Microcontroladores

Semestre: 2021-1

Profesor: Kalun José Lau Gan

Semana 11-12: Comunicación serial en el PIC18F4550

1

Preguntas previas:

- Dónde encuentro info sobre 74HC595?
  - AV -> Unidad2 -> Laboratorio
- Plazo para LB3?
  - Domingo 13/06 18:00

2

Agenda

- Comunicación serial en el PIC18F4550
- Hay dos tipos de comunicación serial: asíncrono y síncrono

  Asíncronos (no esta presente la línea de reloj): UART, 1-wire

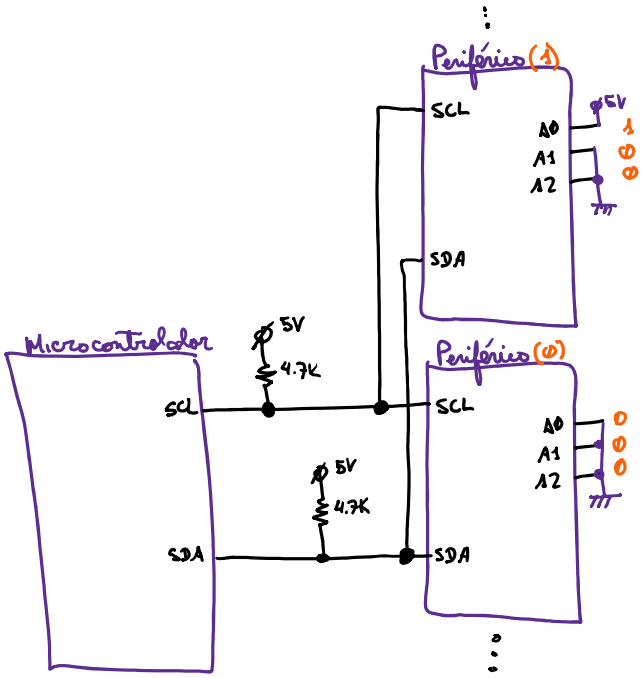
  Síncronos: I^2C, I^2S, SPI
- Hay interfaces y protocolos que vienen implementados por hardware y se pueden realizar en la mayoría de casos en software.
  - En plataforma Arduino tenemos “Software Serial”
- Hay protocolos “custom” donde hay que revisar la hoja técnica del periférico para establecer la comunicación. Ej. DHT11

Escenarios para emplear comunicación serial

- Enviar datos entre dos puntos distantes
- Conectarse con un periférico externo
  - Sensores
  - Memorias
  - RTC
- Monitoreo remoto
- Ahorro de líneas de I/O

Comunicación I²C

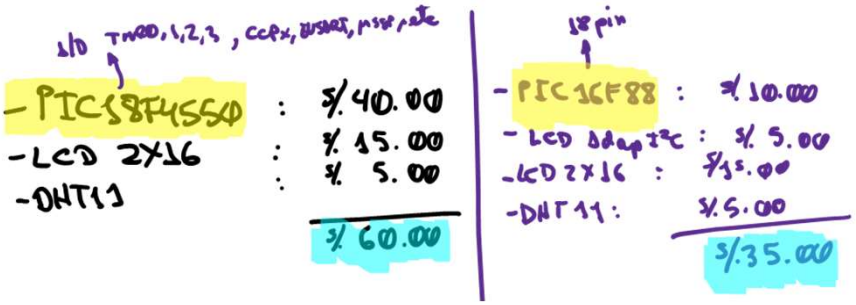
- Inter-Integrated Circuit
- Serial síncrono
- Dos líneas para la comunicación (SCL y SDA)
- Direcciones de 7 bits
- Topología tipo bus
- Diferentes tipos de periféricos:
  - Memorias EEPROM (24Cxx)
  - Expansores de puertos (PCF8574)
  - Drivers de motores de paso
  - Conversores A/D
  - RTC (DS1307)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



5

Estrategia de reducción de costos en desarrollo con microcontroladores

- El principal costo generalmente se da en el modelo de microcontrolador a emplear. Es por ello que se debe de escoger un dispositivo adecuado para la aplicación de acuerdo a los requerimientos.
- Haciendo una combinación de periféricos externos y microcontrolador podremos reducir el costo en la elaboración de aplicaciones.



6

I2C:

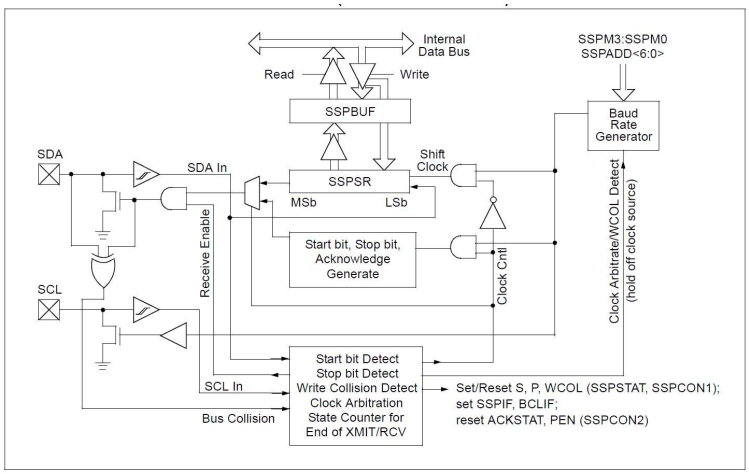
- Tipo de periféricos I²C que encontramos:
  - DS1307 – RTC
  - PCF8574 – Expansor de 8 I/Os
  - 24C01 hasta 24C1024 – Memorias EEPROM
  - AD7997 – A/D 8ch 12bit
  - MCP9808 - Sensor de temperatura de precisión
  - PCA9685 – Servo controller up to 16 units
  - Devantech SRF08 – Sensor de ultrasonido
  - 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
  - PCF8575 – 16 I/O expander

Interface I2C en el Microcontrolador PIC

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

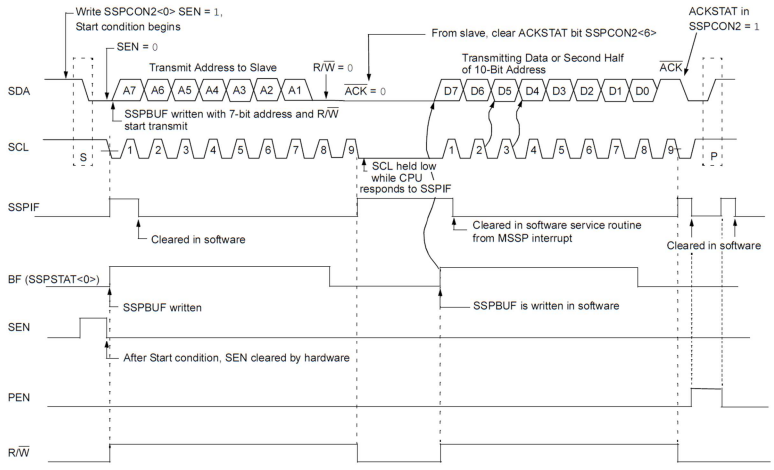
El MSSP en modo I2C Maestro

- Referencia: capítulo 19.4.6 de la hoja técnica del PIC18F4550



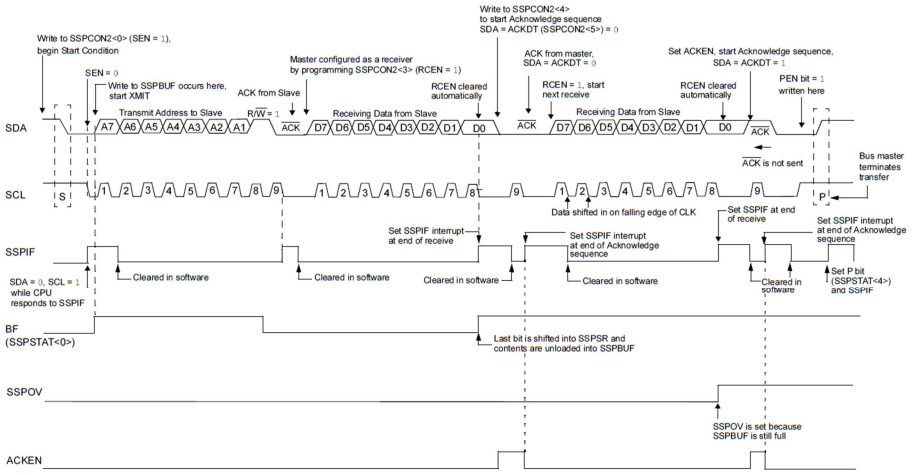
El MSSP en modo I2C Maestro

- Evento de transmisión:



El MSSP en modo I2C Maestro

- Evento de recepción:



11

El MSSP en modo I2C Maestro

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)

- Registros relacionados: El SSPSTAT

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.

12

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.

13

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

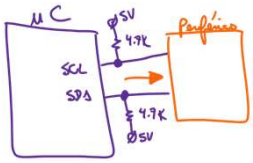
bit 0 **SEN:** Start Condition Enable/Stretch Enable bit(2)
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle

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).

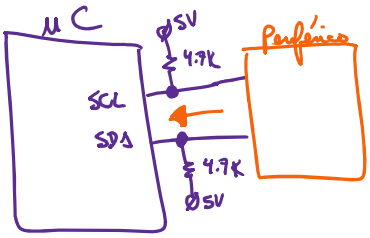
14

Agenda

- Proceso de escritura en I2C con el microcontrolador PIC18F4550



- Proceso de lectura en I2C con el microcontrolador PIC18F4550



15

Configuración del MSSP (modo I²C maestro) para comunicarnos con la 24C01

- Revisar hoja técnica del microcontrolador PIC18F4550 (capítulo 19.4)

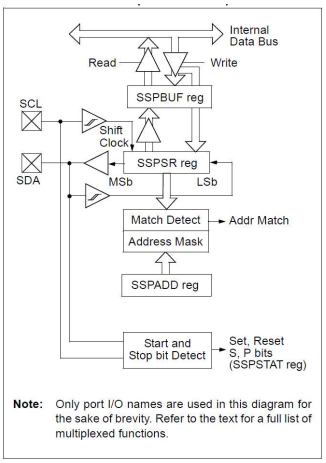
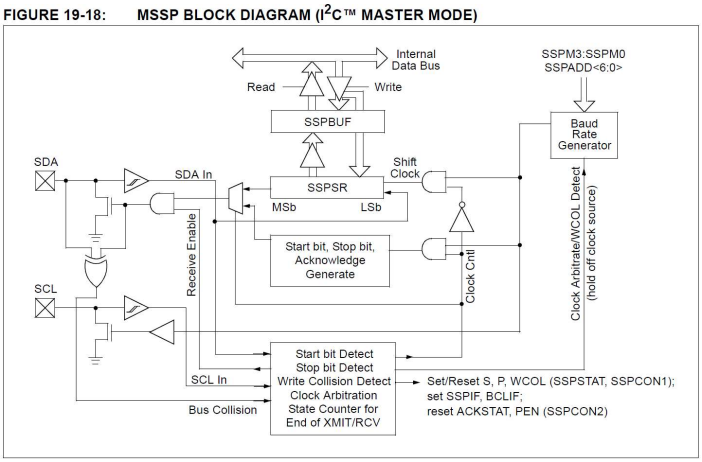
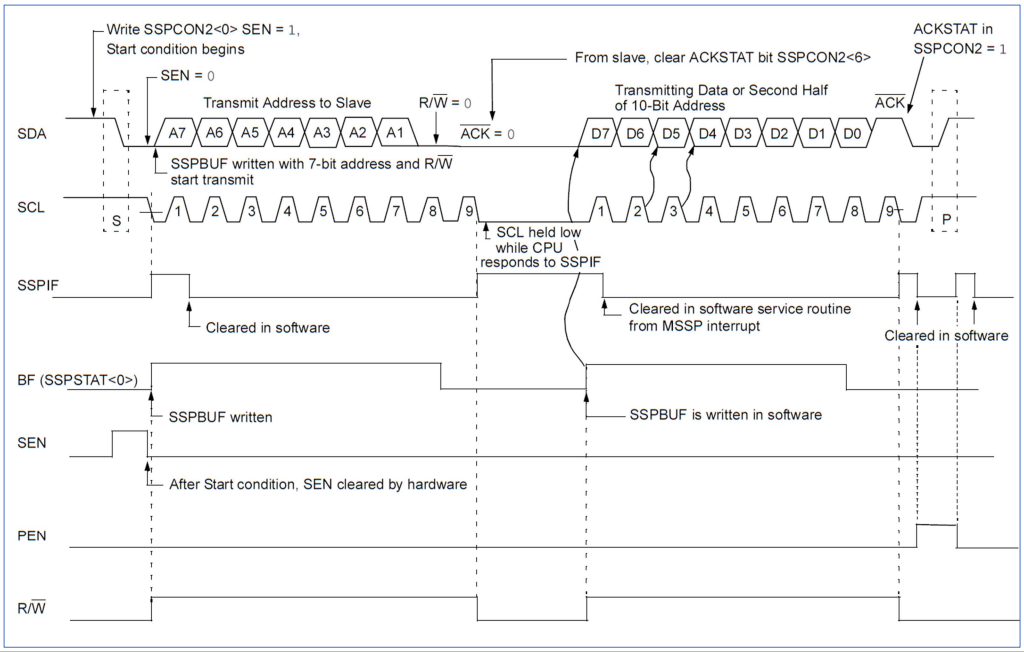


FIGURE 19-18: MSSP BLOCK DIAGRAM (I²C™ MASTER MODE)



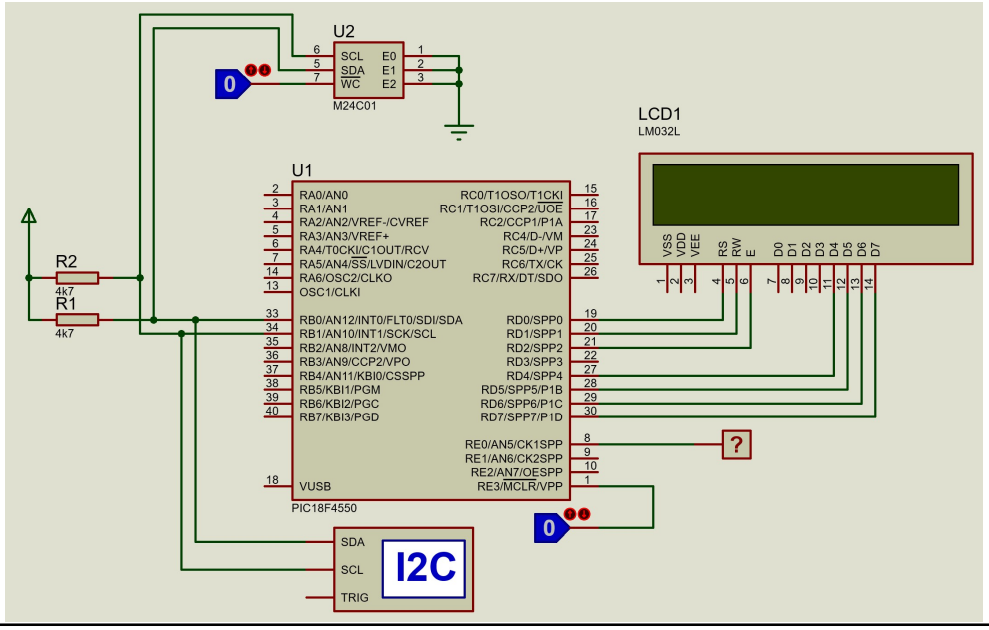
16

Recordando I²C



17

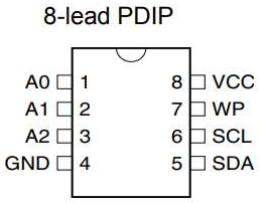
Circuito de prueba



18

Memoria 24C01

- Memoria de 1Kbit (128 x 8) – 128 direcciones de 8 bits
- Interface I²C
- Compatibilidad con 100KHz y 400KHz de frecuencia de datos
- WP: Write Protect (‘1’ solo lectura, ‘0’ escritura y lectura)
- Trabaja con niveles 3.3V y 5V
- 1M ciclos de escritura
- Garantía de retención de datos: 100 años.



Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect
GND	Ground
VCC	Power Supply

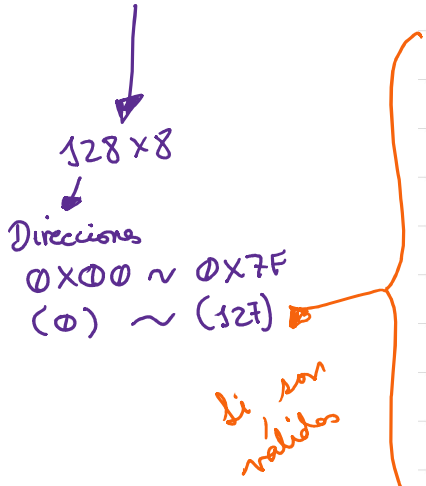
19

Ejemplo 1:

- Escribir los datos 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39 en la memoria 24C01 a partir de su dirección 100

20

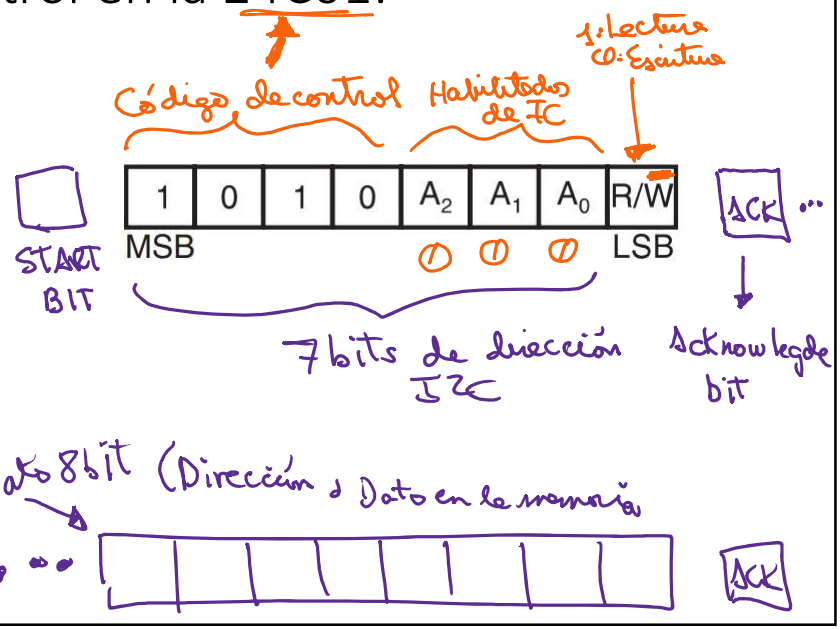
Verificar si las direcciones a usar son válidas en la 24C01:



Dirección	Dato
100	0x30
101	0x31
102	0x32
103	0x33
104	0x34
105	0x35
106	0x36
107	0x37
108	0x38
109	0x39

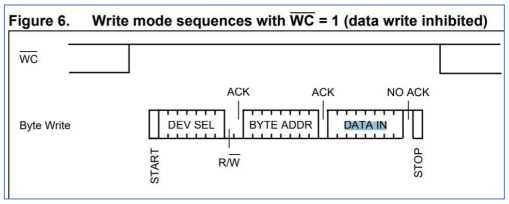
Palabra de control en la 24C01:

- Palabra de control será 0xA0 para que la memoria pase a modo de escritura



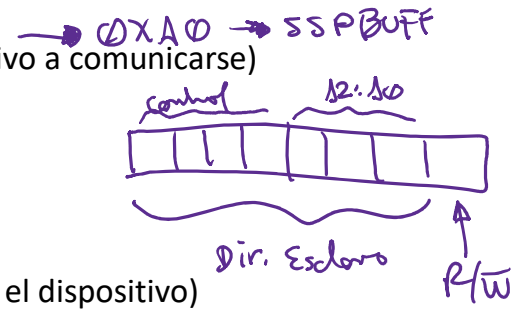
Procedimiento para escribir datos en la 24C01

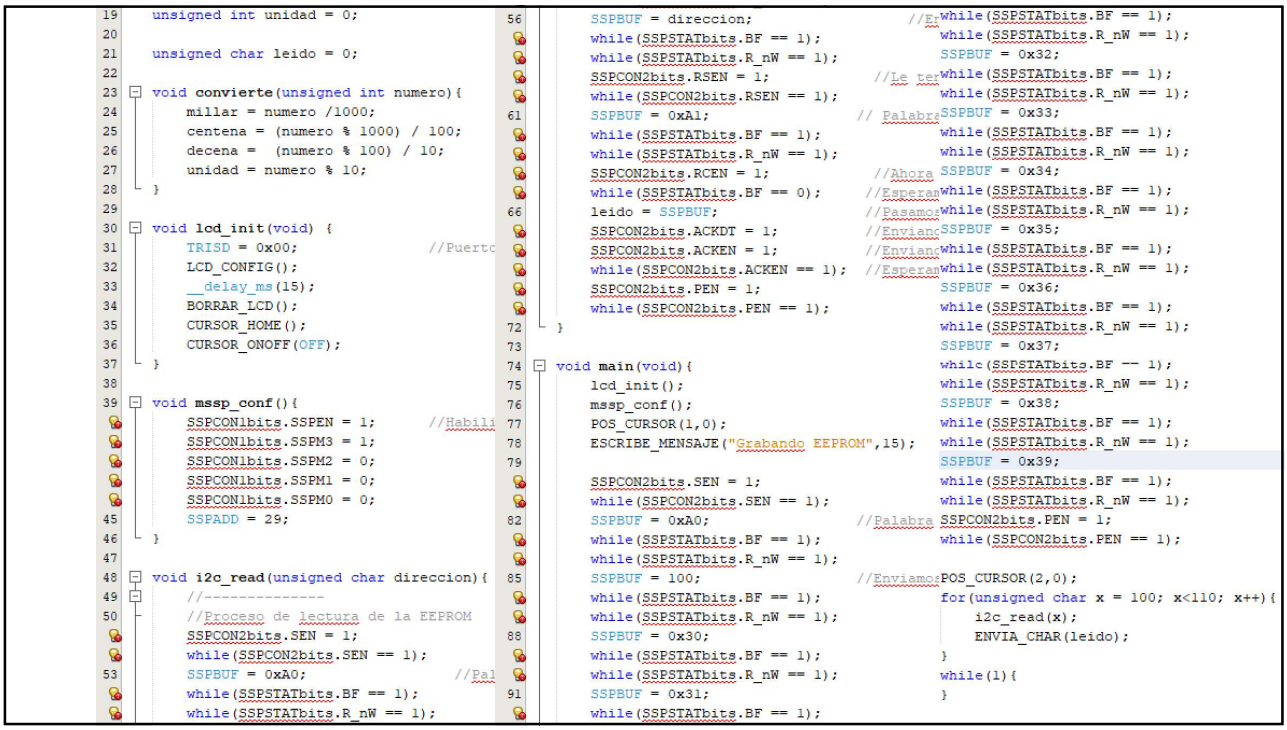
1. Start (S)
2. Byte control (habilitación del dispositivo a comunicarse)
3. ACK
4. Enviar dirección de memoria (100)
5. ACK
6. Dato1 (Se va a almacenar 0x30 en 100)
7. ACK
8. Dato2 (Se va a almacenar 0x31 en 101)
9. ACK
10. .
11. .
12. .
13. Dato9 (Se va a almacenar 0x39 en 109)
14. ACK
15. Stop (P)



Procedimiento para leer un dato en la 24C01

1. Start (S)
2. Byte control (habilitación del dispositivo a comunicarse)
3. ACK
4. Enviar dirección de memoria (100)
5. ACK
6. Restart
7. Byte control (cambiar a modo lectura el dispositivo)
8. Ack
9. Dato1 (Se va a leer el contenido de 100 y se almacenará en SSPBUF)
10. nACK
11. Stop (P)





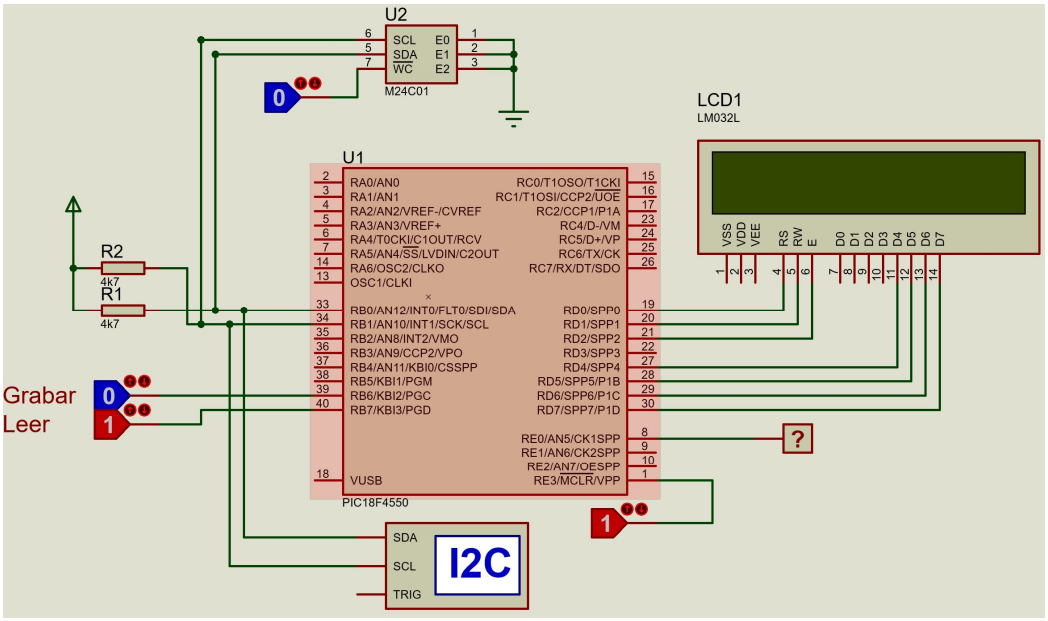
```
unsigned int unidad = 0;

unsigned char leido = 0;

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

void lcd_init(void) {
    TRISD = 0x00;                //Puerto
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void mssp_conf(){
    SSPCON1bits.SSPEN = 1;       //Habili
    SSPCON1bits.SSPM3 = 1;
    SSPCON1bits.SSPM2 = 0;
    SSPCON1bits.SSPM1 = 0;
    SSPCON1bits.SSPM0 = 0;
    SSPADD = 29;
}

void i2c_read(unsigned char direccion){
    //--------------
    //Proceso de lectura de la EEPROM
    SSPCON2bits.SEN = 1;
    while(SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;                   //Pal
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = direccion;                  //Er
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.RSEN = 1;            //Le ter
    while(SSPCON2bits.RSEN == 1);
    SSPBUF = 0xA1;                  // Palabra
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.RCEN = 1;             //Ahora
    while(SSPSTATbits.BF == 0);      //Esperan
    leido = SSPBUF;                  //Pasamos
    SSPCON2bits.ACKDT = 1;           //Enviand
    SSPCON2bits.ACKEN = 1;           //Enviand
    while(SSPCON2bits.ACKEN == 1);   //Esperan
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
}

void main(void){
    lcd_init();
    mssp_conf();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Grabando EEPROM",15);

    SSPCON2bits.SEN = 1;
    while(SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;                  //Palabra
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 100;                   //Enviamos
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x30;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x31;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x32;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x33;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x34;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x35;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x36;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x37;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x38;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 0x39;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
    POS_CURSOR(2,0);
    for(unsigned char x = 100; x<110; x++){
        i2c_read(x);
        ENVIA_CHAR(leido);
    }
    while(1){
    }
```


Ejemplo 2:

- Desarrollar un proceso de escritura de los siguientes valores en la memoria 24C01 de manera secuencial y a partir de la dirección 100
  - “Ingenieria” (10 caracteres ASCII)

Dirección (DEC)	Dato (ASCII)	Dato (HEX)
100	I	0x49
101	n	0x6E
102	g	0x67
103	e	0x65
104	n	0x6E
105	i	0x69
106	e	0x65
107	r	0x72
108	i	0x69
109	a	0x61

Circuito ejemplo



A tener en cuenta:

- El microcontrolador PIC18F4550 será el maestro en el bus I²C
- El 24C01 será dispositivo esclavo en el bus I²C
- El módulo MSSP por lo tanto debe de configurarse para que funcione en modo I²C Maestro
- Palabra de control de 24C01 (revisar datasheet)

	Device Type Identifier(1)				Chip Enable(2),(3)			RW
	b7	b6	b5	b4	b3	b2	b1	b0
M24C01 Select Code	1	0	1	0	E2	E1	E0	RW

7bit slave address

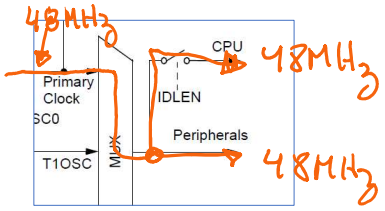
Evento lectura : 0XA1
Evento escritura : 0XA0

A tener en cuenta:

- Se trabajará en 100KHz de frecuencia de bus I²C
- Se tiene que averiguar el valor de SSPADD para que trabaje a 100KHz

1000 = I²C Master mode, clock = Fosc/(4 * (SSPADD + 1))(2,3)

- Sabiendo que Fosc = 48MHz, recordando:



- Despejando:

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

SSPADD = 119

Código en XC8:

```
#pragma config PLLDIV = 1       // PLL P
#pragma config CPUDIV = OSC1_PLL2// 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 = ON      // CCP2
#pragma config PBADEN = OFF     // PORTB
#pragma config MCLRE = ON       // MCLR
#pragma config LVP = OFF        // Singl

#include <xc.h>
#include "LCD.h"
#define _XTAL_FREQ 48000000UL       //fr

unsigned char datoin = 0;
unsigned char nombre[] = {"Ingenieria"};

void lcd_init(void) {
    TRISD = 0x00;               //Puerto
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void mssp_conf(){
    SSPCON1bits.SSPEN = 1;      //Habili
    SSPCON1bits.SSPM3 = 1;
    SSPCON1bits.SSPM2 = 0;
    SSPCON1bits.SSPM1 = 0;
    SSPCON1bits.SSPM0 = 0;
    SSPADD = 29;
}
```

```
unsigned char i2c_read(unsigned char direccion){
    //-------------
    //Proceso de lectura de la EEPROM
    unsigned char leido = 0;
    SSPCON2bits.SEN = 1;
    while(SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;                  //Palabra de
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = direccion;                 //Envia
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.RSEN = 1;               //Le tengo
    while(SSPCON2bits.RSEN == 1);
    SSPBUF = 0xA1;                  // Palabra de
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.RCEN = 1;               //Ahora el
    while(SSPSTATbits.BF == 0);         //Esperamos
    leido = SSPBUF;                     //Pasamos lo
    SSPCON2bits.ACKDT = 1;              //Enviando e
    SSPCON2bits.ACKEN = 1;              //Enviando e
    while(SSPCON2bits.ACKEN == 1);      //Esperamos
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
    return(leido);
}
```

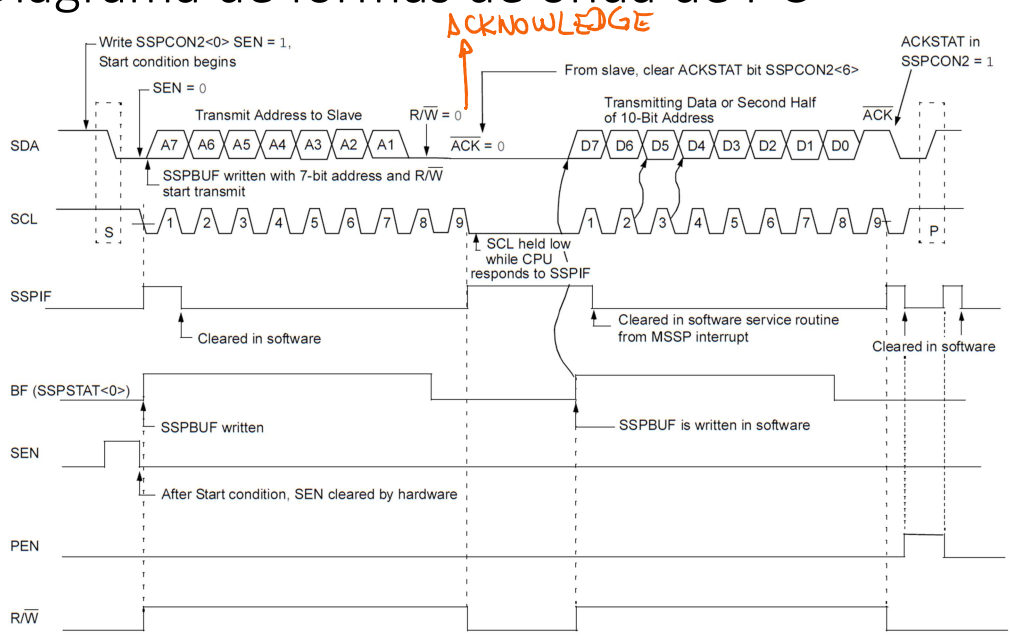
```
void i2c_write(unsigned char direccion1, unsigned char dato1){
    SSPCON2bits.SEN = 1;
    while(SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;                  //Palabra de control
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = direccion1;                    //Enviamos direcció
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = dato1;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
    __delay_ms(10);
}

void main(void){
    lcd_init();
    mssp_conf();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Grabando EEPROM",15);
    for(unsigned char x=0; x<10; x++){
        i2c_write(x+100, nombre[x]);
    }

    POS_CURSOR(2,0);
    for(unsigned char x=0;x<10;x++){
        datoin = i2c_read(x+100);
        ENVIA_CHAR(datoin);
    }
    while(1){
    }
}
```

Ejemplo 3: Datos a grabar en la EEPROM:

- A partir de la dirección 50:
  - "Ingeniero Meca-Elec"

Dirección	Dato
50	I
51	n
52	g
53	e
54	n
55	i
56	e
57	r
58	o
59	
60	M
61	e
62	c
63	a
64	-
65	E
66	l
67	e
68	c

Diagrama de formas de onda de I²C



Procedimiento para escribir datos en el 24C01

1. Evento “Start” (S)
2. Byte de control (Enviar la dirección de esclavo) → 0xA0
3. ACK
4. Dirección de la memoria a acceder → 50
5. ACK
6. Dato1
7. ACK
8. Dato2
9. ACK
10. Dato3
11. ACK
12. Evento “Stop” (P)

Por si deseas grabar un lote de datos

Dirección	Dato
50	Dato1
51	Dato2
52	Dato3

Procedimiento para leer datos en el 24C01

1. Evento “Start” (S)
2. Byte de control (Enviar la dirección de esclavo) → 0xA0
3. ACK
4. Dirección de la memoria a acceder → 50
5. ACK
6. Evento Restart
7. Byte de control → 0xA1
8. ACK
9. MSSP modo lectura (RCEN=1)
10. Dato1
11. ACK
12. Dato2
13. ACK
14. Dato3
15. noACK
16. Evento “Stop” (P)

Si deseas leer los tres en el mismo evento

Dirección	Dato
50	Dato1
51	Dato2
52	Dato3

Código en XC8:

```
#pragma config PLLDIV = 1
#pragma config CPUDIV = OSC1_PLL2
#pragma config FOSC = XTPLL_XT
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config WDT = OFF
#pragma config CCP2MX = ON
#pragma config PBADEN = OFF
#pragma config MCLRE = ON
#pragma config LVP = OFF

#include <xc.h>
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

unsigned char button1_stat = 0;
unsigned char button2_stat = 0;
unsigned char frase[] = {"Ingenier
unsigned char dato = 0;

void lcd_init(void) {
    TRISD = 0x00;
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void mssp_config(void){
    SSPCON1bits.SSPEN = 1;
    SSPCON1bits.SSPM3 = 1;
    SSPCON1bits.SSPM2 = 0;
    SSPCON1bits.SSPM1 = 0;
    SSPCON1bits.SSPM0 = 0;
    SSPADD = 119;
}
```

```
void m24c01_write(void){
    SSPCON2bits.SEN = 1;        //Start
    while (SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;              //Activ
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 50;                //Direc
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    for (unsigned char x=0;x<14;x++){
        SSPBUF = frase[x];
        while(SSPSTATbits.BF == 1);
        while(SSPSTATbits.R_nW == 1);
    }
    SSPCON2bits.PEN = 1;        //Stop
    while(SSPCON2bits.PEN == 1);
    __delay_ms(15);

    SSPCON2bits.SEN = 1;        //Start
    while (SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;              //Activ
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 64;                //Direc
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    for (unsigned char x=14;x<20;x++){
        SSPBUF = frase[x];
        while(SSPSTATbits.BF == 1);
        while(SSPSTATbits.R_nW == 1);
    }
    SSPCON2bits.PEN = 1;        //Stop
    while(SSPCON2bits.PEN == 1);
}
```

```
unsigned char m24c01_read(unsigned char direccion){
    unsigned char leido = 0;
    SSPCON2bits.SEN = 1;        //Start
    while (SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;              //Activación de 24C
    while(SSPSTATbits.BF == 1);     //Pregunto si
    while(SSPSTATbits.R_nW == 1);   //Pregunto si
    SSPBUF = direccion;
    while(SSPSTATbits.BF == 1);     //Pregunto si
    while(SSPSTATbits.R_nW == 1);   //Pregunto si
    SSPCON2bits.RSEN = 1;       //Restart
    while(SSPCON2bits.RSEN == 1);
    SSPBUF = 0xA1;              //Activación de 24C
    while(SSPSTATbits.BF == 1);     //Pregunto si
    while(SSPSTATbits.R_nW == 1);   //Pregunto si
    SSPCON2bits.RCEN = 1;       //modo lectura en
    while(SSPSTATbits.BF == 0); //Esperamos a que s
    leido = SSPBUF;
    SSPCON2bits.ACKDT = 1;      //noACK
    SSPCON2bits.ACKEN = 1;      //iniciamos secuenc
    while(SSPCON2bits.ACKEN == 1);
    SSPCON2bits.PEN = 1;        //Stop
    while(SSPCON2bits.PEN == 1);
    return(leido);
    //__delay_ms(10);
}
```

```
void main(void){
    lcd_init();
    mssp_config();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Semana 11 EEPROM",16);
    while(1){
        if(PORTBbits.RB6 == 1 && button1_stat == 0){
            m24c01_write();
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Writing",7);
            button1_stat = 1;
        }
        if(PORTBbits.RB6 ==0 && button1_stat ==1){
            button1_stat = 0;
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("       ",7);
        }
        if(PORTBbits.RB7 == 1 && button2_stat == 0){
            //Aca leeremos la memoria letra por letra
            POS_CURSOR(2,0);
            for(unsigned char y=0;y<20;y++){
                dato = m24c01_read(50+y);
                ENVIA_CHAR(dato);
            }
            button2_stat = 1;
        }
        if(PORTBbits.RB7 ==0 && button2_stat ==1){
            button2_stat = 0;
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("                    ",20);
        }
    }
}
```

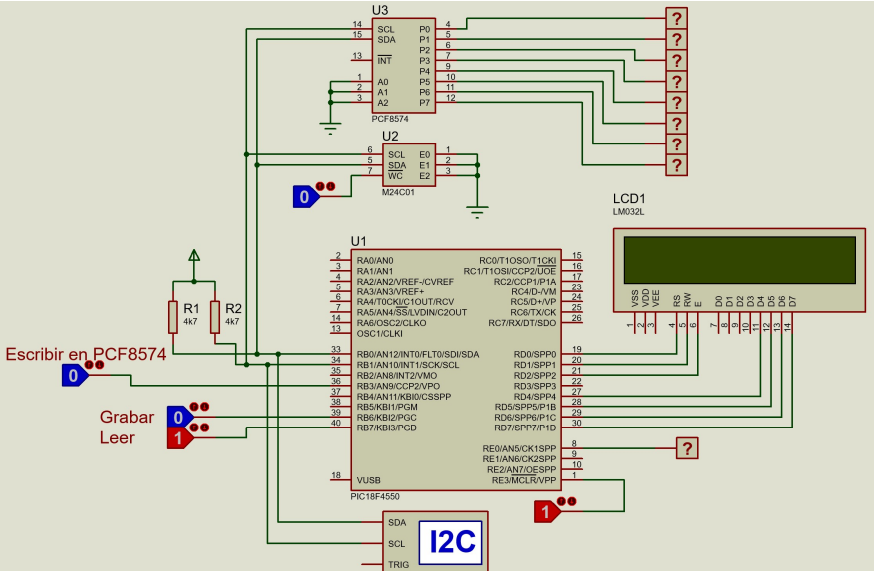

Nuevo ejercicio:

- M24C01: EEPROM 1Kbit (128x8)
- Se requiere grabar en memoria desde la dirección 20:
  - “KeepCalm UPC” – cadena de 12 caracteres

Dirección	Dato
20	K
21	e
22	e
23	p
24	C
25	a
26	l
27	m
28	
29	U
30	P
31	C

- Se requiere enviar un dato cualquiera hacia los I/Os del PCF8574

Circuito:



Nuevo ejercicio:

- M24C01: EEPROM 1Kbit (128x8)
- Palabra de control para seleccionar el M24C01:

Table 2. Device select code

Package	Device type identifier(1)				Chip Enable address			R/W
	b7	b6	b5	b4	b3	b2	b1	b0
TSSOP8,SO8,PDIP8, UFDFPN8	1	0	1	0	E2	E1	E0	R/W

7bit slave address

Evento lectura: 0XA1
Evento escritura: 0XA0

- Palabra de control para seleccionar el PCF8574:

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

7bit slave address

No hay conflicto

Evento lectura: 0X41
Evento escritura: 0X40

En el 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

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

= 119

39

Procedimiento para escribir datos en el 24C01

1. Evento "Start" (S)
2. Byte de control (Enviar la dirección de esclavo) → 0xA0
3. ACK
4. Dirección de la memoria a acceder → 20
5. ACK
6. Dato1
7. ACK
8. Dato2
9. ACK
10. Dato3
11. ACK
12. Evento "Stop" (P)

Por si deseas grabar un lote de datos

Dirección	Dato
20	Dato1
21	Dato2
22	Dato3

40

Procedimiento para leer datos en el 24C01

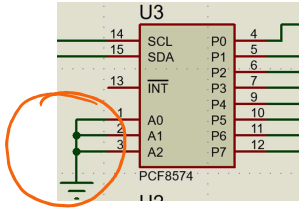
1. Evento “Start” (S)
2. Byte de control (Enviar la dirección de esclavo) → 0xA0
3. ACK
4. Dirección de la memoria a acceder → 20
5. ACK
6. Evento “Restart” (RS)
7. Byte de control → 0xA1 (la memoria pase a modo lectura)
8. ACK
9. MSSP modo lectura (RCEN=1)
10. Dato1
11. ACK
12. Dato2
13. ACK
14. Dato3
15. noACK
16. Evento “Stop” (P)

Si deseas leer los tres en el mismo evento

Dirección	Dato
20	Dato1
21	Dato2
22	Dato3

Procedimiento para escribir datos en la PCF8574

1. Evento o condición “Start” (S)
2. Byte de control (Enviar la dirección de esclavo) → 0x40
3. ACK
4. Dato en escribir en el puerto del PCF8574
5. ACK
6. Evento o condición “Stop” (P)



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

Código en XC8

```
#pragma config PLLDIV = 1
#pragma config CPUDIV = OSC1_PLL2
#pragma config FOSC = XTPLL_XT
#pragma config PWRT = ON
#pragma config BOR = OFF
#pragma config WDT = OFF
#pragma config CCP2MX = ON
#pragma config PBADEN = OFF
#pragma config MCLRE = ON
#pragma config LVP = OFF

#include <xc.h>
#include "LCD.h"
#define _XTAL_FREQ 48000000UL

unsigned char button1_stat = 0;
unsigned char button2_stat = 0;
unsigned char button3_stat = 0;
unsigned char frase[] = {"KeepCal

unsigned char letra = 0;


void lcd_init(void) {
    TRISD = 0x00;
    LCD_CONFIG();
    __delay_ms(15);
    BORRAR_LCD();
    CURSOR_HOME();
    CURSOR_ONOFF(OFF);
}

void mssp_conf(void){
    SSPCON1bits.SSPEN = 1;
    SSPCON1bits.SSPM3 = 1;
    SSPCON1bits.SSPM2 = 0;
    SSPCON1bits.SSPM1 = 0;
    SSPCON1bits.SSPM0 = 0;
    SSPADD = 119;
}
```

```
void m24c01_escribir(void){
    SSPCON2bits.SEN = 1;
    while(SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = 20;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    for(unsigned char x=0;x<12;x++){
        SSPBUF = frase[x];
        while(SSPSTATbits.BF == 1);
        while(SSPSTATbits.R_nW == 1);
    }
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
}

unsigned char m24c01_leer(unsigned char direccion){
    unsigned char dato_leido = 0;
    SSPCON2bits.SEN = 1;
    while(SSPCON2bits.SEN == 1);
    SSPBUF = 0xA0;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = direccion;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.RSEN = 1;
    while(SSPCON2bits.RSEN == 1);
    SSPBUF = 0xA1;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.RCEN = 1;
    while(SSPSTATbits.BF == 0);
    dato_leido = SSPBUF;
    SSPCON2bits.ACKDT = 1;
    SSPCON2bits.ACKEN = 1;
    while(SSPCON2bits.ACKEN == 1);
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
    return(dato_leido);
}
```

```
void pcf8574_write(unsigned char dato){
    SSPCON2bits.SEN = 1;
    while(SSPCON2bits.SEN == 1);
    SSPBUF = 0x40;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPBUF = dato;
    while(SSPSTATbits.BF == 1);
    while(SSPSTATbits.R_nW == 1);
    SSPCON2bits.PEN = 1;
    while(SSPCON2bits.PEN == 1);
}
```

```
void main(void){
    lcd_init();
    mssp_conf();
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Semana 11 Lab",13);
    while(1){
        if (PORTBbits.RB6 == 1 && button1_stat == 0){
            //Función para grabar la EEPROM
            m24c01_escribir();
            button1_stat = 1;
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Grabando....",12);
        }
        else if(PORTBbits.RB6 == 0 && button1_stat == 1){
            button1_stat = 0;
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Grabado.....",12);
        }
        if (PORTBbits.RB7 == 1 && button2_stat == 0){
            //Función para leer la EEPROM
            POS_CURSOR(2,0);
            for (unsigned char y=0;y<12;y++){
                letra = m24c01_leer(20+y);
                ENVIA_CHAR(letra);
            }
            button2_stat = 1;
        }
        else if(PORTBbits.RB7 == 0 && button2_stat == 1){
            button2_stat = 0;
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("            ",12);
        }
        if (PORTBbits.RB3 == 1 && button3_stat == 0){
            //Función para leer la EEPROM
            pcf8574_write(0x55);
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("Dato PCF8574",12);
            button3_stat = 1;
        }
        else if(PORTBbits.RB3 == 0 && button3_stat == 1){
            button3_stat = 0;
            pcf8574_write(0xAA);
            POS_CURSOR(2,0);
            ESCRIBE_MENSAJE("            ",12);
        }
    }
}
```

43

Fin de la sesión

44