Microcontroladores

Semana 12-13

Profesor Kalun Lau

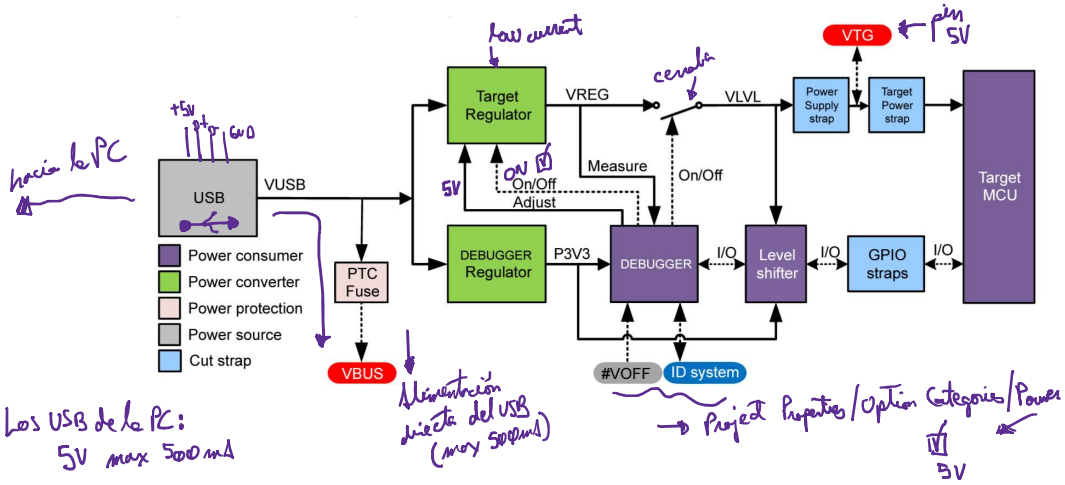
1

Preguntas previas

- Tuve un problema con errores al momento de programar el curiosity nano, me salía error aun cambiando de VBUS a VTG, abriendo y cerrando la aplicación del MPLABX, reiniciando la PC y conectando y desconectado el cable USB, también intenté cambiar a otro cable y nada, pero hoy arranqué todo y funciona normal. ¿Qué ha pasado?
  - Muy posible problema de firmware del curiosity nano

2

Preguntas previas (cont…)



3

Agenda Semana 12:

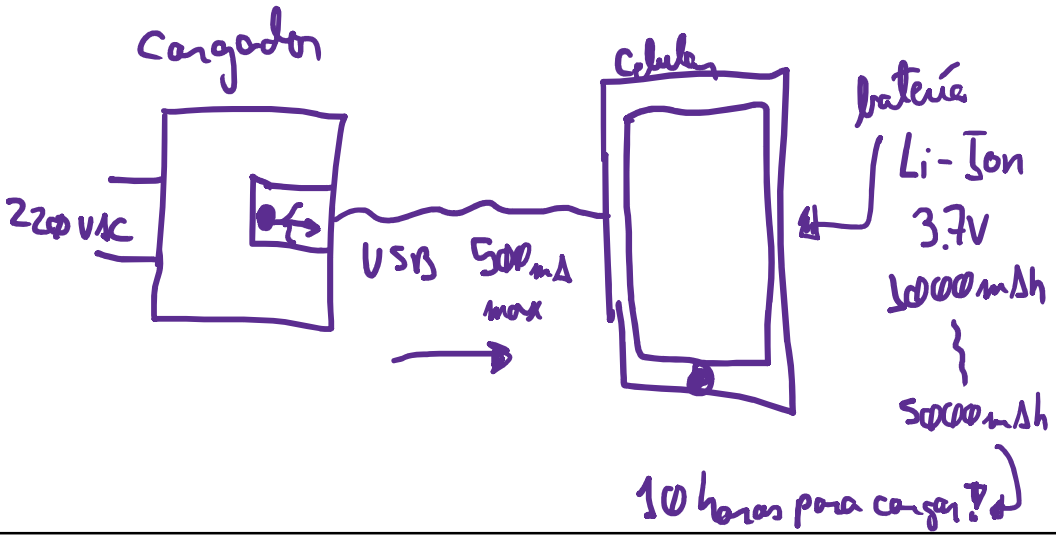
Comunicaciones seriales en microcontroladores

- Comunicación UART
  - Niveles lógicos RS232, TTL, RS485
  - Protocolo de comunicación
- El UART del PIC18F57Q43
  - Generador de baudios
  - Modo transmisión
  - Modo recepción

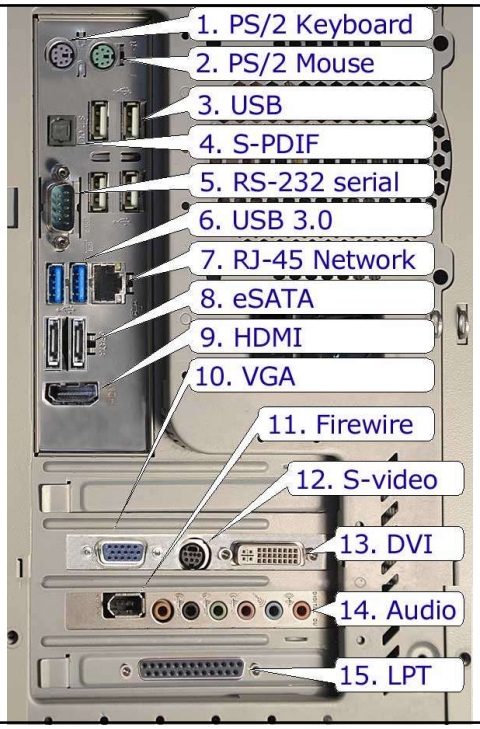
4

Caso de comunicación serial en la actualidad

- El cargador de tu celular



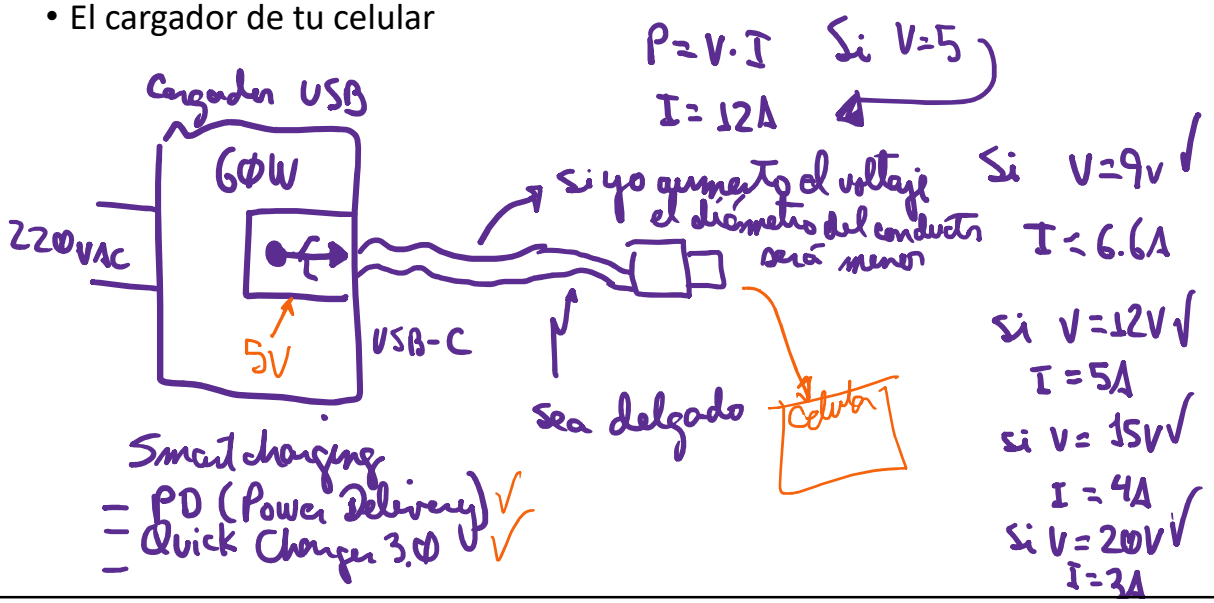
5



6

Caso de comunicación serial en la actualidad

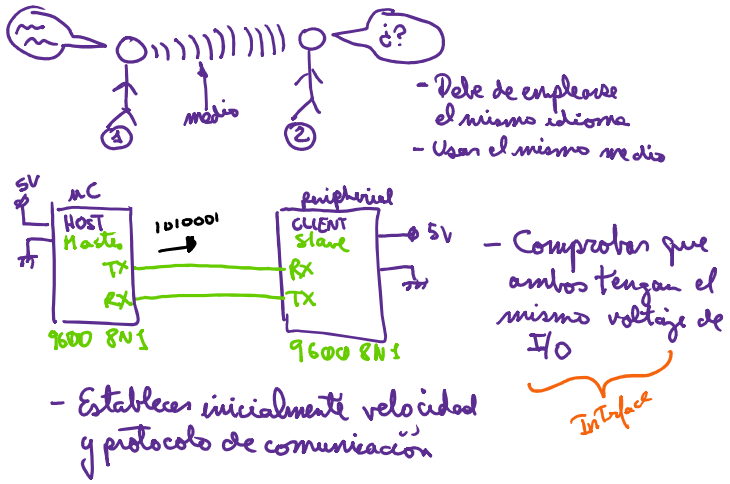
- El cargador de tu celular



7

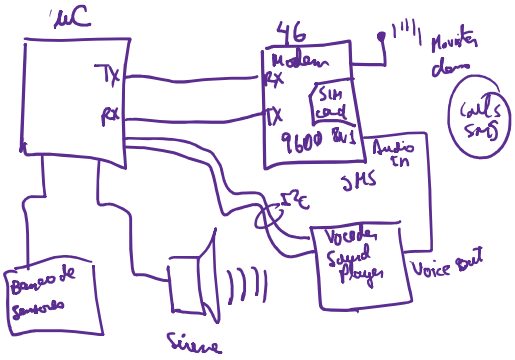
Comunicación asíncrona UART

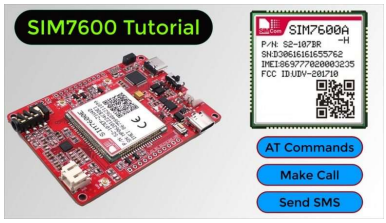
- Asíncrona: No hay una señal dedicada de reloj



8

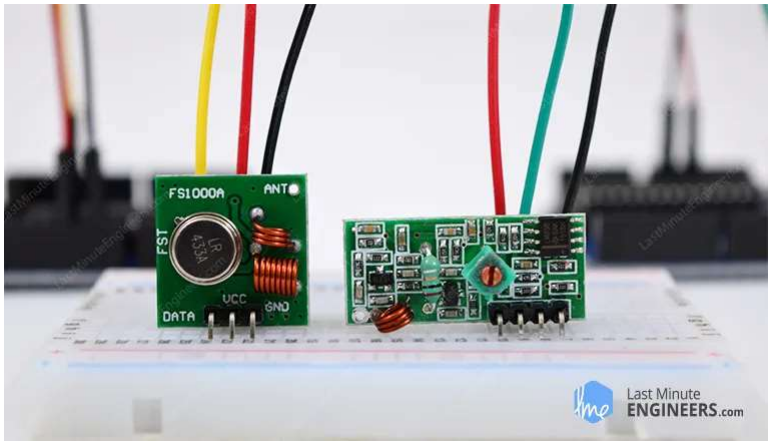
Diferentes periféricos externos al microcontrolador





9

Módulos de radio SAW 434MHz



434MHz ✓

SAW → Surface Acoustic Wave

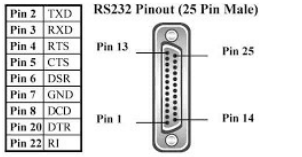
10

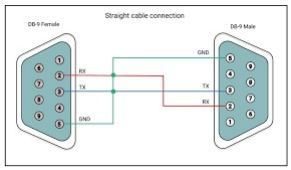
Sobre niveles de voltaje en comunicación UART

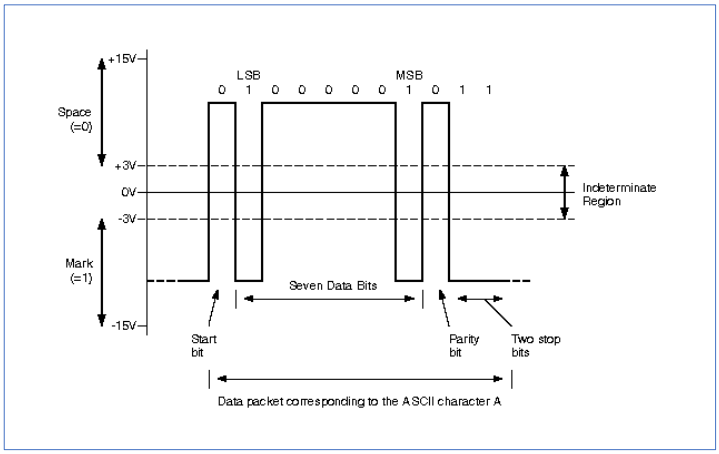
- RS232 (EIA/TIA 232) – Comunicación a distancias medianas (hasta 15 metros)

RS232 25 Pin

Pin 2	TXD
Pin 3	RXD
Pin 4	RTS
Pin 5	CTS
Pin 6	DSR
Pin 7	GND
Pin 8	DCD
Pin 20	DTR
Pin 22	RI

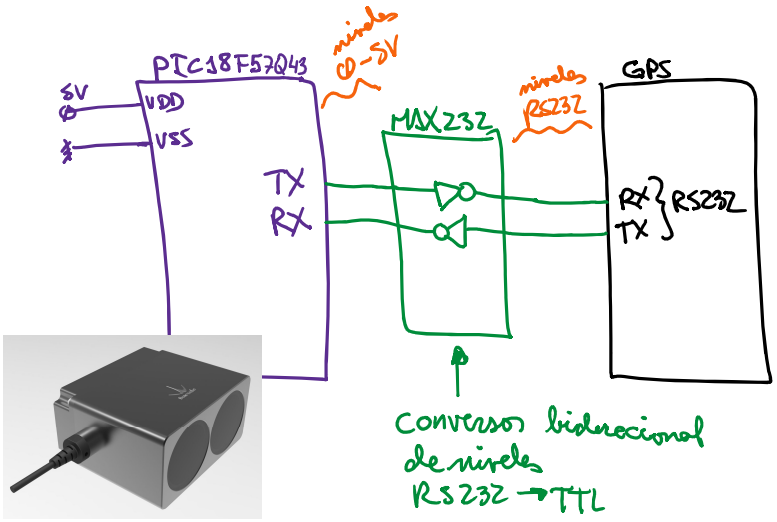


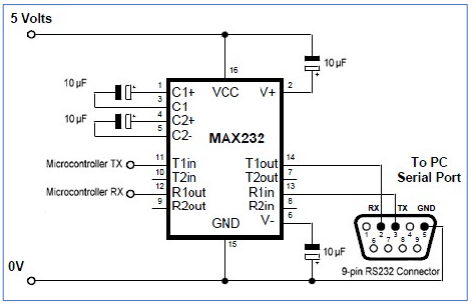




11

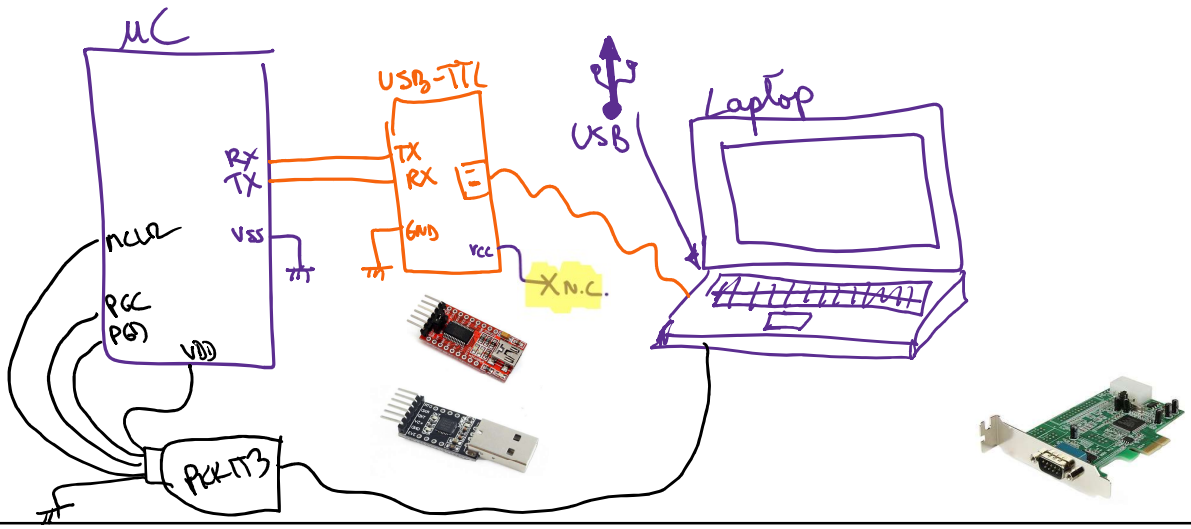
¿Cómo hago para conectar un dispositivo RS232 al PIC18F57Q43?





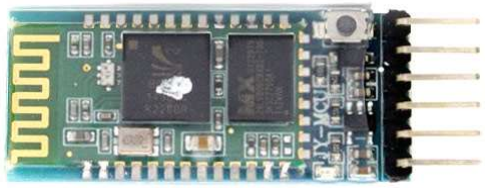
12

Cómo conectar un microcontrolador PIC hacia un computador mediante USB

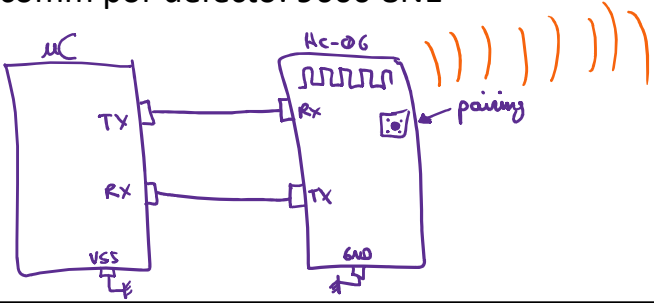


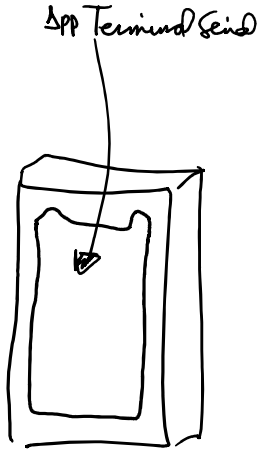
13

Módulo Bluetooth HC-06



- Interface de comunicación UART a niveles TTL (5V)
- Velocidad de comm por defecto: 9600 8N1



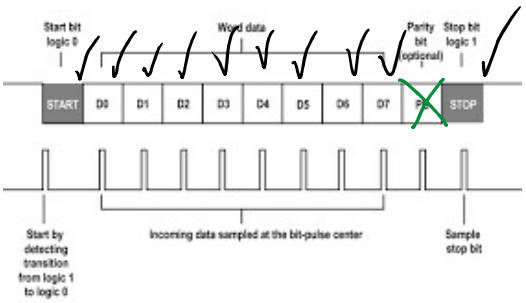


14

Cálculo de la velocidad en comunicación UART

- Formato completo es
  - Velocidad – Protocolo
  - Ej. 9600 - 8N1
- Protocolo común: 8N1
  - 8: El dato es de 8 bits
  - N: no paridad (O:paridad impar, E:paridad par)
  - 1: un bit de stop

LSB First



- En total se estan enviando 10 bits por cada dato de 8 bit

15

Cálculo de la velocidad en comunicación UART

- Si tengo 9600 (VTX) 8N1:

Tiempo de bit: $T_{bit} = \frac{1}{V_{TX}} = \frac{1}{9600} = 1.04 \times 10^{-4} s$

$= 0.1\ ms$

- Si para enviar un dato de 8 bits usamos 10 bits:

Tiempo para enviar un dato de 8bit = 1.04E-4 (10) = 1.04ms.

16

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de 100KByte por una canal de comm. serial a 9600 8N1?

Recordar que 1byte = 8bit, 1Kbyte = 1024bytes

1° Hallar cuantos bits se va a enviar:

100 x 1024 = 1024000 bytes

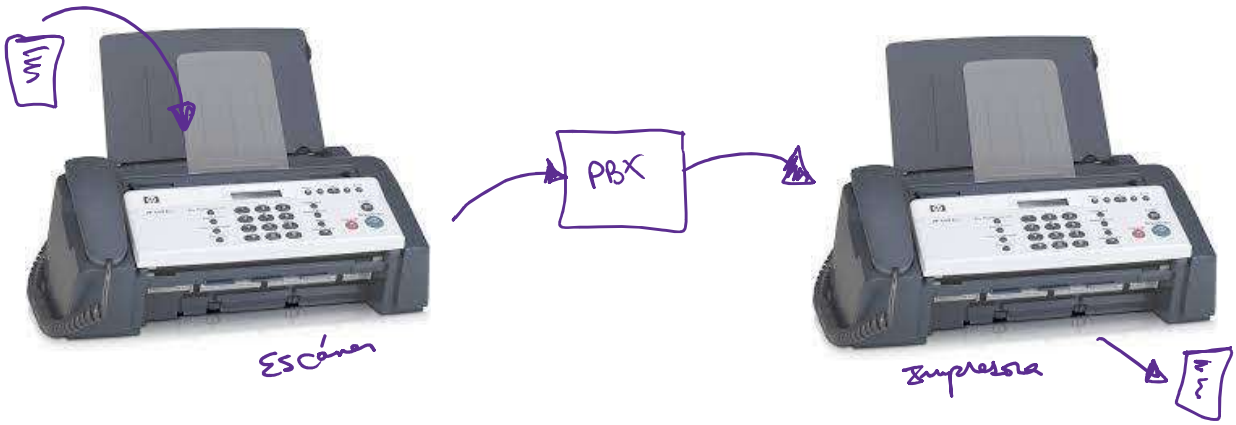
Cada byte representará 10 bits enviados ⇒ 10240000 bits enviados

$tiempobit_{9600} = 1.04 \times 10^{-4}$

tiempo requerido = 106.496 segundos

17

Recordando las máquinas FAX



- Transmitía el documento ingresado por línea telefónica a 9600 8N1

18

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de música de 4MByte por una canal de comm. serial a 57600 8N1?

5 minutos

19

Ejemplo de cálculo de cuánto tiempo demora en enviar datos vía comm. serial:

- ¿Cuánto demoro en enviar un archivo de música de 4MByte por una canal de comm. serial a 57600 8N1?

4MByte => 4K x 1024 = 4096 Kbytes

4096K x 1024 = 4194304 byts

Cant. bits a Transmitir: 4194304 x 10 = 41 943 040 bits

Tiempo de bit => $\frac{1}{57600}$ = 17.361 µs

Tiempo para transmitir los bits: 728.177s

12 minutos y 8.177 segundos

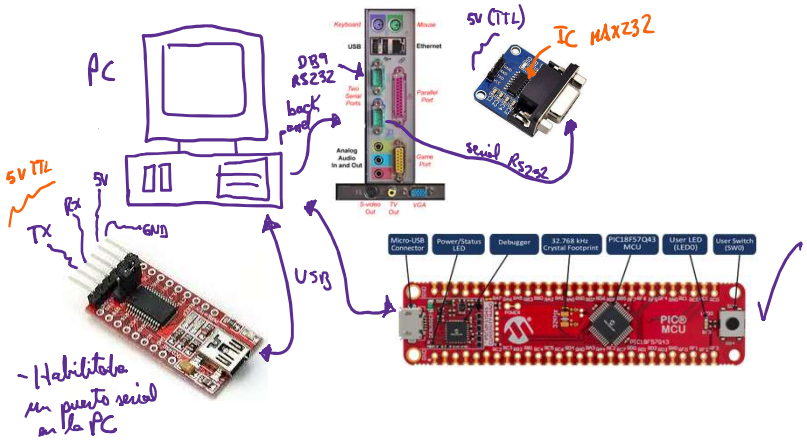
20

Sobre el UART en el PIC18F57Q43

- Revisar capítulo 34 de la hoja técnica del PIC18F57Q43
- Hay 5 módulos UART:
  - 3 full featured (U1, U3, U5)
  - 2 non full featured (U2, U4)

Comunicación serial UART con el PIC18F57Q43

- Curiosity Nano -> PC



Ejemplo: Comunicación serial UART

- Circuito de prueba

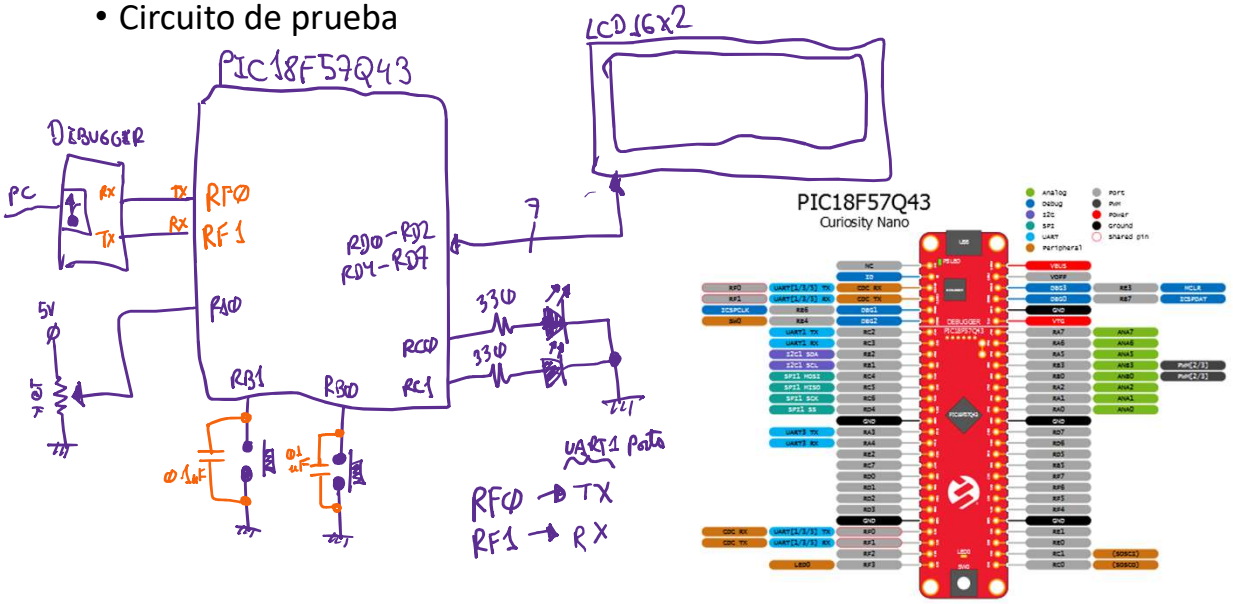
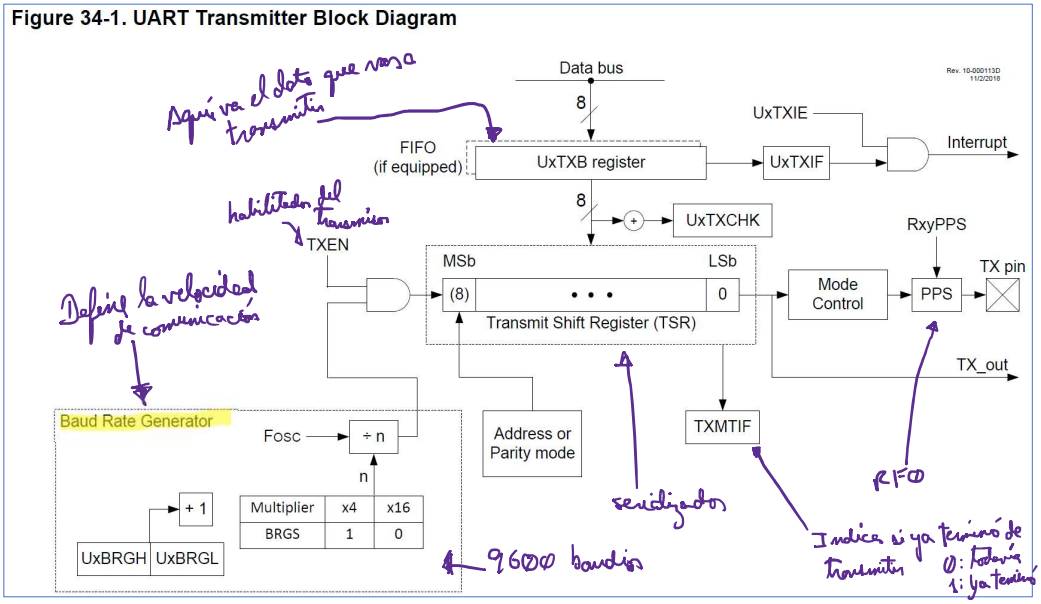


Diagrama de bloques del transmisor del UART

Figure 34-1. UART Transmitter Block Diagram



Fórmula para calcular el BRG:

The UART Baud Rate equals [Fosc*(1+(BRGS*3)]/[(16*(BRG+1))]

Tenemos que despejar BRG

Baud rate = 9600

Fosc = 32MHz

BRGS = 0

9600 = [32E6 (1 + 0x3)] / 16X(BRG+1)

0 → U1BRGH:U1BRGL → 207

BRGH:BRGL

BRG+1 = [32E6 (1)] / 16 x 9600

BRG = (32000000 / 153600) - 1 = 207.3333

Para 19200:

BRG+1 = [32E6 (1)] / 16 x 19200

BRG = (32000000 / 307200) - 1

BRG = 103

Para 38400:

BRG = (32000000 / 16 x 38400) - 1

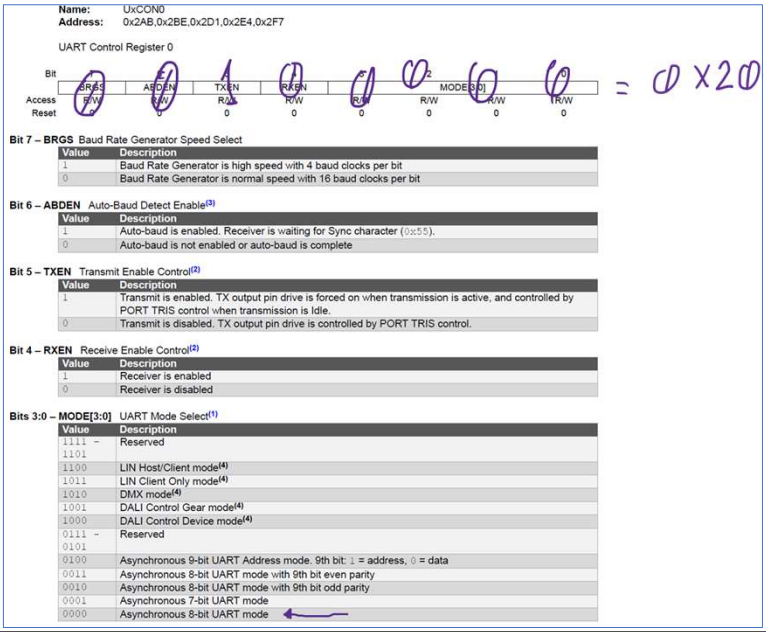
BRG = (32000000 / 614400) - 1

BRG = 51

25

Registro UxCON0

- Se configura el BRGS, los habilitadores de TX y RX y el modo de trabajo del UART1 (normalmente acíncrono y 8 bits de datos)



Name: UxCON0
Address: 0x2AB,0x2BE,0x2D1,0x2E4,0x2F7

UART Control Register 0

Bit	7	6	5	4	3	2	1	0
	BRGS	ABDEN	TXEN	RXEN		MODE[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

= 0 X 20

Bit 7 – BRGS Baud Rate Generator Speed Select

Value	Description
1	Baud Rate Generator is high speed with 4 baud clocks per bit
0	Baud Rate Generator is normal speed with 16 baud clocks per bit

Bit 6 – ABDEN Auto-Baud Detect Enable[3]

Value	Description
1	Auto-baud is enabled. Receiver is waiting for Sync character (0x55).
0	Auto-baud is not enabled or auto-baud is complete

Bit 5 – TXEN Transmit Enable Control[2]

Value	Description
1	Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is Idle.
0	Transmit is disabled. TX output pin drive is controlled by PORT TRIS control.

Bit 4 – RXEN Receive Enable Control[2]

Value	Description
1	Receiver is enabled
0	Receiver is disabled

Bits 3:0 – MODE[3:0] UART Mode Select[1]

Value	Description
1111 – 1101	Reserved
1100	LIN Host/Client mode[4]
1011	LIN Client Only mode[4]
1010	DMX mode[4]
1001	DALI Control Gear mode[4]
1000	DALI Control Device mode[4]
0111 – 0101	Reserved
0100	Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data
0011	Asynchronous 8-bit UART mode with 9th bit even parity
0010	Asynchronous 8-bit UART mode with 9th bit odd parity
0001	Asynchronous 7-bit UART mode
0000	Asynchronous 8-bit UART mode

26

Registro UxCON1

- Se configura el habilitador del UARTx

Name: UxCON1
Address: 0x2AC,0x2BF,0x2D2,0x2E5,0x2F8

UART Control Register 1

Bit	7	6	5	4	3	2	1	0
	ON			WUE	RXBIMD		BRKOVR	SENDB
Access	R/W			R/W/HC	R/W		R/W	R/W/HC
Reset	0			0	0		0	0

Bit 7 – ON Serial Port Enable

Value	Description
1	Serial port enabled
0	Serial port disabled (held in Reset)

Bit 4 – WUE Wake-Up Enable

Value	Description
1	Receiver is waiting for falling RX input edge which will set the UxIF bit. Cleared by hardware on wake-up event. Also requires the UxIE bit of PIEx to enable wake.
0	Receiver operates normally

Bit 3 – RXBIMD Receive Break Interrupt Mode Select

Value	Description
1	Set RXBKIF immediately when RX in has been low for the minimum Break time
0	Set RXBKIF on rising RX input after RX in has been low for the minimum Break time

Bit 1 – BRKOVR Send Break Software Override

Value	Description
1	TX output is forced to non-Idle state
0	TX output is driven by transmit shift register

Bit 0 – SENDB Send Break Control(1)

Value	Description
1	Output Break upon UxTXB write. Written byte follows Break. Bit is cleared by hardware.
0	Break transmission completed or disabled

27

Registro U1CON2

- Lo vamos a dejar con los valores por defecto (0x00)

28

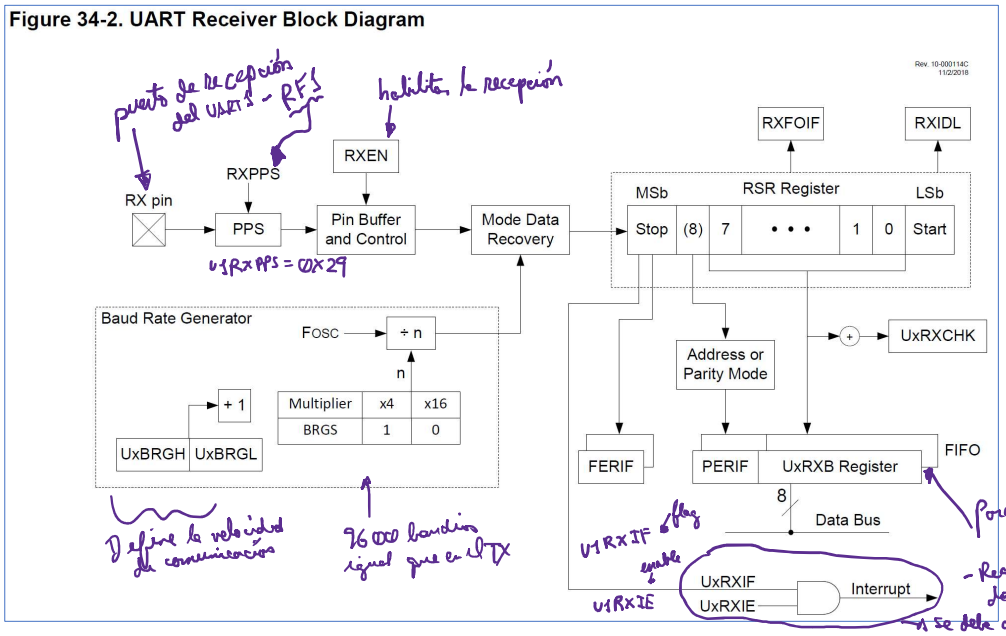
Configuración del UART1 para transmitir un dato:

- Primero: Puertos de E/S:
  - En el Curiosity Nano RF0 es el TX, RF1 es el RX
- Segundo: Establecer velocidad con U1BRG
- Tercero: Configurar registros U1CONx:
  - U1CON0: BRGS (normal/high), TX enable, RX enable, UART mode
  - U1CON1: ON (serial port enable)
  - U1CON2: déjalo con los valores por defecto
- Cuarto: El PPS para TX
  - En el Curiosity Nano: RF0PPS = 0x20;
- Quinto: Para transmitir un dato de 8 bits
  - Colocar el dato en U1TXB
  - Esperar a que se termine de transmitir (U1ERRIR bit TXMTIF = 1)

29

Diagrama de bloques del receptor del UART

Figure 34-2. UART Receiver Block Diagram



30

Configuración del PPS de entrada del UART1 RX con el puerto RF1

- Esta por defecto el RC7 para UART1 RX

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	Available Input Port											
				28-Pin Devices			40-Pin Devices				48-Pin Devices				
SPI1 Client Select	SPI1SSPPS	RA5	'b000 101	A	—	C	A	—	—	D	—	A	—	—	D	—	—
SPI2 Clock	SPI2SCKPPS	RB3	'b001 011	—	B	C	—	B	—	D	—	—	B	—	D	—	—
SPI2 Data	SPI2SDIPPS	RB2	'b001 010	—	B	C	—	B	—	D	—	—	B	—	D	—	—
SPI2 Client Select	SPI2SSPPS	RA4	'b000 100	A	—	C	A	—	—	D	—	A	—	—	D	—	—
I2C1 Clock	I2C1SCLPPS(1)	RC3	'b010 011	—	B	C	—	B	C	—	—	—	B	C	—	—	—
I2C1 Data	I2C1SDAPPS(1)	RC4	'b010 100	—	B	C	—	B	C	—	—	—	B	C	—	—	—
UART1 Receive	U1RXPPS	RC7	'b010 111	—	B	C	—	B	C	—	—	—	—	C	—	—	F
UART1 Clear to Send	U1CTSPPS	RC6	'b010 110	—	B	C	—	B	C	—	—	—	—	C	—	—	F
UART2 Receive	U2RXPPS	RB7	'b001 111	—	B	C	—	B	—	D	—	—	B	—	D	—	—
UART2 Clear to Send	U2CTSPPS	RB6	'b001 110	—	B	C	—	B	—	D	—	—	B	—	D	—	—
UART3 Receive	U3RXPPS	RA7	'b000 111	A	B	—	A	B	—	—	—	A	—	—	—	—	F
UART3 Clear to Send	U3CTSPPS	RA6	'b000 110	A	B	—	A	B	—	—	—	A	—	—	—	—	F
UART4 Receive	U4RXPPS	RB5	'b001 101	—	B	C	—	B	—	D	—	—	B	—	D	—	—
UART4 Clear to Send	U4CTSPPS	RB4	'b001 100	—	B	C	—	B	—	D	—	—	B	—	D	—	—
UART5 Receive	U5RXPPS	RA5	'b000 101	A	—	C	A	—	C	—	—	A	—	—	—	—	F
UART5 Clear to Send	U5CTSPPS	RA4	'b000 100	A	—	C	A	—	C	—	—	A	—	—	—	—	F

21.8.1 xxxPPS

Name: xxxPPS U1RXPPS = 0x29

Peripheral Input Selection Register

Bit	7	6	5	4	3	2	1	0
	—	—	PORT[2:0]			PIN[2:0]		
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			m	m	m	m	m	m

Bits 5:3 – PORT[2:0] Peripheral Input PORT Selection(1)
See the PPS Input Selection Table for the list of available Ports and default pin locations.

PORT	Selection
101	PORTF
100	PORTE
011	PORTD
010	PORTC
001	PORTB
000	PORTA

Reset States: POR = mmm
All other Resets = uuu

Bits 2:0 – PIN[2:0] Peripheral Input PORT Pin Selection(2)
Reset States: POR = mmm
All other Resets = uuu

Value	Description
111	Peripheral input is from PORTx Pin 7 (Rx7)
110	Peripheral input is from PORTx Pin 6 (Rx6)
101	Peripheral input is from PORTx Pin 5 (Rx5)
100	Peripheral input is from PORTx Pin 4 (Rx4)
011	Peripheral input is from PORTx Pin 3 (Rx3)
010	Peripheral input is from PORTx Pin 2 (Rx2)
001	Peripheral input is from PORTx Pin 1 (Rx1)
000	Peripheral input is from PORTx Pin 0 (Rx0)

31

Registro U1CON0

- Se configura el BRGS, los habilitadores de TX y RX y el modo de trabajo del UART1

Name: UxCON0
Address: 0x2AB,0x2BE,0x2D1,0x2E4,0x2F7

U1CON = 0x30

UART Control Register 0

Bit	7	6	5	4	3	2	1	0
	BRGS	ABDEN	TXEN	RXEN	MODE[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

Bit 7 – BRGS Baud Rate Generator Speed Select

Value	Description
1	Baud Rate Generator is high speed with 4 baud clocks per bit
0	Baud Rate Generator is normal speed with 16 baud clocks per bit

Bit 6 – ABDEN Auto-Baud Detect Enable(3)

Value	Description
1	Auto-baud is enabled. Receiver is waiting for Sync character (0x55).
0	Auto-baud is not enabled or auto-baud is complete

Bit 5 – TXEN Transmit Enable Control(2)

Value	Description
1	Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is Idle.
0	Transmit is disabled. TX output pin drive is controlled by PORT TRIS control.

Bit 4 – RXEN Receive Enable Control(2)

Value	Description
1	Receiver is enabled
0	Receiver is disabled

Bits 3:0 – MODE[3:0] UART Mode Select(1)

Value	Description
1111 – 1101	Reserved
1100	LIN Host/Client mode(4)
1011	LIN Client Only mode(4)
1010	DMX mode(4)
1001	DALI Control Gear mode(4)
1000	DALI Control Device mode(4)
0111 – 0101	Reserved
0100	Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data
0011	Asynchronous 8-bit UART mode with 9th bit even parity
0010	Asynchronous 8-bit UART mode with 9th bit odd parity
0001	Asynchronous 7-bit UART mode
0000	Asynchronous 8-bit UART mode

32

Ubicación de la interrupción de recepción del UART1:

0x04A1	PIE3	7:0	TMR0IE	CCP1IE	TMR1GIE	TMR1IE	TMR2IE	SPI1IE	SPI1TXIE	SPI1RXIE
0x04A2	PIE4	7:0	PWM1IE	PWM1PIE			U1IE	U1EIE	U1TXIE	U1RXIE
0x04A3	PIE5	7:0	PWM2IE	PWM2PIE	TMR3GIE	TMR3IE		SPI2IE	SPI2TXIE	SPI2RXIE
0x04A4	PIE6	7:0	DMA2AIE	DMA2ORIE	DMA2DCNTIE	DMA2SCNTIE	NCO1IE	CWG1IE	CLC2IE	INT1IE
0x04A5	PIE7	7:0	PWM3IE	PWM3PIE	CLC3IE		I2C1EIE	I2C1IE	I2C1TXIE	I2C1RXIE
0x04A6	PIE8	7:0	SCANIE	CCP2IE	TMR5GIE	TMR5IE	U2IE	U2EIE	U2TXIE	U2RXIE
0x04A7	PIE9	7:0			CLC4IE		U3IE	U3EIE	U3TXIE	U3RXIE
0x04A8	PIE10	7:0	DMA3AIE	DMA3ORIE	DMA3DCNTIE	DMA3SCNTIE	NCO2IE	CWG2IE	CLC5IE	INT2IE
0x04A9	PIE11	7:0	DMA4AIE	DMA4ORIE	DMA4DCNTIE	DMA4SCNTIE	TMR4IE	CWG3IE	CLC6IE	CCP3IE
0x04AA	PIE12	7:0	DMA5AIE	DMA5ORIE	DMA5DCNTIE	DMA5SCNTIE	U4IE	U4EIE	U4TXIE	U4RXIE
0x04AB	PIE13	7:0	DMA6AIE	DMA6ORIE	DMA6DCNTIE	DMA6SCNTIE	U5IE	U5EIE	U5TXIE	U5RXIE
0x04AC	PIE14	7:0					NCO3IE	CM2IE	CLC7IE	
0x04AD	PIE15	7:0					TMR6IE	CRCIE	CLC8IE	NVMIE
0x04AE	PIR0	7:0	IOCIF		CLC1IF		CSWIF	OSFIF	HLVDIF	SWIF
0x04AF	PIR1	7:0	SMT1PWAIF	SMT1PRAIF	SMT1IF	CM1IF	ACTIF	ADIF	ZCDIF	INT0IF
0x04B0	PIR2	7:0	DMA1AIF	DMA1ORIF	DMA1DCNTIF	DMA1SCNTIF				ADTIF
0x04B1	PIR3	7:0	TMR0IF	CCP1IF	TMR1GIF	TMR1IF	TMR2IF	SPI1IF	SPI1TXIF	SPI1RXIF
0x04B2	PIR4	7:0	PWM1IF	PWM1PIF			U1IF	U1EIF	U1TXIF	U1RXIF
0x04B3	PIR5	7:0	PWM2IF	PWM2PIF	TMR3GIF	TMR3IF		SPI2IF	SPI2TXIF	SPI2RXIF

Configuración adicional del UART1 para recepcionar un dato:

- Primero: Estas configuraciones son adicionales a lo que se establecieron en la transmisión
- Segundo: Cerciorarse que el RX esté enabled en el U1CON0
- Tercero: Habilitar la interrupción de la recepción del UART1:
  - Habilitador U1RXIE se encuentra en PIE4 y bandera U1RXIF se encuentra en PIR4
  - No olvidar habilitar el GIE que esta en INTCON0.
- Cuarto: El PPS para RX
  - En el Curiosity Nano: U1RXPPS = 0x29
- Quinto: Para recibir un dato de 8 bits
  - Definir la función de interrupción de la recepción del UART1

```
void __interrupt(irq(IRQ_U1RX)) U1RX_ISR(void){
    PIR4bits.U1RXIF = 0;        //bajamos bandera de RX de UART1
```

Agenda Semana 13:

Comunicaciones seriales en microcontroladores

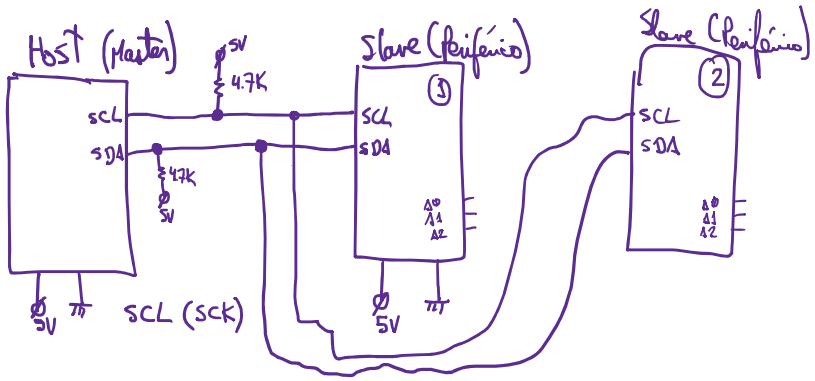
- Comunicación I2C
  - Interface eléctrica
  - Protocolo de comunicación
  - Periféricos externos con capacidad I2C
- El I2C del PIC18F57Q43
  - Configuración

Si tienen alguna consulta, levantan la mano para poder atenderlos.

35

Comunicación I2C

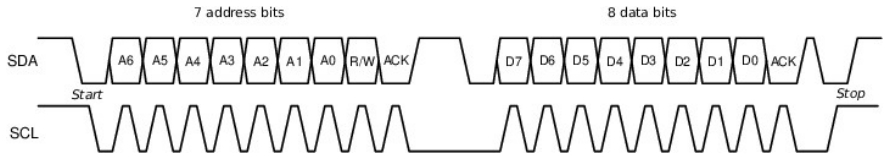
- Comunicación desarrollada para la comunicación entre circuitos integrados. Phillips Semiconductor (NXP)
- Comunicación serial síncrona (presencia de señal de reloj)
- I2C – I2S -> Sonido
- Podemos construir una red de dispositivos (topología bus)



36

Comunicación I2C

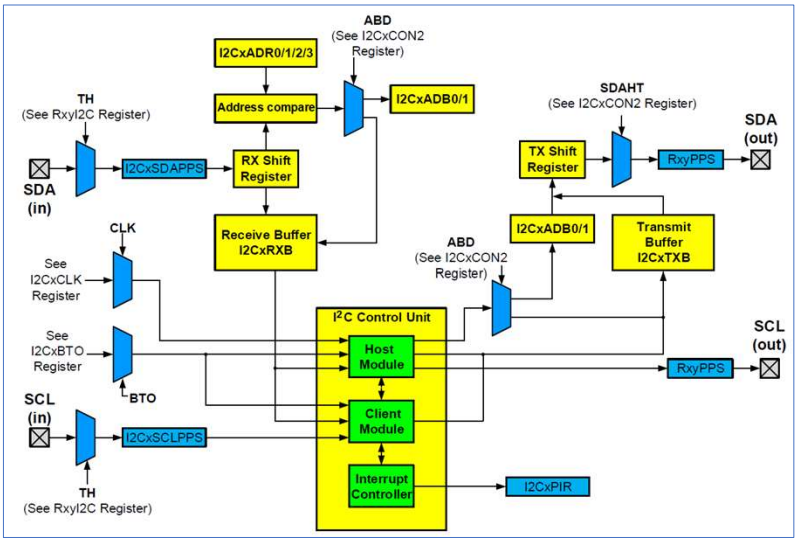
- Direcciones de 7 bits ó 10 bits
- Frecuencia de reloj de 100Khz (legacy o normal), 400Khz (fast mode), 3.4Hz (high-speed mode), 5MHz (ultra-fast mode).
- Puede funcionar en modo HOST (master) o en modo CLIENT (slave)
- En el I2C tenemos condiciones (comandos u órdenes):
  - Condición **START**: Inicio de la comunicación I2C.
  - Condición **RESTART**: Resetear la línea de comunicación, generalmente se emplea para cambiar entre escritura y lectura dentro de un evento de comunicación de un periférico.
  - Condición **STOP**: Término de la comunicación I2C.
  - **ACK** (Aknowledge): Evento de confirmación.
  - **NOACK** (No Acknowledge): Evento de no confirmación.
- Se debe de revisar la hoja técnica del periférico I2C para ver la trama I2C que necesita para comunicarse



37

El módulo I2C del PIC18F57Q43

- Revisar capítulo 36 de la hoja técnica:



38

El módulo I2C del PIC18F57Q43

- Para obtener 100KHz de frecuencia de trabajo (modo normal):
  - Se emplea MFINTOSC ya que al ser de 500KHz y configurando FME (Fast Mode Enabled) a cero se obtienen los 100KHz:

Equation 36-1. SCL Frequency (FME = 0)

$$f_{SCL} = \frac{f_{I2CxCLK}}{5}$$

Example:

- I2CxCLK: MFINTOSC (500 kHz)
- FME: FME = 0

39

El módulo I2C del PIC18F57Q43

- Escritura de datos

Figure 36-26. 7-Bit Host Write Diagram

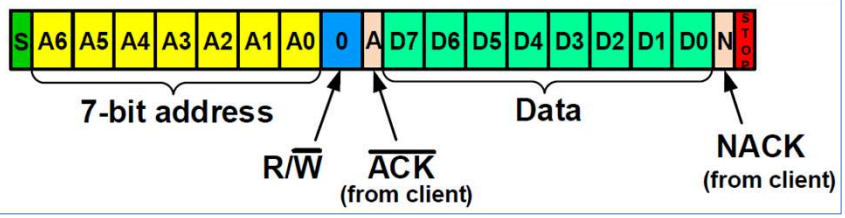
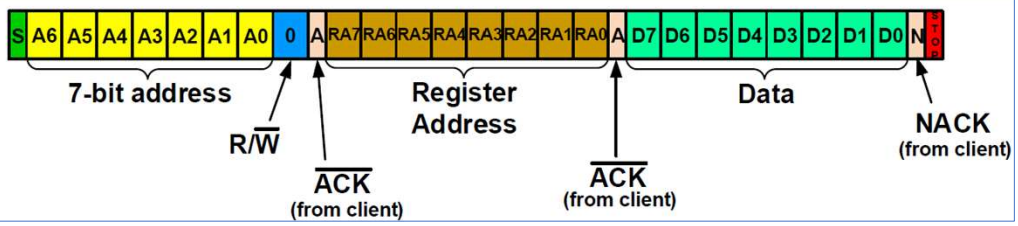


Figure 36-27. 7-Bit Host Write Diagram (To a Specific Memory/Register Location)



40

El módulo I2C del PIC18F57Q43

- Procedimiento para hacer una recepción:

36.4.2.7.2 Host Reception (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
   a. When ABD is clear (ABD = 0), the address buffer, I2CxADB1, is enabled. In this case, the 7-bit client address and R/$\overline{W}$ bit are loaded into I2CxADB1, with the R/$\overline{W}$ bit set (R/$\overline{W}$ = 1). The number of expected received data bytes are loaded into I2CxCNT. After these registers are loaded, software must set the Start (S) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
   b. When ABD is set (ABD = 1), the address buffer is disabled. In this case, the number of expected received data bytes are loaded into I2CxCNT, and the client's 7-bit address and R/$\overline{W}$ bit are loaded into I2CxTXB. A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (MMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2CxIF is also set.
3. Host hardware transmits the 7-bit client address and R/$\overline{W}$ bit.
4. Host hardware samples SCL to determine if the client is stretching the clock, and continues to sample SCL until the line is sampled high.
5. Host hardware transmits the 9th clock pulse, and receives the $\overline{ACK}$/NACK response from the client.
   If an $\overline{ACK}$ is received, host hardware receives the first seven bits of the data byte into the receive shift register.
   If a NACK is received, hardware sets the NACK Detect Interrupt Flag (NACKIF), and:
   a. ABD = 0: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
   b. ABD = 1: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored.
   If the NACK Detect Interrupt Enable (NACKIE) is also set, hardware sets the generic I2CxEIF bit.
6. If previous data remains in the I²C Receive Buffer (I2CxRXB) when the first seven bits of the new byte are received into the receive shift register (RXBF = 1), the MDR bit is set (MDR = 1), and the clock is stretched after the 7th falling edge of SCL. This allows the host time to read I2CxRXB, which clears the RXBF bit, and prevents receive buffer overflows. Once RXBF is clear, hardware releases SCL.
7. The host clocks in the 8th bit of the data byte into the receive shift register, then transfers the full byte into I2CxRXB. Host hardware sets the I²C Receive Interrupt Flag (I2CxRXIF) and RXBF, and if the I²C Receive Interrupt Enable (I2CxRXIE) is set, the generic I2CxIF is also set. Finally, I2CxCNT is decremented by one.
8. Host hardware checks I2CxCNT for a zero value.
   If I2CxCNT is non-zero (I2CxCNT != 0), hardware transmits the value of the Acknowledge Data (ACKDT) bit as the acknowledgement response to the client. It is up to user software to properly configure ACKDT. In most cases, ACKDT should be clear (ACKDT = 0), which indicates an $\overline{ACK}$ response.
   If I2CxCNT is zero (I2CxCNT = 0), hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the acknowledgement response to the client. CNTIF is set, and host hardware either issues a Stop condition or a Restart condition. It is up to user software to properly configure ACKCNT. In most cases, ACKCNT should be set (ACKCNT = 1), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled bus condition.
9. Host hardware receives the first seven bits of the next data byte into the receive shift register.
10. Repeat Steps 6 – 9 until all expected bytes have been received.

41

El módulo I2C del PIC18F57Q43

- Lectura de datos:

Figure 36-23. 7-Bit Host Read Diagram

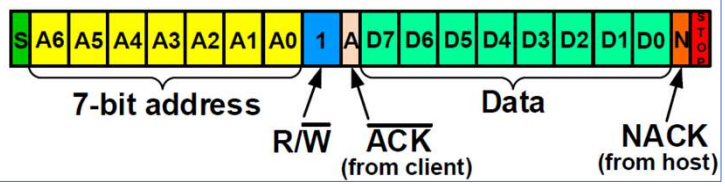
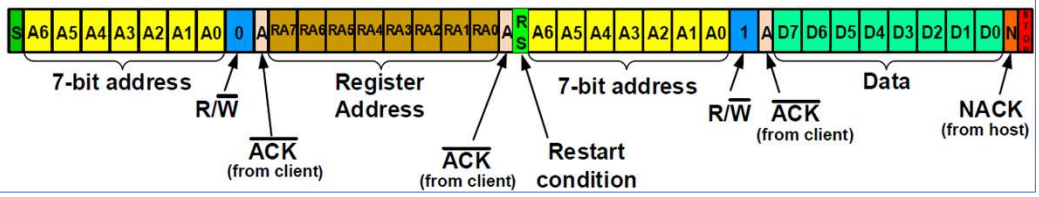


Figure 36-24. 7-Bit Host Read Diagram (From a Specific Memory/Register Location)



42

El módulo I2C del PIC18F57Q43

- Procedimiento para hacer una transmisión:

36.4.2.7.1 Host Transmission (7-Bit Addressing Mode)

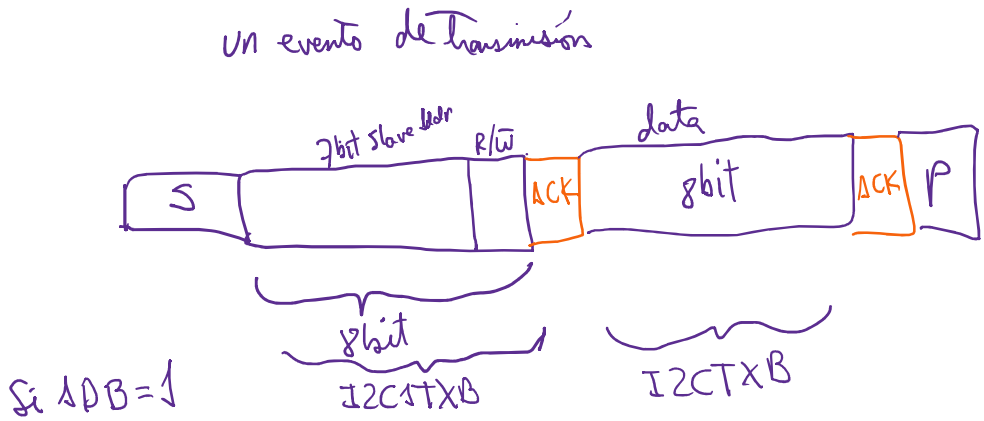
The following section describes the sequence of events that occur when the module is transmitting data in 7-bit Addressing mode:

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
   a. When ABD is clear (ABD = 0), the address buffer, I2CxADB1, is enabled. In this case, the 7-bit client address and R/$\overline{W}$ bit are loaded into I2CxADB1, with the R/$\overline{W}$ bit clear (R/$\overline{W}$ = 0). The number of data bytes are loaded into I2CxCNT and the first data byte is loaded into I2CxTXB. After these registers are loaded, software must set the Start (S) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
   b. When ABD is set (ABD = 1), the address buffer is disabled. In this case, the number of data bytes are loaded into I2CxCNT, and the client's 7-bit address and R/$\overline{W}$ bit are loaded into I2CxTXB. A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (MMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2CxIF is also set.
3. Host hardware transmits the 7-bit client address and R/$\overline{W}$ bit.
4. If upon the 8th falling edge of SCL, I2CxTXB is empty (Transmit Buffer Empty Status (TXBE) = 1), I2CxCNT is non-zero (I2CxCNT != 0), and the Clock Stretching Disable (CSD) bit is clear (CSD = 0):
   – The I^2C Transmit Interrupt Flag (I2CxTXIF) is set. If the I^2C Transmit Interrupt Enable (I2CxTXIE) bit is also set, the generic I2CxIF is also set.
   – The Host Data Request (MDR) bit is set, and the clock is stretched, allowing time for software to load I2CxTXB with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.
5. Hardware transmits the 9th clock pulse and waits for an $\overline{ACK}$/NACK response from the client. If the host receives an $\overline{ACK}$, module hardware transfers the data from I2CxTXB into the transmit shift register, and I2CxCNT is decremented by one. If the host receives a NACK, hardware will attempt to issue a Stop condition. If the clock is currently being stretched by a client, the host must wait until the bus is free before issuing the Stop.
6. Host hardware checks I2CxCNT for a zero value. If I2CxCNT is zero:
   a. If ABD is clear (ABD = 0), host hardware issues a Stop condition, or sets MDR if the Restart Enable (RSEN) bit is set and waits for software to set the Start bit to issue a Restart condition. CNTIF is set.
   b. If ABD is set (ABD = 1), host hardware issues a Stop condition, or sets MDR if RSEN is set and waits for software to load I2CxTXB with a new client address. CNTIF is set.
7. Host hardware transmits the data byte.
8. If upon the 8th falling edge of SCL I2CxTXB is empty (TXBE = 1), I2CxCNT is non-zero (I2CxCNT != 0), and CSD is clear (CSD = 0):
   – I2CxTXIF is set. If the I2CxTXIE bit is also set, the generic I2CxIF is also set.
   – The MDR bit is set and the clock is stretched, allowing time for software to load I2CxTXB with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.

   If TXBE is set (TXBE = 1) and I2CxCNT is zero (I2CxCNT = 0):
   – I2CxTXIF is NOT set.
   – CNTIF is set.
   – Host hardware issues a Stop condition, setting PCIF.
9. Repeat Steps 5 – 8 until all data has been transmitted.

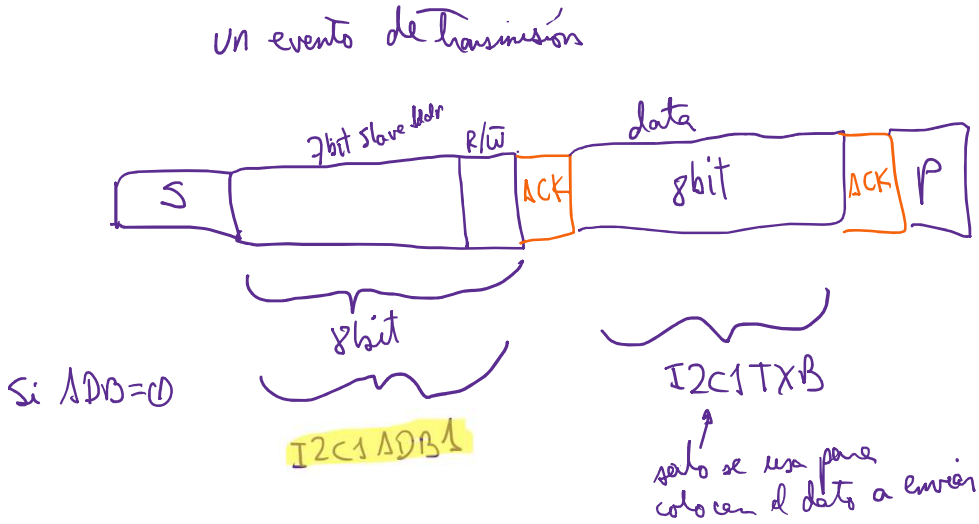
43

Transmisión de datos con opción ADB=1



44

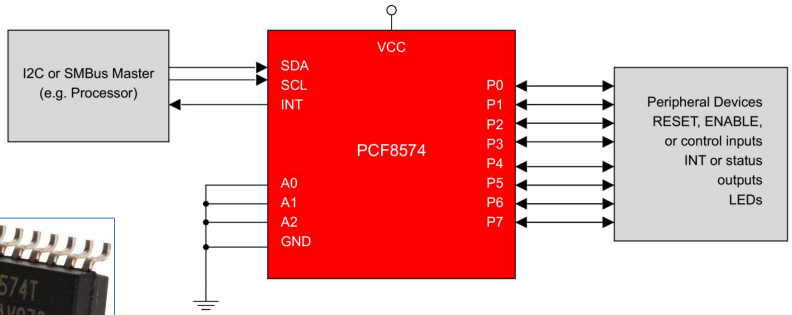
Transmisión de datos con opción ADB=0

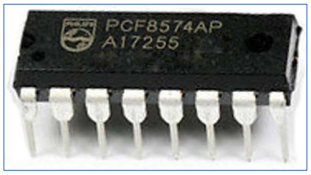


45

Caso: PCF8574

- Expansor de 8 bits de E/S I2C
- Hasta 8 dispositivos PCF8574 en un bus I2C





46

PCF8574: Modo escritura

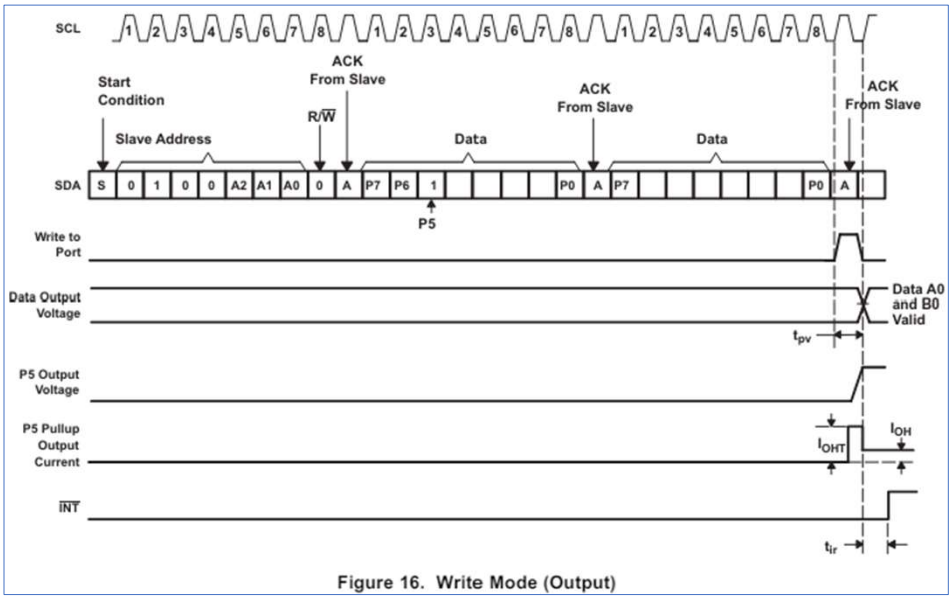
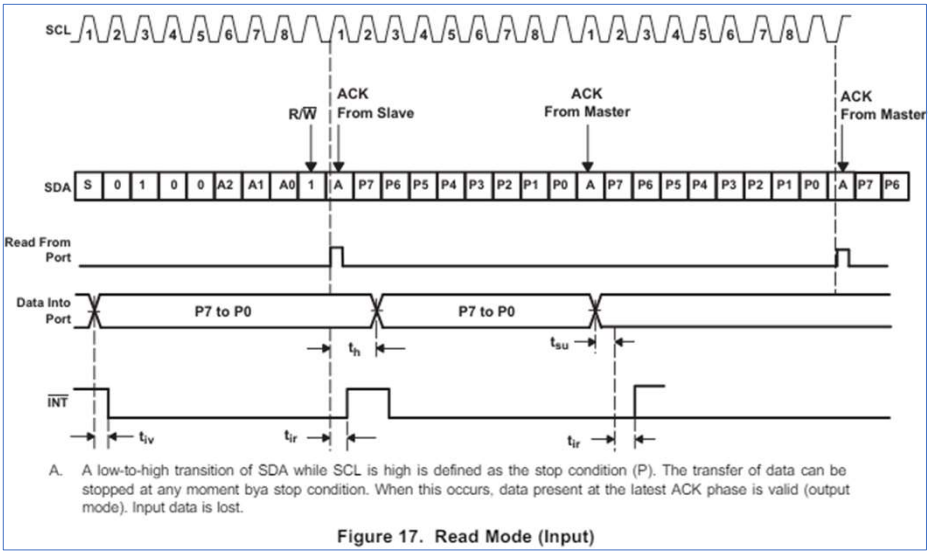


Figure 16. Write Mode (Output)

47

PCF8574: Modo lectura



A. A low-to-high transition of SDA while SCL is high is defined as the stop condition (P). The transfer of data can be stopped at any moment bya stop condition. When this occurs, data present at the latest ACK phase is valid (output mode). Input data is lost.

Figure 17. Read Mode (Input)

48

Proceso de envío de datos al PCF8574 (en I2C)

<S> <slave address + write> <**ACK**> <data out> <**ACK**> <data out> <**ACK**> ...
<data out> <**ACK**> <P>

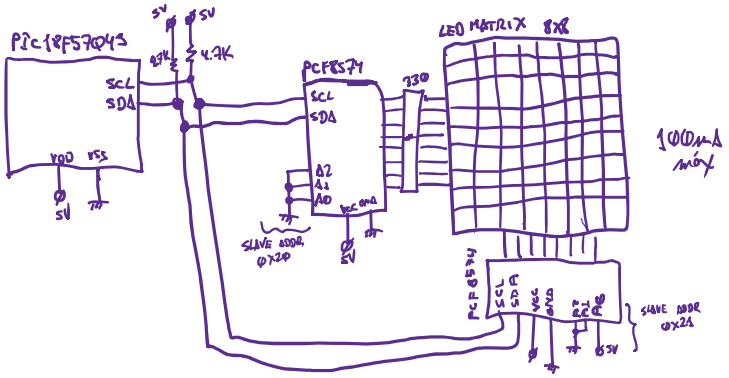
Remark: Bold type = generated by slave device.

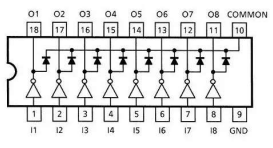
- Primero enviar condición de START
- Enviar la palabra de control (dirección de esclavo junto con la acción de escritura R/~W = 0)
- Esperar señal de ACK de parte del esclavo
- El host envía el dato de 8 bits
- Esperar señal de ACK de parte del esclavo
- El host envía condición de STOP

49

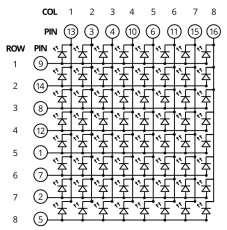
Caso: PCF8574

- Expansor de 8 bits de E/S I2C
- Hasta 8 dispositivos PCF8574 en un bus I2C



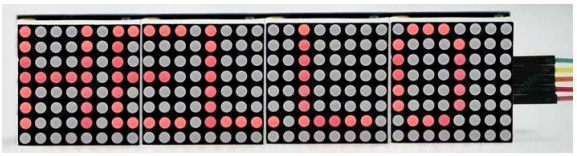


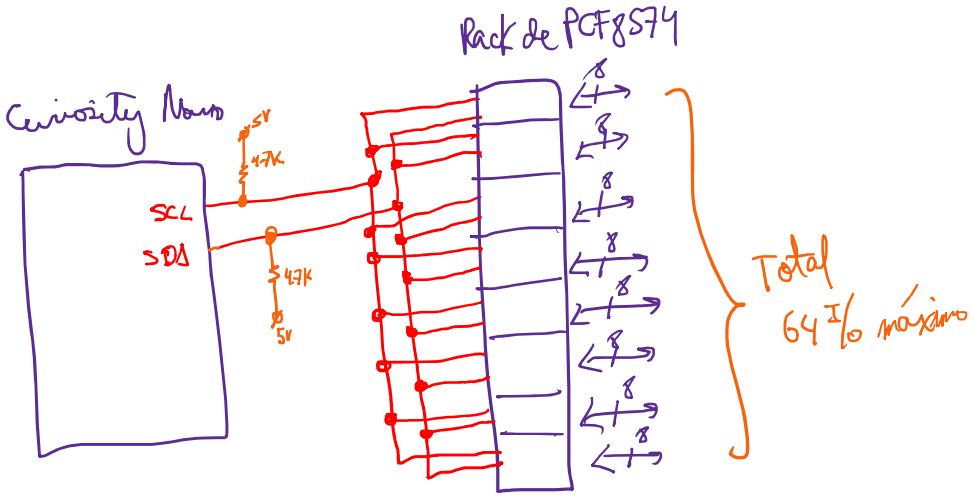
ULN2803 - ULN2804



50

Caso: PCF8574

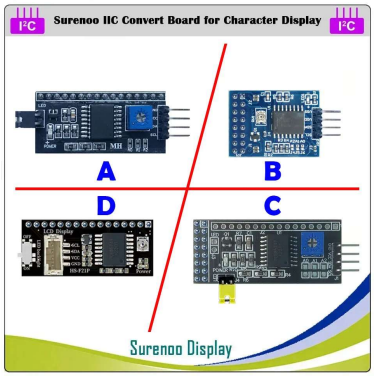


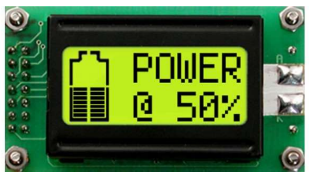


51

Caso: PCF8574

- Tarjeta I2C para display LCD 16x2 HD44780
- Posee un conector de 4 pines (Vcc, Gnd, SDA y SCL)
- Utiliza un PCF8574!





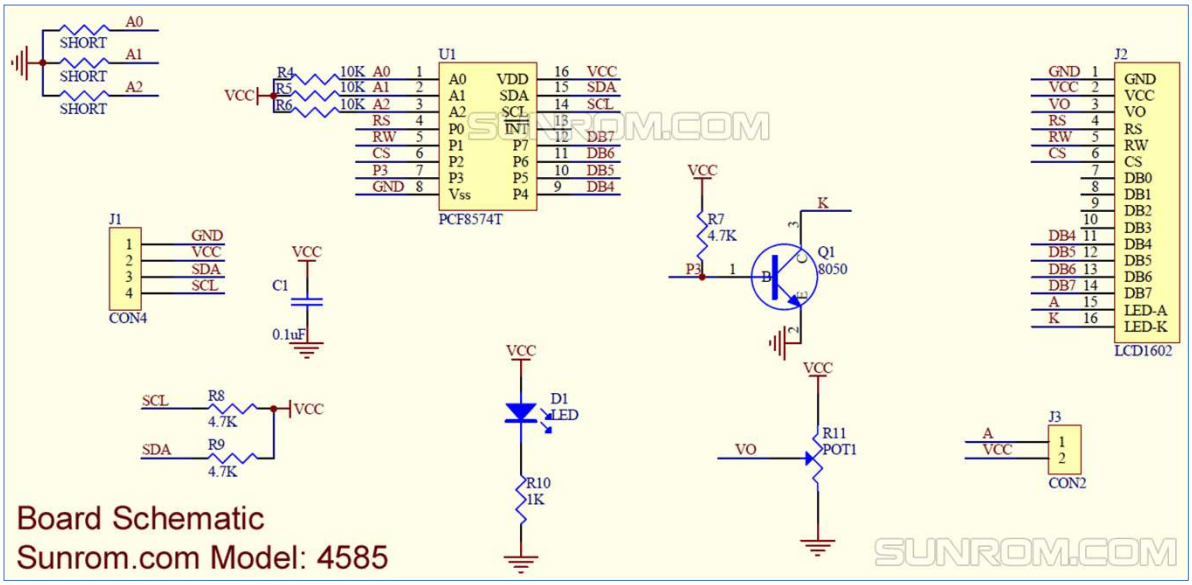
52

Comunicación I2C: Interface a IC PCF8574

- Por defecto la tarjeta tiene dirección de esclavo 0x27

53

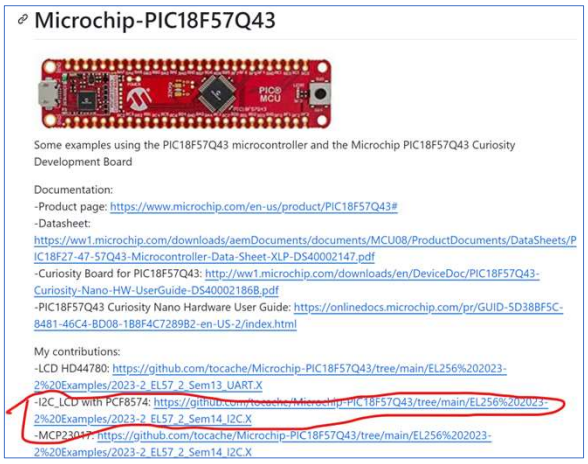
Comunicación I2C: Interface a IC PCF8574



54

Librería I2C_LCD

- Se ha desarrollado una librería para manipular el LCD a través del módulo adaptador I2C.
- Revisar el repositorio, la librería tiene como nombre I2C_LCD

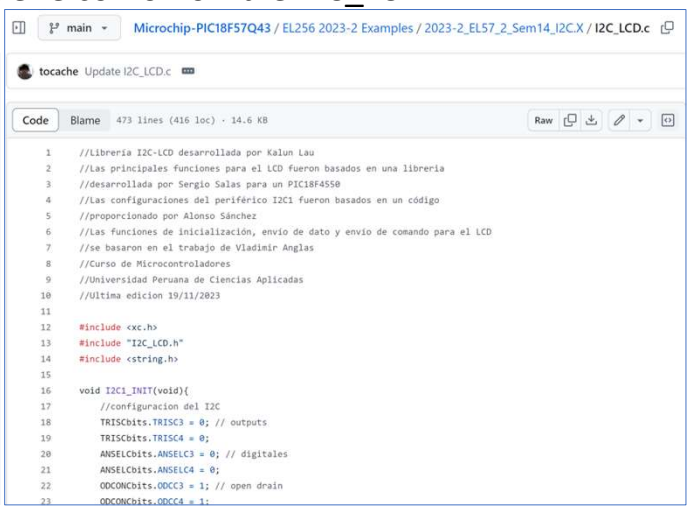


Microchip-PIC18F57Q43

Some examples using the PIC18F57Q43 microcontroller and the Microchip PIC18F57Q43 Curiosity Development Board

Documentation:
-Product page: https://www.microchip.com/en-us/product/PIC18F57Q43#
-Datasheet: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/PIC18F27-47-57Q43-Microcontroller-Data-Sheet-XLP-DS40002147.pdf
-Curiosity Board for PIC18F57Q43: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F57Q43-Curiosity-Nano-HW-UserGuide-DS40002186B.pdf
-PIC18F57Q43 Curiosity Nano Hardware User Guide: https://onlinedocs.microchip.com/pr/GUID-5D38BF5C-8481-46C4-BD08-1B8F4C7289B2-en-US-2/index.html

My contributions:
-LCD HD44780: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem13_UART.X
-I2C_LCD with PCF8574: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C.X
-MCP23017: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C.X



main · Microchip-PIC18F57Q43 / EL256 2023-2 Examples / 2023-2_EL57_2_Sem14_I2C.X / I2C_LCD.c

tocache Update I2C_LCD.c

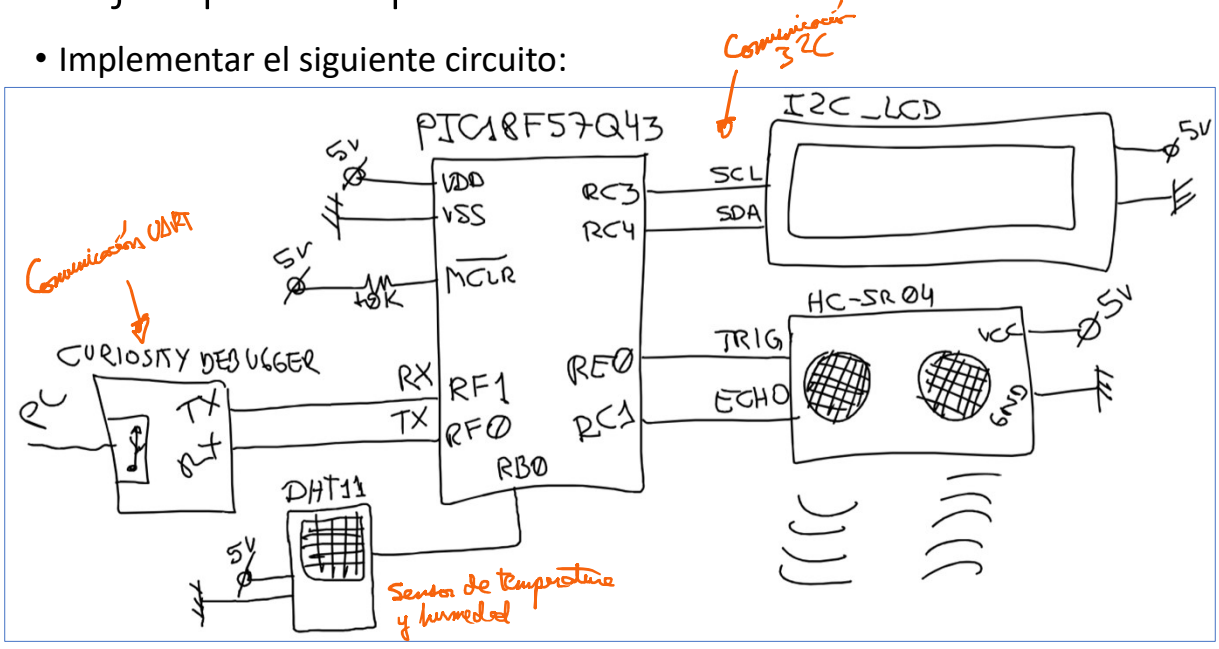
Code · Blame · 473 lines (416 loc) · 14.6 KB · Raw

```
//Librería I2C-LCD desarrollada por Kalun Lau
//Las principales funciones para el LCD fueron basados en una libreria
//desarrollada por Sergio Salas para un PIC18F4550
//Las configuraciones del periférico I2C1 fueron basados en un código
//proporcionado por Alonso Sánchez
//Las funciones de inicialización, envio de dato y envio de comando para el LCD
//se basaron en el trabajo de Vladimir Anglas
//Curso de Microcontroladores
//Universidad Peruana de Ciencias Aplicadas
//Ultima edicion 19/11/2023

#include <xc.h>
#include "I2C_LCD.h"
#include <string.h>

void I2C1_INIT(void){
    //configuracion del I2C
    TRISCbits.TRISC3 = 0; // outputs
    TRISCbits.TRISC4 = 0;
    ANSELCbits.ANSELC3 = 0; // digitales
    ANSELCbits.ANSELC4 = 0;
    ODCONCbits.ODCC3 = 1; // open drain
    ODCONCbits.ODCC4 = 1;
```

55

Ejemplo de aplicación 2024-1

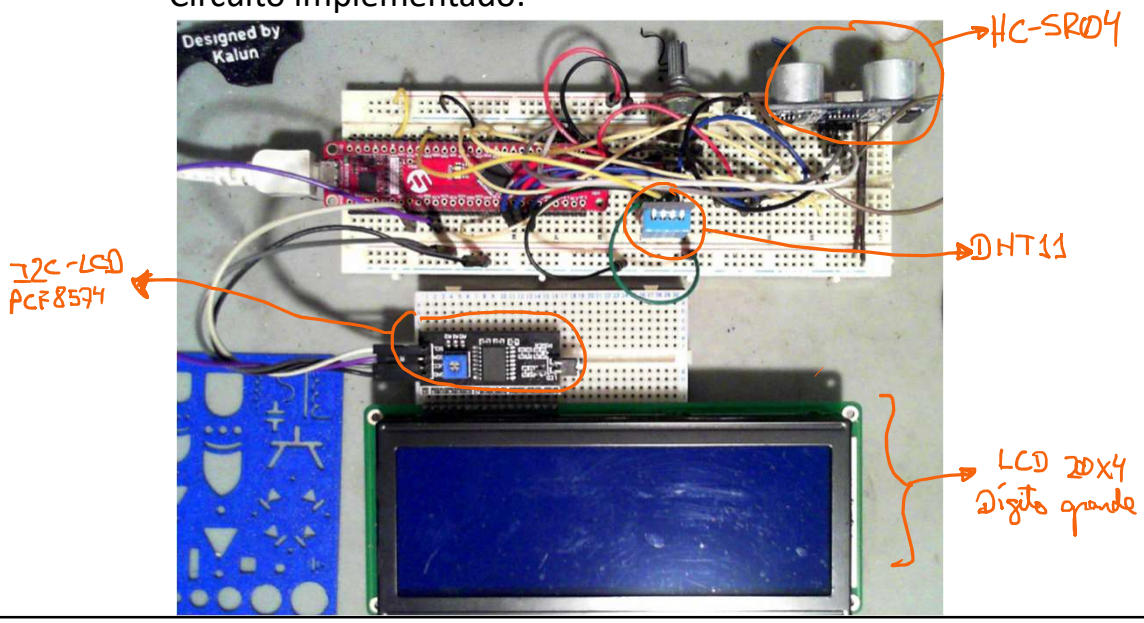
- Implementar el siguiente circuito:



56

Ejemplo de aplicación 2024-1

- Circuito implementado:



57

Ejemplo de aplicación 2024-1

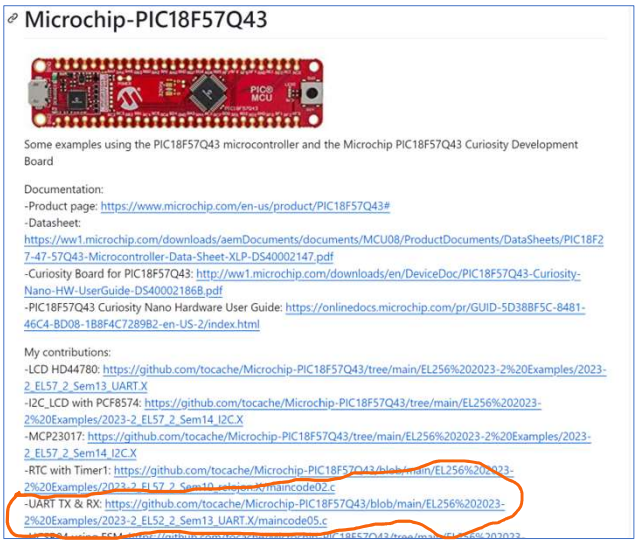
Detalle del circuito:

- Sensor DHT11 ó DHT22 conectado a RB0.
- Módulo I2C montado en el LCD 16x2 HD44780 y conectado el SCL a RC3 y SDA a RC4.
- Módulo HC-SR04 conectado el TRIG a RE0 y ECHO a RC1.
- Se empleará el puerto serial integrado en Curiosity Nano para el envio y recepción de datos hacia/desde la PC empleando un software de terminal serial (por ejemplo el Serial Monitor del Arduino IDE)

58

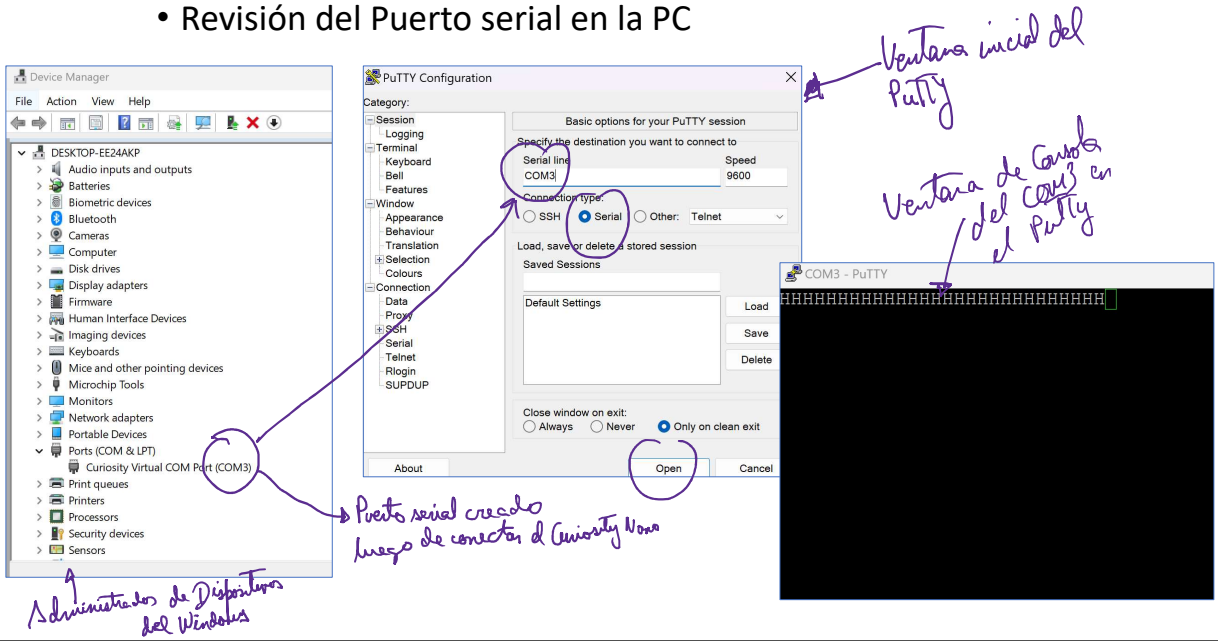
Ejemplo de aplicación 2024-1

- Las funciones para el uso del UART1 se encuentran en el repositorio:



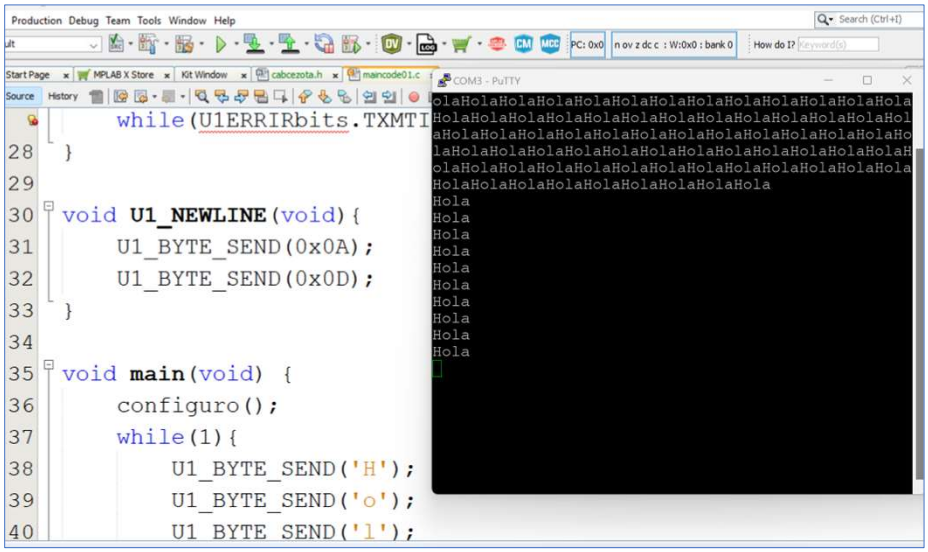
Ejemplo de aplicación 2024-1

- Revisión del Puerto serial en la PC



Ejemplo de aplicación 2024-1

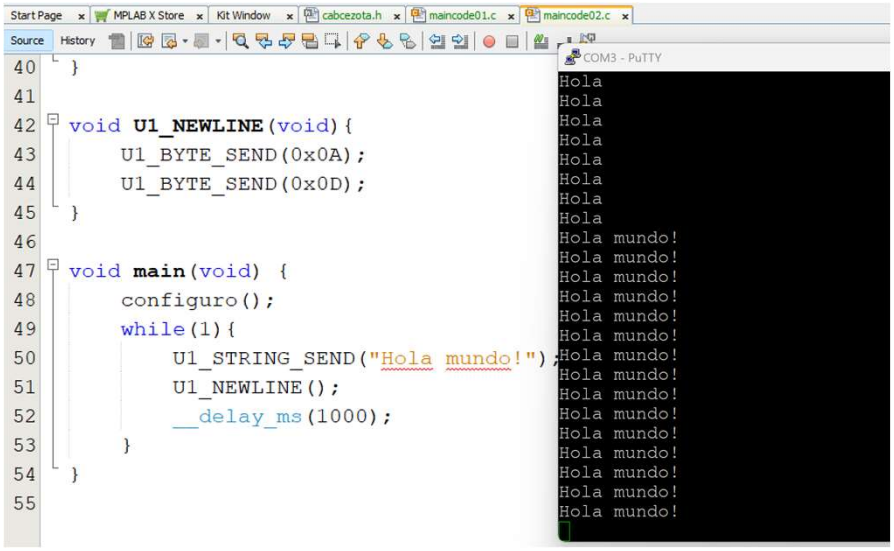
- Pruebas de envío de datos vía UART al terminal serial de la PC



61

Ejemplo de aplicación 2024-1

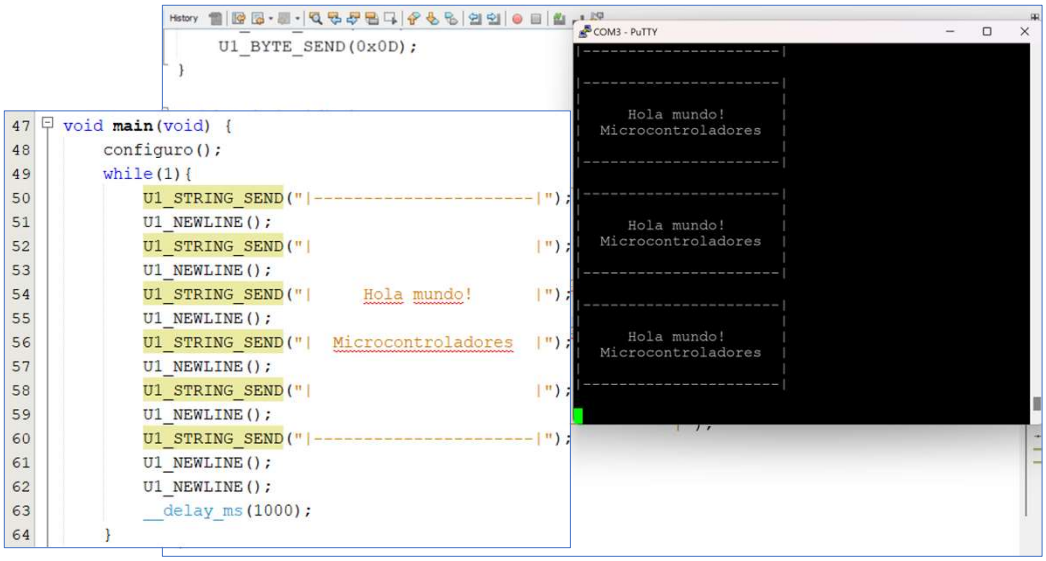
- Pruebas de envío de datos vía UART al terminal serial de la PC



62

Ejemplo de aplicación 2024-1

- Pruebas de envío de datos vía UART al terminal serial de la PC

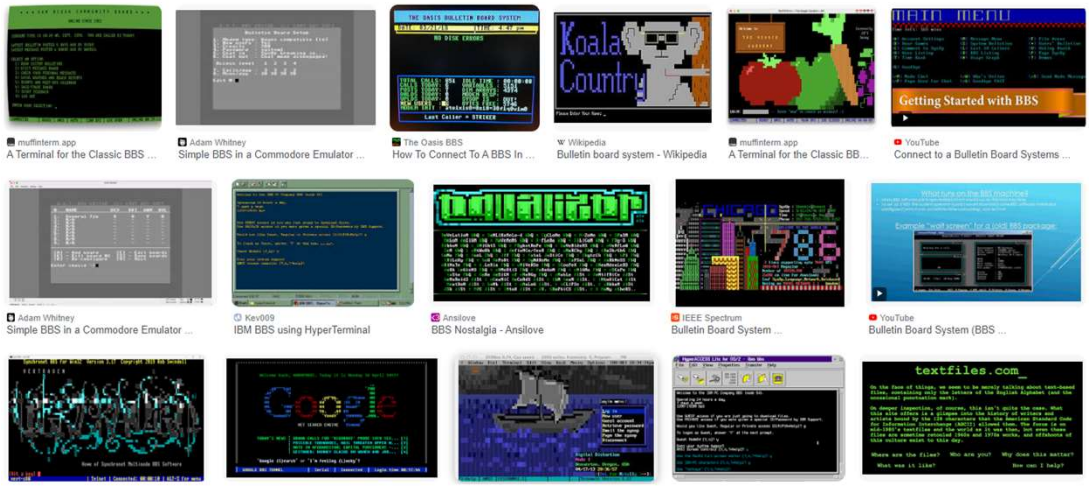


Ejemplo de aplicación 2024-1

- Pruebas de envío de datos vía UART al terminal serial de la PC
- https://patorjk.com/software/taag/

Ejemplo de aplicación 2024-1

- Antiguas visualizaciones en BBS

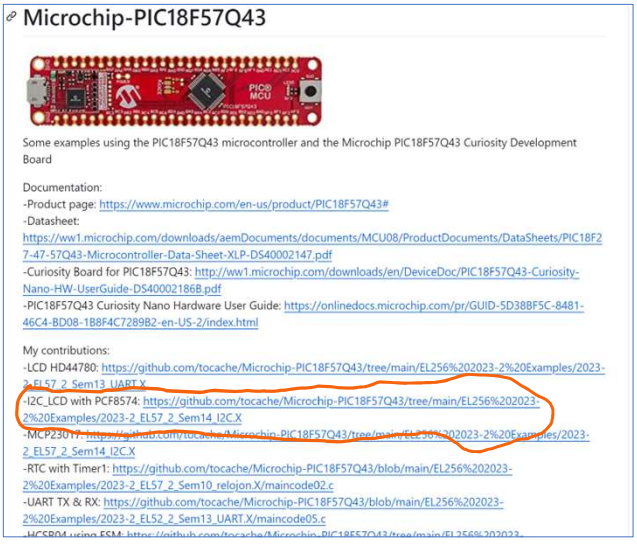


muffinterm app
A Terminal for the Classic BBS ...
Adam Whitney
Simple BBS in a Commodore Emulator ...
The Oasis BBS
How To Connect To A BBS In ...
Wikipedia
Bulletin board system - Wikipedia
muffinterm app
A Terminal for the Classic BB...
YouTube
Connect to a Bulletin Board Systems ...
Adam Whitney
Simple BBS in a Commodore Emulator ...
Kev009
IBM BBS using HyperTerminal
Ansilove
BBS Nostalgia - Ansilove
IEEE Spectrum
Bulletin Board System ...
YouTube
Bulletin Board System (BBS ...

65

Ejemplo de aplicación 2024-1

- La librería I2C-LCD a emplear en los siguientes ejemplos se encuentra en el repositorio:
- La función I2C_LCD_INIT() es el empleado para inicializar el periférico I2C1 del PIC18F57Q43 y para inicializar el modulo LCD. Se deberá llamar a dicha función antes de operar el LCD.
- Las funciones para operar el display HD44780 son muy similares a las empleadas en la librería LCD vista anteriormente.



Microchip-PIC18F57Q43

Some examples using the PIC18F57Q43 microcontroller and the Microchip PIC18F57Q43 Curiosity Development Board

Documentation:
-Product page: https://www.microchip.com/en-us/product/PIC18F57Q43#
-Datasheet:
https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/PIC18F27-47-57Q43-Microcontroller-Data-Sheet-XLP-DS40002147.pdf
-Curiosity Board for PIC18F57Q43: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F57Q43-Curiosity-Nano-HW-UserGuide-DS40002186B.pdf
-PIC18F57Q43 Curiosity Nano Hardware User Guide: https://onlinedocs.microchip.com/pr/GUID-5D38BF5C-8481-46C4-BD08-1B8F4C7289B2-en-US-2/index.html

My contributions:
-LCD HD44780: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem13_UART.X
-I2C_LCD with PCF8574: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C.X
-MCP23017: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem14_I2C.X
-RTC with Timer1: https://github.com/tocache/Microchip-PIC18F57Q43/blob/main/EL256%202023-2%20Examples/2023-2_EL57_2_Sem10_relojon.X/maincode02.c
-UART TX & RX: https://github.com/tocache/Microchip-PIC18F57Q43/blob/main/EL256%202023-2%20Examples/2023-2_EL52_2_Sem13_UART.X/maincode05.c
-MCSP04 using FSM: https://github.com/tocache/Microchip-PIC18F57Q43/tree/main/EL256%202023-

66

Ejemplo de aplicación 2024-1

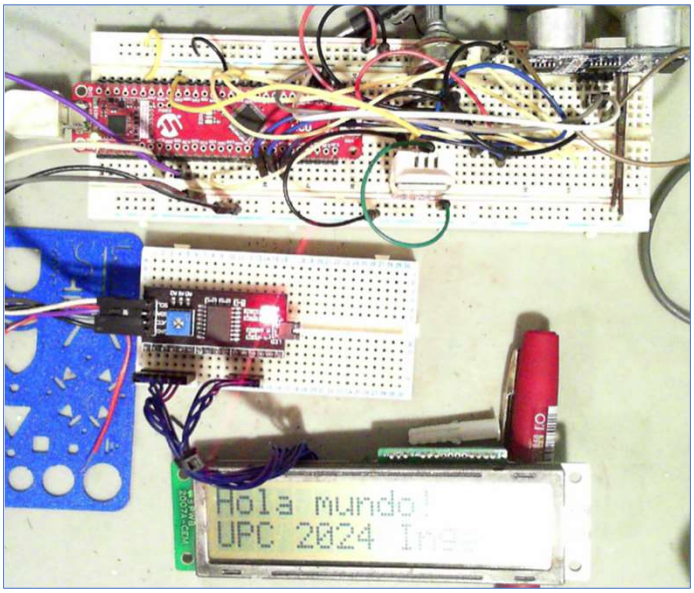
- Evidencia de visualización en el LCD 16x2 conectado con el I2C-LCD

```
48  void main(void) {
49      configuro();
50      I2C_LCD_INIT();
51      while(1){
52          U1_STRING_SEND("|----------------------|");
53          U1_NEWLINE();
54          U1_STRING_SEND("|                      |");
55          U1_NEWLINE();
56          U1_STRING_SEND("|      Hola mundo!     |");
57          U1_NEWLINE();
58          U1_STRING_SEND("|  Microcontroladores  |");
59          U1_NEWLINE();
60          U1_STRING_SEND("|                      |");
61          U1_NEWLINE();
62          U1_STRING_SEND("|----------------------|");
63          U1_NEWLINE();
64          U1_NEWLINE();
65          I2C_POS_CURSOR(1,0);
66          I2C_ESCRIBE_MENSAJE2("Hola mundo!");
67          I2C_POS_CURSOR(2,0);
68          I2C_ESCRIBE_MENSAJE2("UPC 2024 Ingee");
69          __delay_ms(1000);
70      }
71  }
```

Inicialización del PCF8574 del I2C-LCD y del LCD en modo 4bit

Comp. serial

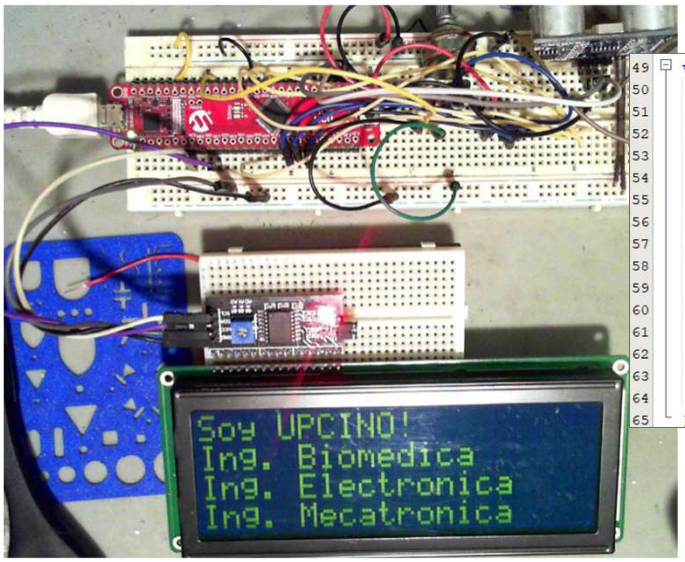
control I2C-LCD



67

Ejemplo de aplicación 2024-1

- Pruebas de impression en LCD con cuatro líneas:

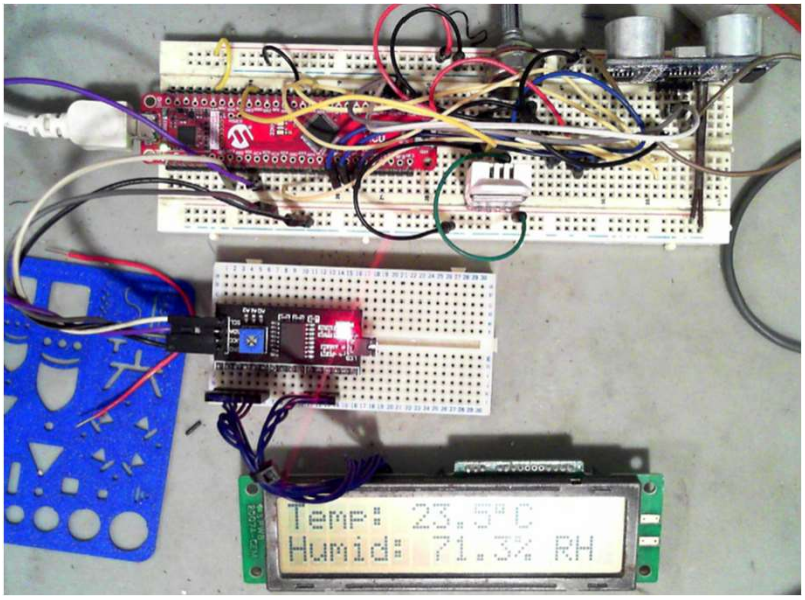



```
49  void main(void) {
50      configuro();
51      I2C_LCD_INIT();
52      while(1){
53          U1_STRING_SEND("Hola mundo! Mi nombre es Kalun");
54          U1_NEWLINE();
55          I2C_POS_CURSOR(1,0);
56          I2C_ESCRIBE_MENSAJE2("Soy UPCINO!");
57          I2C_POS_CURSOR(2,0);
58          I2C_ESCRIBE_MENSAJE2("Ing. Biomedica");
59          I2C_POS_CURSOR(3,0);
60          I2C_ESCRIBE_MENSAJE2("Ing. Electronica");
61          I2C_POS_CURSOR(4,0);
62          I2C_ESCRIBE_MENSAJE2("Ing. Mecatronica");
63          __delay_ms(2000);
64      }
65  }
```


68

Ejemplo de aplicación 2024-1

- Pruebas de medición con el sensor DHT22:



69

Fin de la sesión

70