR0 temporiza 300uS
            while(PIR3bits.TMR0IF == 0);  //pregunto si terminó de temporizar (TMR0IF=1)
            PIR3bits.TMR0IF = 0;          //bajo banderda TMR0IF
        }
    }
}
```

- Verificar en osciloscopio el cambio de duty cycle en RB0 al pulsar botón en RB4
- Colocar un LED en RB0 para verificar si cambia su intensidad al pulsar botón en RB4

27

Asignación:



- Tomando referencia los ejemplos anteriores, desarrollar un driver para un motor DC de 5V-6V empleando PWM.
- La señal PWM será de 5KHz con DC variable mediante la combinatoria de dos entradas (RE1 y RE0) y según la tabla mostrada.
- Validar la señal obtenida empleando un osciloscopio.
- Verificar la corriente de consumo del motor ya que si excede de 100mA deberán de cambiar el transistor a uno que soporte mayor corriente, por ejemplo el BD135.

28

Fin de la sesión

- Realizar los ejercicios propuestos pendientes siguiendo el workflow