Microcontroladores
Semana 6

Semestre 2021-0

Profesor: Kalun Lau

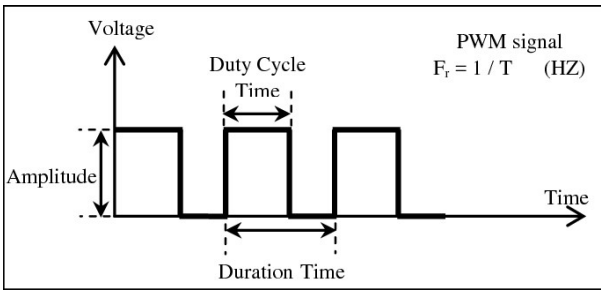
Preguntas previas

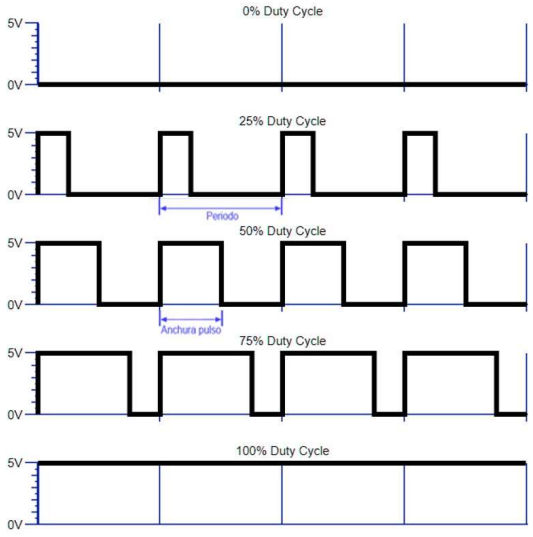
- ¿Problemas para acceder a la carpeta compartida de TF?

Agenda

- PWM con CCP
- Comunicación serial
  - I2C
  - Serial síncrona
  - Serial asíncrona
  - Otros protocolos e interfaces

PWM : Modulación de ancho de pulso



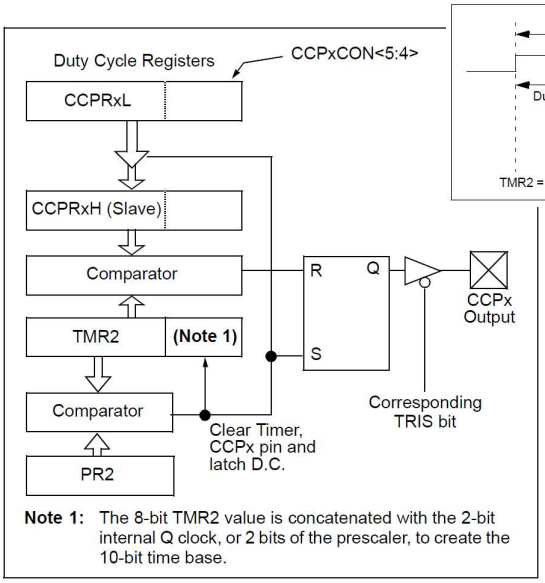


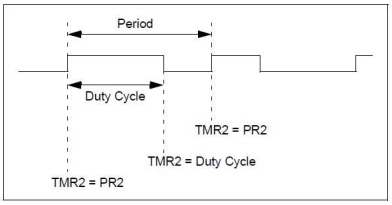
- Aumento de eficiencia en diferentes aplicaciones de circuitos electrónicos.
- Aplicaciones:
  - Arranque y control de velocidad de motores eléctricos.
  - Comunicaciones digitales
  - Amplificadores de audio!
  - Control de intensidad luminosa en lámparas LED
  - Cargadores de celular y demás fuentes de energía conmutadas
  - Largo etc

Fuente lineal vs fuente conmutada

5

PWM con el CCP





1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPRxL register and CCPxCON<5:4> bits.
3. Make the CCPx pin an output by clearing the appropriate TRIS bit.
4. Set the TMR2 prescale value, then enable Timer2 by writing to T2CON.
5. Configure the CCPx module for PWM operation.

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot Tosc \cdot (\text{TMR2 Prescale Value})$$

$$\text{PWM Duty Cycle} = (CCPRxL{:}CCPxCON\langle 5{:}4\rangle) \cdot Tosc \cdot (\text{TMR2 Prescale Value})$$

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{Fosc}{FPWM}\right)}{\log(2)} \text{ bits}$$

6

Generar 1KHz 50% con el CCP en modo PWM

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot Tosc \cdot (\text{TMR2 Prescale Value})$$

- ¿PR2 si Fosc=48MHz?
  - PR2=749 => inválido
- ¿PR2 si Fosc=16MHz?
  - PR2=249 => válido
- Valor de CCPR2L para 50% DC?
  - 125
- TRISCbits.RC1 = 0; //poner el puerto de PWM como salida
- T2CON = 0x06
- CCP2CON = 0x0C

$\frac{1}{1KHz} = [(PR2)+1] \cdot 4 \cdot \left(\frac{1}{48MHz}\right) \cdot (16)$

PR2 = 749

T2CON:

REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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 **Unimplemented:** Read as '0'

bit 6-3 **T2OUTPS3:T2OUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

T2CON = 0x06

CCP2CON:

REGISTER 15-1: CCPxCON: STANDARD CCPx CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—(1)	—(1)	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
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'(1)

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle Bit 1 and Bit 0 for CCPx Module

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Module Mode Select bits

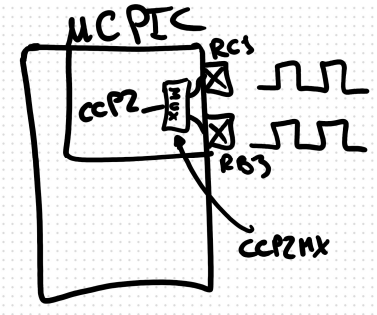
0000 = Capture/Compare/PWM disabled (resets CCPx module)
0001 = Reserved
0010 = Compare mode: toggle output on match (CCPxIF bit is set)
0011 = Reserved
0100 = Capture mode: every falling edge
0101 = Capture mode: every rising edge
0110 = Capture mode: every 4th rising edge
0111 = Capture mode: every 16th rising edge
1000 = Compare mode: initialize CCPx pin low; on compare match, force CCPx pin high (CCPxIF bit is set)
1001 = Compare mode: initialize CCPx pin high; on compare match, force CCPx pin low (CCPxIF bit is set)
1010 = Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state)
1011 = Compare mode: trigger special event, reset timer, start A/D conversion on CCPx match (CCPxIF bit is set)
11xx = PWM mode

Note 1: These bits are not implemented on 28-pin devices and are read as '0'.

CCP2CON = 0X0C

9

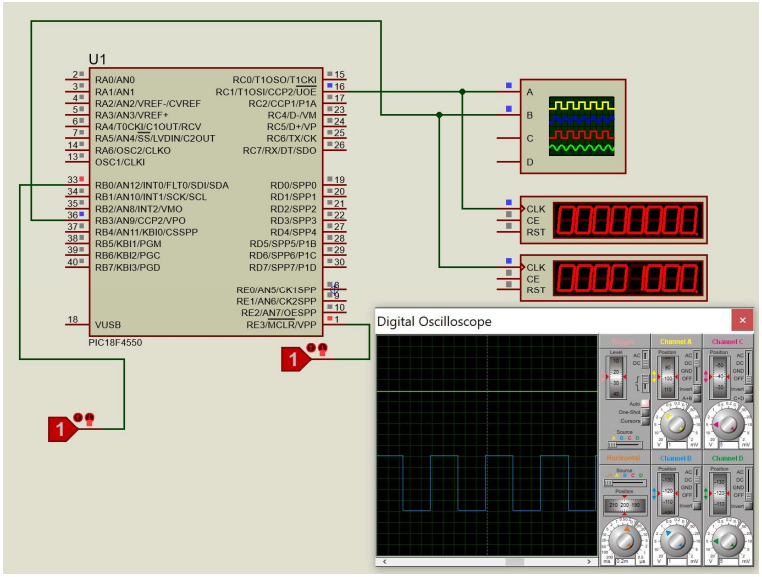
Código en XC8:




```
#pragma config PLLDIV = 1         // PLL P
#pragma config CPUDIV = OSC4_PLL6// Syst
#pragma config FOSC = XTPLL_XT  // Oscil
#pragma config PWRT = ON        // Power
#pragma config BOR = OFF        // Brown
#pragma config WDT = OFF        // Watch
#pragma config CCP2MX = OFF      // CCP2
#pragma config PBADEN = OFF     // PORTB
#pragma config MCLRE = ON       // MCLR
#pragma config LVP = OFF        // Singl

#include <xc.h>
#define _XTAL_FREQ 16000000UL       //fr

void init_config(void){
    PR2 = 249;          //PWMfreq=1KHz
    CCPR2L = 125;       //PWMdc=50%
    //TRISCbits.RC1 = 0; //CCP2MX = ON
    TRISBbits.RB3 = 0;  //CCP2MX = OFF
    T2CON = 0x06;       //PSC1:16
    CCP2CON = 0x0C;     //PWM mode
}

void main(void){
    init_config();
    while(1){
        if(PORTBbits.RB0 == 1){
            CCP2CON = 0x0C; //CCP2 PWM
        }
        else{
            CCP2CON = 0x00; //CCP2 OFF
        }
    };
}
```

10

Simulación

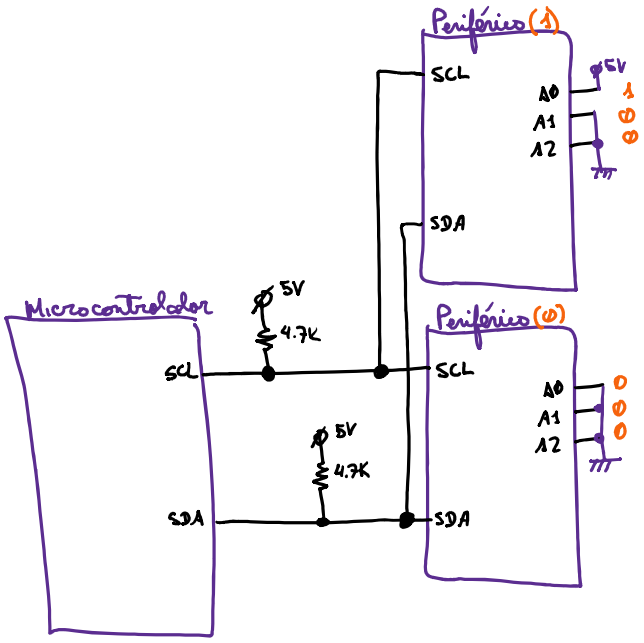


Comunicaciones seriales

- ¿Por qué necesitamos comunicarnos serialmente con el microcontrolador?
  - Hay periféricos especializados no incluídos dentro del microcontrolador
  - Para conectar dichos periféricos hay que saber qué protocolo e interfaz utiliza.
  - I2C, SPI, 1-wire, UART, I2S, etc

Comunicación I²C

- Inter-Integrated Circuit
- Serial síncrono
- Dos líneas para la comunicación (SCL y SDA)
- Direcciones de 7 bits
- Topología tipo bus
- Diferentes tipos de periféricos:
  - Memorias EEPROM (24Cxx)
  - Expansores de puertos (PCF8574)
  - Drivers de motores de paso
  - Conversores A/D
  - RTC (DS1307)
  - Sensores de temperatura (MCP9808)
  - etc



13

I2C:

- Tipo de periféricos I²C que encontramos:
  - DS1307 – RTC
  - PCF8574 – Expansor de 8 I/Os
  - 24C01 hasta 24C1024 – Memorias EEPROM
  - AD7997 – A/D 8ch 12bit
  - MCP9808 - Sensor de temperatura de precisión
  - PCA9685 – Servo controller up to 16 units
  - Devantech SRF08 – Sensor de ultrasonido
  - SH1106 – OLED interface display
  - HTU21D – Sensor de temperatura/humedad
  - MPU6050 – Acelerómetro
  - PCF8591 – A/D 4ch & D/A 1ch 8bit converter
  - HT16K33 – LED driver
  - DS3231 – RTC
  - DS1621 – Thermometer and Thermostat
  - SHT31 – Temperature & Humidity Sensor
  - MCP23016 - 16 I/O port expander

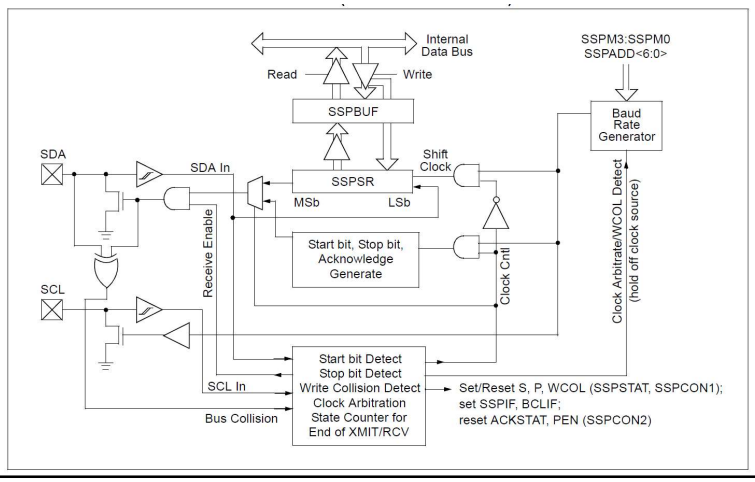
14

Interface I2C en el Microcontrolador PIC

- Dos opciones:
  - Hardware (módulo periférico MSSP)
  - Software (implementar una librería)

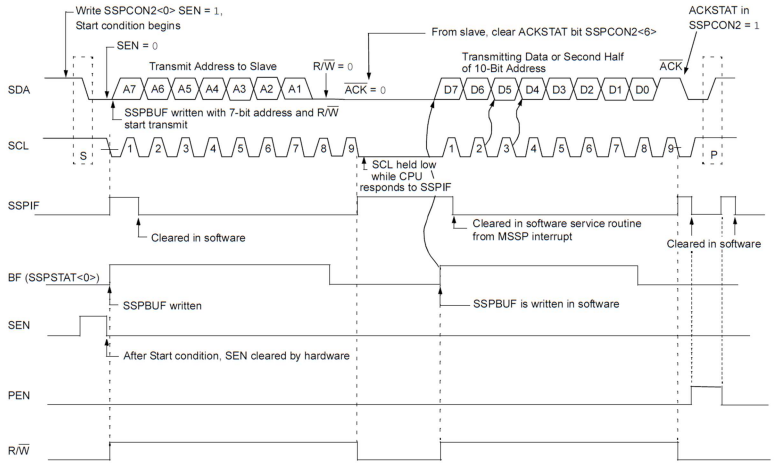
El MSSP en modo I2C Maestro

- Referencia: capítulo 19.4.6 de la hoja técnica del PIC18F4550



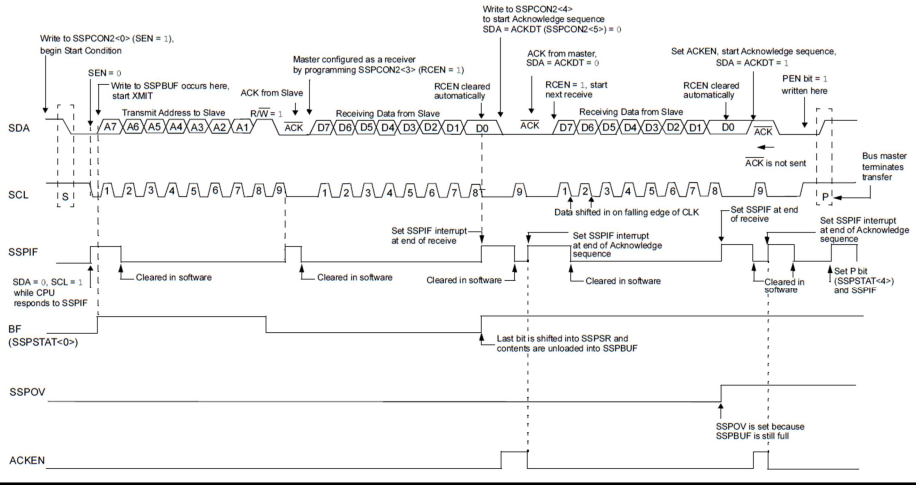
El MSSP en modo I2C Maestro

- Evento de transmisión:



El MSSP en modo I2C Maestro

- Evento de recepción:



El MSSP en modo I2C Maestro

- Registros relacionados: El SSPSTAT

REGISTER 19-3: SSPSTAT: MSSP STATUS REGISTER (I²C™ MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P(1)	S(1)	R/$\overline{W}$(2,3)	UA	BF
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 **SMP:** Slew Rate Control bit
In Master or Slave mode:
1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit
In Master or Slave mode:
1 = Enable SMBus specific inputs
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit
In Master mode:
Reserved.
In Slave mode:
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit(1)
1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

bit 3 **S:** Start bit(1)
1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ Information bit(2,3)
In Slave mode:
1 = Read
0 = Write
In Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress

bit 1 **UA:** Update Address bit (10-Bit Slave mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit
In Transmit mode:
1 = SSPBUF is full
0 = SSPBUF is empty
In Receive mode:
1 = SSPBUF is full (does not include the $\overline{ACK}$ and Stop bits)
0 = SSPBUF is empty (does not include the $\overline{ACK}$ and Stop bits)

Note 1: This bit is cleared on Reset and when SSPEN is cleared.
2: This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{ACK}$ bit.
3: ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Active mode.

19

El MSSP en modo I2C Maestro

- Registros relacionados: El SSPCON1

REGISTER 19-4: SSPCON1: MSSP CONTROL REGISTER 1 (I²C™ MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
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 **WCOL:** Write Collision Detect bit
In Master Transmit mode:
1 = A write to the SSPBUF register was attempted while the I²C conditions were not valid for a transmission to be started (must be cleared in software)
0 = No collision
In Slave Transmit mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision
In Receive mode (Master or Slave modes):
This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit
In Receive mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)
0 = No overflow
In Transmit mode:
This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit
1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins(1)
0 = Disables serial port and configures these pins as I/O port pins(1)

bit 4 **CKP:** SCK Release Control bit
In Slave mode:
1 = Release clock
0 = Holds clock low (clock stretch), used to ensure data setup time
In Master mode:
Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits
1111 = I²C Slave mode, 10-bit address with Start and Stop bit interrupts enabled(2)
1110 = I²C Slave mode, 7-bit address with Start and Stop bit interrupts enabled(2)
1011 = I²C Firmware Controlled Master mode (slave Idle)(2)
1000 = I²C Master mode, clock = FOSC/(4 * (SSPADD + 1))(2,3)
0111 = I²C Slave mode, 10-bit address(2)
0110 = I²C Slave mode, 7-bit address(2)

Note 1: When enabled, the SDA and SCL pins must be properly configured as input or output.
2: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.
3: Guideline only; exact baud rate slightly dependent upon circuit conditions, but the highest clock rate should not exceed this formula. SSPADD values of '0' and '1' are not supported.

20

El MSSP en modo I2C Maestro

- Registros relacionados: El SSPCON2

REGISTER 19-5: SSPCON2: MSSP CONTROL REGISTER 2 (I²C™ MASTER MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT(1)	ACKEN(2)	RCEN(2)	PEN(2)	RSEN(2)	SEN(2)
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 **GCEN:** General Call Enable bit (Slave mode only)
Unused in Master mode.

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)(1)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit(2)
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master Receive mode only)(2)
1 = Enables Receive mode for I²C
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit(2)
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

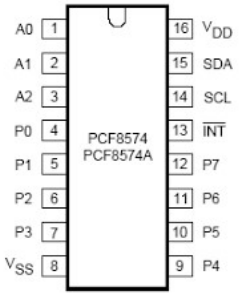
bit 1 **RSEN:** Repeated Start Condition Enable bit(2)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enable/Stretch Enable bit(2)
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle

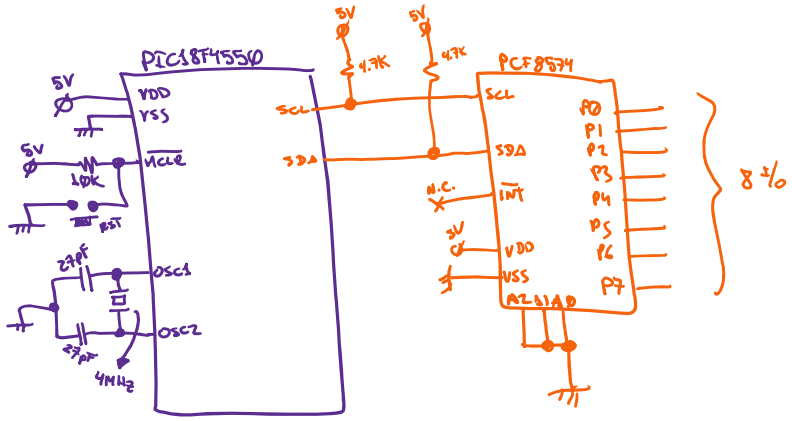
Note 1: Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
2: If the I²C module is active, these bits may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

Ejemplo: I2C con el MSSP para conectar un PCF8574

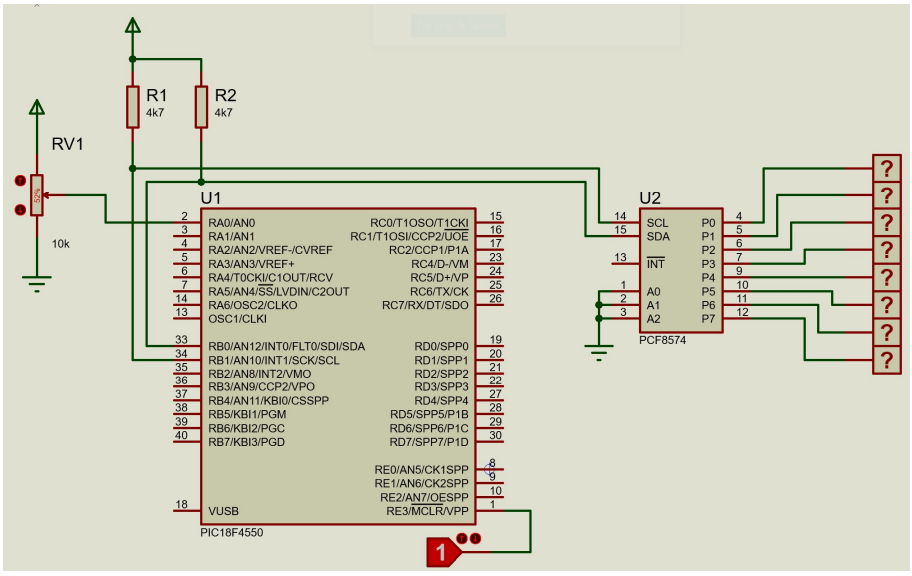
- PCF8574: Expansor de puertos de 8 bits bidireccional
  - Datasheet: https://www.ti.com/lit/gpn/pcf8574



Ejemplo: I2C con el MSSP para conectar un PCF8574



Simulación en Proteus



Palabras de control para el PCF8574

7bit (dirección de esclavo)

BYTE	BIT							
	7 (MSB)	6	5	4	3	2	1	0 (LSB)
I^2C slave address	L	H	L	L	A2	A1	A0	R/$\overline{W}$
I/O data bus	P7	P6	P5	P4	P3	P2	P1	P0

Evento lectura: 0X41

Evento escritura: 0X40

25

Procedimiento para escribir un dato en el PCF8574

- Evento de inicio (start)
- Enviar la palabra de control (envío de la dirección del esclavo: 0x40
- Esperar el ACK
- Enviar el dato a escribir (0xAA)
- Esperar el ACK
- Evento de parada (stop)

$$100000 = \frac{FOSC}{(4X(SSPADD+1))}$$ (48EC)

SSPADD = 119

26

Configuración del módulo MSSP:

- SSPCON1.SSPEN = 1 (Habilitador del módulo MSSP)
- Modo de trabajo: Master

1000 = I^2C Master mode, clock = FOSC/(4 * (SSPADD + 1))(2,3)

- Valor SSPADD (velocidad)

Fosc = 48MHz
Bitrate = 100000

$$SSPADD = \frac{\left(\frac{F_{osc}}{4}\right)}{BitRate} - 1 \quad = 119$$

27

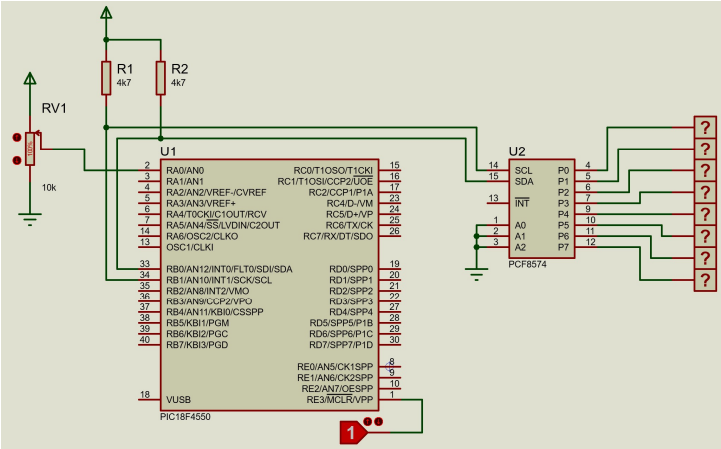
Código ejemplo en XC8 para escribir 0xAA alternado con 0x55 en el puerto del PCF8574

```
#pragma config PLLDIV = 1         // PLL Prescaler Sele
#pragma config CPUDIV = OSC1_PLL2// System Clock Post
#pragma config FOSC = XTPLL_XT  // Oscillator Selecti
#pragma config PWRT = ON        // Power-up Timer Ena
#pragma config BOR = OFF        // Brown-out Reset En
#pragma config WDT = OFF        // Watchdog Timer Ena
#pragma config CCP2MX = OFF      // CCP2 MUX bit (CCP
#pragma config PBADEN = OFF     // PORTB A/D Enable b
#pragma config MCLRE = ON       // MCLR Pin Enable bi
#pragma config LVP = OFF        // Single-Supply ICSP

#include <xc.h>
#define _XTAL_FREQ 48000000UL       //frecuencia de t

unsigned char dato = 0xAA;

void init_config(void){
    SSPCON1bits.SSPEN = 1;      //Encendemos el MSSP
    SSPCON1bits.SSPM3 = 1;
    SSPCON1bits.SSPM2 = 0;
    SSPCON1bits.SSPM1 = 0;
    SSPCON1bits.SSPM0 = 0;      //MSSP en I2C master
    SSPADD = 119;               // clock en 100khz
}
```

```
void pcf8574_escribir(unsigned char dato){
    SSPCON2bits.SEN = 1;          //evento de inicio
    while(SSPCON2bits.SEN == 1);    //espero a que
    SSPBUF = 0x40;
    while(SSPSTATbits.BF ==1);  //espero a que se t
    while(SSPSTATbits.R_nW ==1);
    SSPBUF = dato;
    while(SSPSTATbits.BF ==1);  //espero a que se t
    while(SSPSTATbits.R_nW ==1);
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
}

void main(void){
    init_config();
    while(1){
        pcf8574_escribir(0x55);
        __delay_ms(100);
        pcf8574_escribir(0xAA);
        __delay_ms(100);
    }
}
```

28

Ejemplo: Lectura de señal analógica en AN0 a través del ADC y emisión del resultado de la conversión en 8 bits por los puertos del PCF8574



Ejemplo: Lectura de señal analógica en AN0 a través del ADC y emisión del resultado de la conversión en 8 bits por los puertos del PCF8574

```
15  void init_config(void){
16      SSPCON1bits.SSPEN = 1;      //Encendemos el MSSI
17      SSPCON1bits.SSPM3 = 1;
18      SSPCON1bits.SSPM2 = 0;
19      SSPCON1bits.SSPM1 = 0;
20      SSPCON1bits.SSPM0 = 0;      //MSSP en I2C master
21      SSPADD = 119;               // clock en 100khz
22      ADCON2 = 0x24;      // tiempo de conversion ADFI
23      ADCON1 = 0x0E;      //AN0 como anbalógico
24      ADCON0 = 0x01;      //Activar A/D
25  }
26
27  void pcf8574_escribir(unsigned char dato){
28      SSPCON2bits.SEN = 1;        //evento de inicio
29      while(SSPCON2bits.SEN == 1);    //espero a que s
30      SSPBUF = 0x40;              //palabra de cor
31      while(SSPSTATbits.BF ==1);  //espero a que se te
32      while(SSPSTATbits.R_nW ==1);
33      SSPBUF = dato;              //dato a escribir er
34      while(SSPSTATbits.BF ==1);  //espero a que se te
35      while(SSPSTATbits.R_nW ==1);
36      SSPCON2bits.PEN = 1;        //evento de parada
37      while(SSPCON2bits.PEN == 1);
38  }
39
40  void main(void){
41      init_config();
42      while(1){
43          ADCON0bits.GODONE = 1;
44          while(ADCON0bits.GODONE == 1);
45          pcf8574_escribir(ADRESH);
46      }
47  }
```

Fin de la sesión