Programación Estructurada con Microcontroladores

M.C. Geovanny Giorgana
ggiorgana@yahoo.com.mx

Universidad Anáhuac Mayab

7 de noviembre de 2018

PIC18F4550

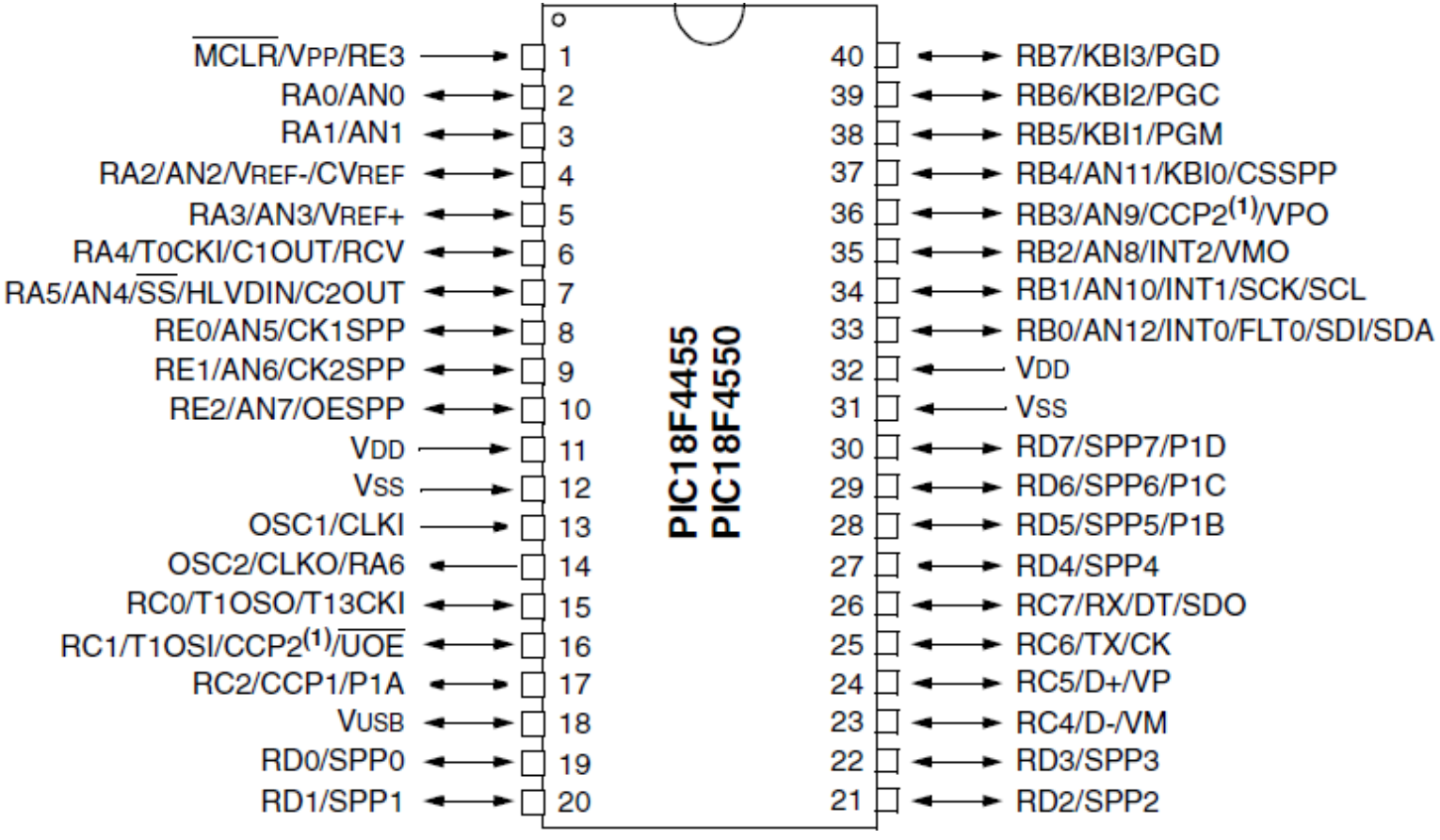


Figura 1: Diagrama de pines del PIC18F4550, encapsulado 40 PDIP.

PIC18F4550: Lista de materiales para prender un LED

- 1x PIC18F4550 40 pines (PDIP)
- 1x Cristal de cuarzo de 4 MHz ó 20 MHz.
- 1x LED 5mm de cualquier color
- 1x R330 Ω
- 1x Diodo Schottky
- 1x Capacitor electrolítico de 0.1 μF
- 1x R10 KΩ
- 1x Push button para protoboard
- Cable para protoboard de diferentes colores.
- 2x Capacitores de 20 pF.
- 5x Cables Du-Pont Macho-Macho o 1x Header de pines macho de 5 posiciones.

PIC18F4550: Diagrama de reloj

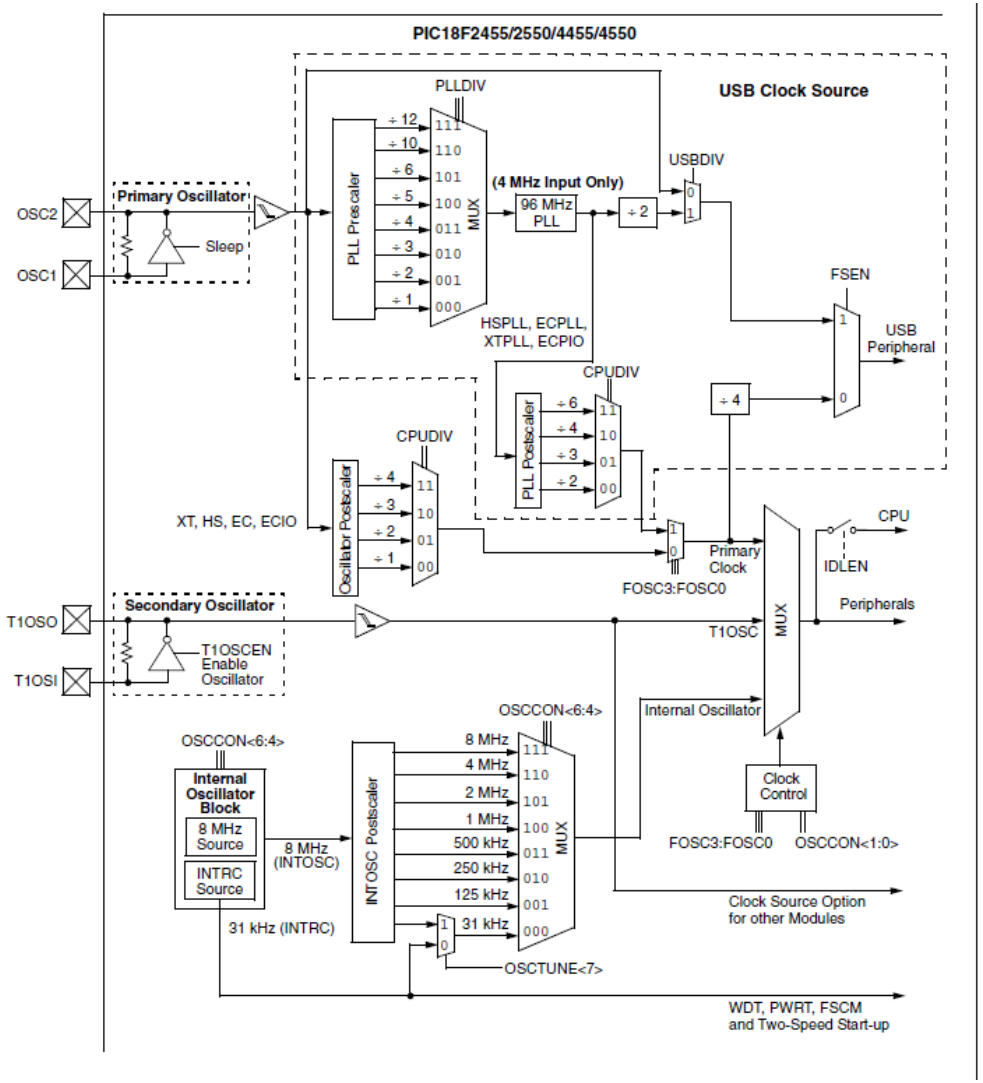


Figura 2: Diagrama de reloj del PIC18F4550.

Timer0

El PIC18F4550 cuenta con un módulo conocido como TIMER0 con las siguientes características:

- Contador/temporizador de 8 bits o 16 bits
- Prescalador de 8 bits
- Fuente de señal de reloj programable: externa o interna
- Selección programable del flanco para reloj externo
- Interrupción por desbordamiento

Timer0

El funcionamiento del TIMER0 se controla a través de los siguientes registros:

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TMR0L	Timer0 Register Low Byte								52
TMR0H	Timer0 Register High Byte								52
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	51
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	51
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	52
TRISA	—	TRISA6(1)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	54

Legend: — = unimplemented locations, read as '0'. Shaded cells are not used by Timer0.

Note 1: RA6 is configured as a port pin based on various primary oscillator modes. When the port pin is disabled, all of the associated bits read '0'.

Figura 3: Registros asociados al TIMER0.

Timer0

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
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 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

Figura 4: Registro de control del TIMER0 (**T0CON**).

Timer0: 8 bits

Cuando se usa como temporizador, el registro TMR0L aumenta su valor por cada ciclo de instrucción (sin preescalador).

Cuando el registro TMR0L llega a 255, el siguiente ciclo de instrucción hará que

- el registro TMR0L regresa a 0, y que
- la bandera de interrupción T0IF se ponga a 1.

La bandera T0IF debe ponerse a 0 manualmente.

Timer0

El módulo preescalador del TIMER0 divide la frecuencia del reloj para prolongar el tiempo total de temporización:

La asignación del preescalador y su valor se controla por medio de los bits **PSA** y **T0PS<2:0>**, repectivamente:

bit 3	**PSA**: Timer0 Prescaler Assignment bit 1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler. 0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.
bit 2-0	**T0PS2:T0PS0**: Timer0 Prescaler Select bits 111 = 1:256 Prescale value 110 = 1:128 Prescale value 101 = 1:64 Prescale value 100 = 1:32 Prescale value 011 = 1:16 Prescale value 010 = 1:8 Prescale value 001 = 1:4 Prescale value 000 = 1:2 Prescale value

Timer0

La ecuación que define el tiempo total de temporización es

$$\frac{1}{T_{out}} = f_{out} = \frac{f_{clk}}{4 * \text{Prescaler} * (256 - \text{TMR0L}) * \text{Count}}$$

donde:

T_{out} = Tiempo de temporización total en seg.

f_{clk} = Frecuencia del oscilador externo en Hz.

Prescaler = Valor del multiplicador.

TMR0L = Valor del registro TMR0L.

Count = Número de veces que se desbordará TMR0L.

Timer0

Práctica 1. Hacer parpadear a un LED de forma que se prenda 5 segundos y después se apague 2 segundos.

1. Utilizando las funciones *wait_5_sec()* y *wait_2_sec()*.
2. Utilizando la función *wait_in_ms(type time_in_ms)*.

Timer0 - Función de contador

El módulo TMR0 también puede funcionar como contador de transiciones externas.

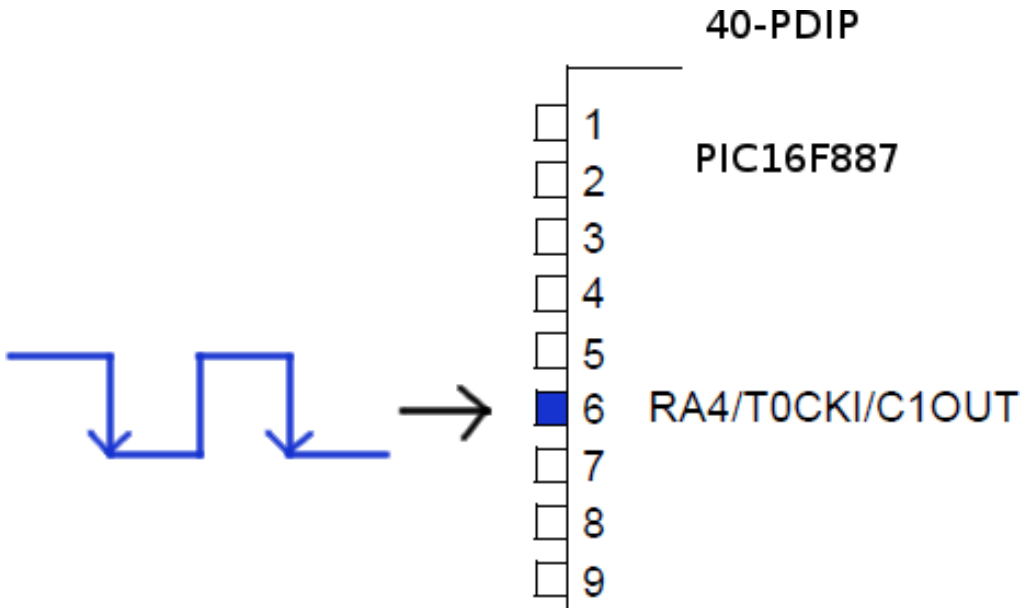


Figura 5: La patita T0CKI del PIC sirve para registrar la cantidad de flancos provenientes de una señal externa.

Timer0 - Función de contador

Por medio del registro **T0CON** logramos

- seleccionar la función de contador para el TIMER0 (**T0CS**),
- configurarlo como contador de 8 bits o de 16 bits (**T08BIT**),
- seleccionar el tipo de transición que incremeta el valor del registro TMR0 (**T0SE**),
- habilitar el prescalador y asignarle un valor (**PSA** y **T0PS2:0**).

El valor del contador se almacena en el registro **TMR0**.

El bit **T0IF** se prende al desbordarse **TMR0**.

Timer0

Práctica 2. Uso del contador del PIC.

a) Crear una función que retorne cuando se detecten N transiciones.

void waitForNCounts(unsigned int N, unsigned char edge)

Donde

N = Número de transiciones que debe esperar para salir.

edge = Tipo de transición que debe esperar (0 Rising, 1 Falling).

Cambiar el estado de un LED cada 300 transiciones. Usar el generador de funciones.

b) Prender un LED únicamente cuando el valor del contador es 10. Utilice un pulsador para obtener las transiciones. El contador debe incrementar con transición de alto a bajo. Mantén el ruido del pulsador al mínimo.

Interrupción externa

La interrupción externa se activa al presentarse un flanco de subida o un flanco de bajada en cualquiera de los siguientes pines

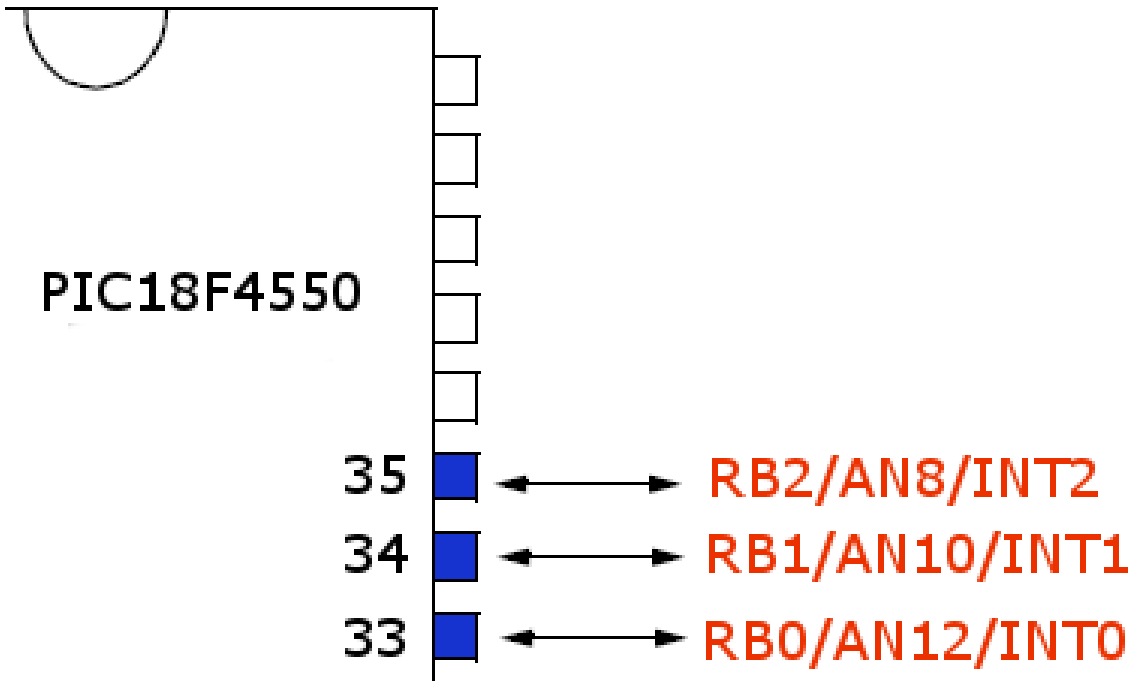


Figura 6: Los pines INT capturan las señales que generan las interrupciones externas.

Interrupción externa

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all high priority interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

Figura 7: Registro INTCON.

Interrupción externa

bit 4 **INT0IE:** INT0 External Interrupt Enable bit

1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit

1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit

1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit

1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit(1)

1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Figura 8: Registro INTCON (continuación).

Interrupción externa

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6 **SBOREN:** BOR Software Enable bit(1)
For details of bit operation, see Register 4-1.

bit 5 **Unimplemented:** Read as '0'

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
For details of bit operation, see Register 4-1.

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
For details of bit operation, see Register 4-1.

bit 2 **$\overline{PD}$:** Power-Down Detection Flag bit
For details of bit operation, see Register 4-1.

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(2)
For details of bit operation, see Register 4-1.

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
For details of bit operation, see Register 4-1.

Figura 9: Registro RCON.

Interrupción externa

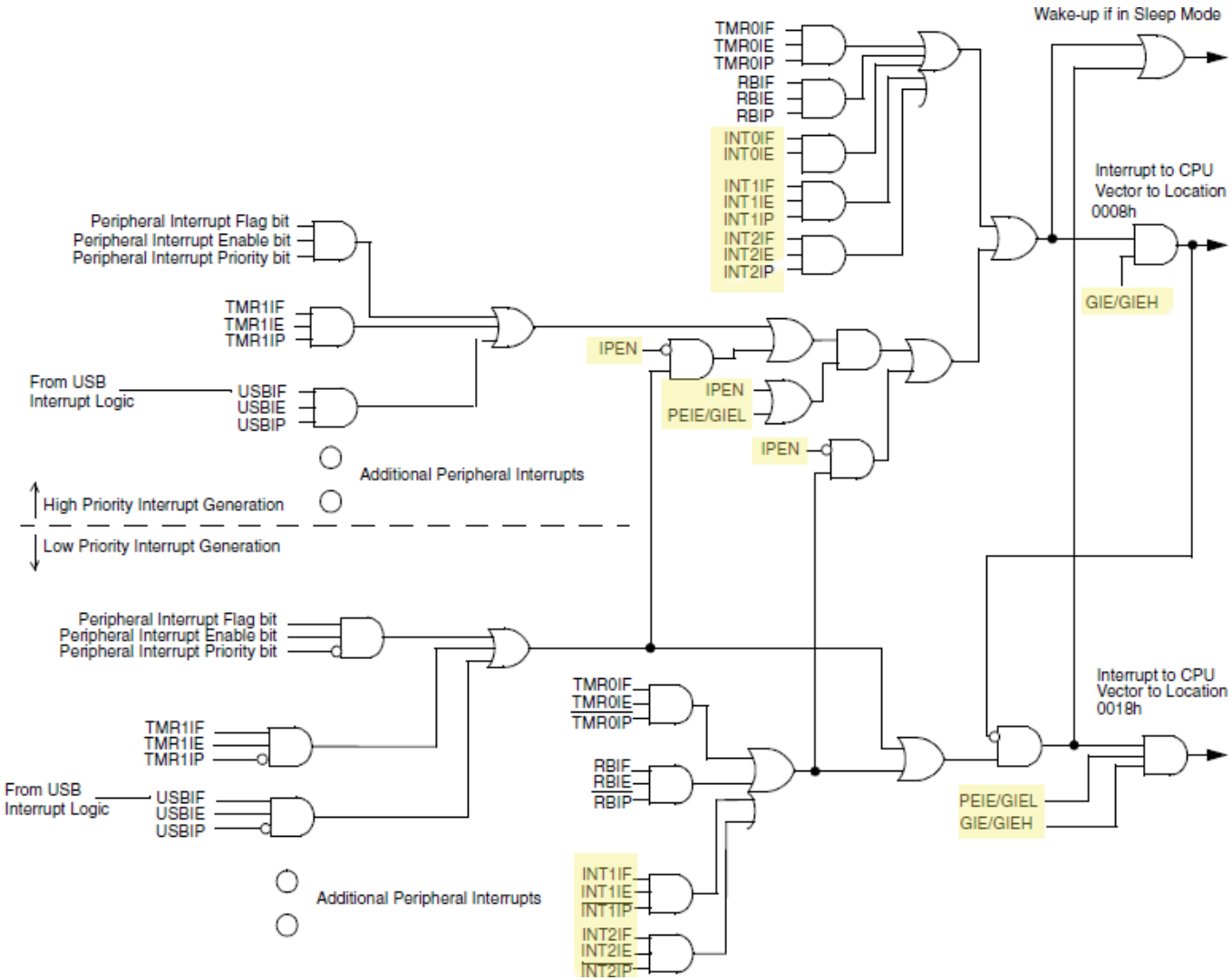


Figura 10: Lógica de interrupción del PIC18F4550.

Interrupción externa

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
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 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **Unimplemented:** Read as '0'

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **Unimplemented:** Read as '0'

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

Figura 11: Registro INTCON3.

Interrupción externa

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
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 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 3 **Unimplemented:** Read as '0'

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as '0'

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

Figura 12: Registro INTCON2.

Interrupciones: Igual prioridad

```
// General interrupts.

void __interrupt() isr(void) {          // Doble guión bajo.
    if(PIR1bits.TMR1IF) {
      PIR1bits.TMR1IF = 0;
      // Do something...
    }
    else if(INTCONbits.INT0IF) {
      INTCONbits.INT0IF = 0;
      // Do something...
    }
}
```

Fuentes de interrupción del PIC18F4550

- Lectura/Escritura del puerto paralelo.
- Conversión de Analógico a digital.
- Recepción de datos EUSART.
- Transmisión de datos EUSART.
- Desbordamiento de temporizadores TMR0, TMR1, TMR2 o TMR3.
- Captura o comparación del módulo CCP.
- Cambio de la entrada del comparador.
- Escritura de la memoria EEPROM del μC.
- USB.
- Fallo del sistema oscilador.
- Colisión del bus.
- Condición de alta/baja de voltaje.

Interrupciones: Alta prioridad y baja prioridad

```
// High-priority interrupts.
#pragma code high_isr = 0x08
void high_isr(void) {
    if (T0IF) ...
}
#pragma code

// Low-priority interrupts.
#pragma code low_isr = 0x18
void low_isr(void) {
    if (INT0IF) ...
}
#pragma code
```

Interrupciones: Alta prioridad y baja prioridad

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
SPPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
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 **SPPIP:** Streaming Parallel Port Read/Write Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP:** EUSART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP:** EUSART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP:** Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

Note 1: This bit is reserved on 28-pin devices; always maintain this bit clear.

Figura 13: Registro IPR1.

Interrupciones: Alta prioridad y baja prioridad

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	CMIP	USBIP	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP
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 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **CMIP:** Comparator Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **USBIP:** USB Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **EEIP:** Data EEPROM/Flash Write Operation Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **BCLIP:** Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **HLVDIP:** High/Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP:** CCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

Figura 14: Registro IPR2.

Interrupción externa

Práctica 3.

a) Prender un LED al apretar el pulsador 1 usando la interrupción externa INT0 del PIC. Usar flancos de subida.

b) Alternar el encendido y apagado de un LED apretando el pulsador 1. Usar la interrupción externa INT1 del PIC. Usar flancos de bajada.

c) Prender un LED al apretar el pulsador 1, sólo si han transcurrido más de 10 segundos. Usar la interrupción externa INT2 del PIC y flancos de bajada.

Convertidor A/D

El PIC18F4550 cuenta con un módulo convertidor de analógico a digital (ADC) de 10 bits y 13 canales.

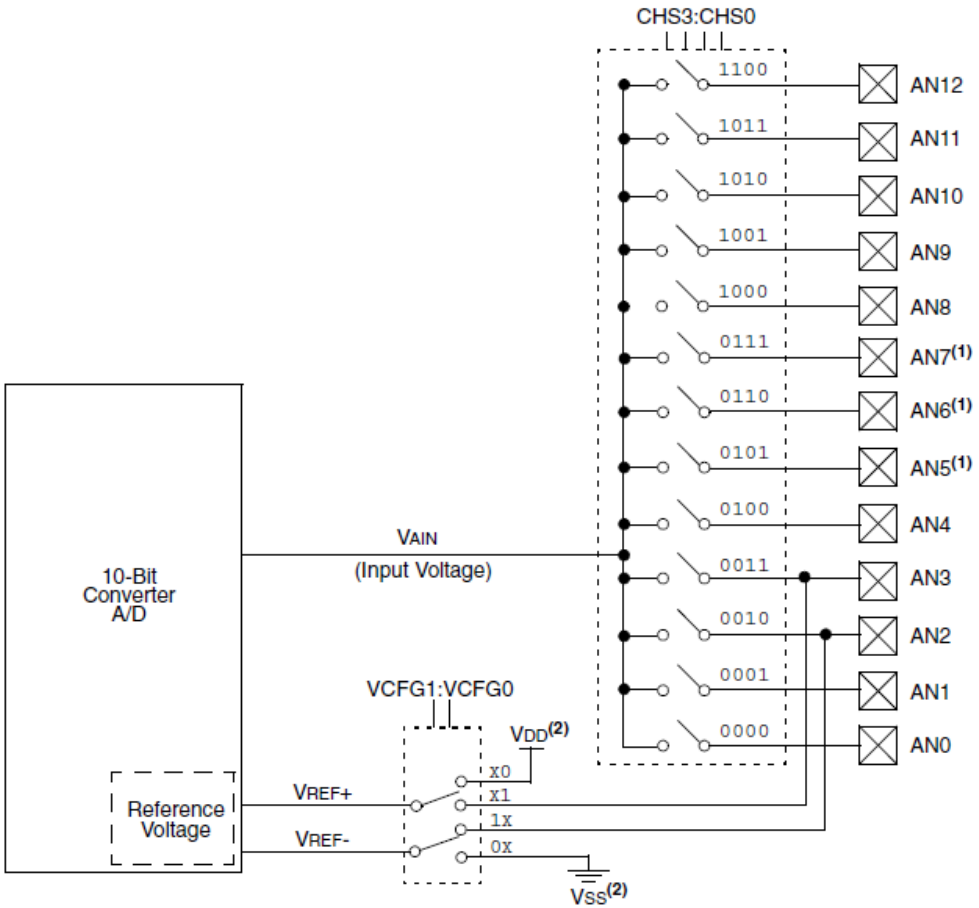


Figura 15: Diagrama de bloques del ADC.

Convertidor A/D

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits

0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

Figura 16: Registro ADCON0.

Convertidor A/D

bit 1	**GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit When ADON = 1: 1 = A/D conversion in progress 0 = A/D Idle
bit 0	**ADON:** A/D On bit 1 = A/D converter module is enabled 0 = A/D converter module is disabled

Figura 17: Registro ADCON0 (continuación).

Poner a 1 el bit **GO/$\overline{\text{DONE}}$** genera una conversión.

El bit se pone en 0 automáticamente cuando la conversión termina.

Convertidor A/D

La resolución de cada bit procedente de la conversión depende de la tensión de referencia V_{ref}, de acuerdo a la siguiente fórmula:

$$\text{Resolución} = (V_{ref+} - V_{ref-})/1024 \quad (1)$$

Si $V_{ref+} = 5$V y $V_{ref-} = 0$V, la resolución de cada bit será igual a 4.883 mV/bit.

Si $V_{ref+} = 5$V y $V_{ref-} = 2$V, la resolución de cada bit será igual a 2.93 mV/bit.

Convertidor A/D

Voltaje analógico de entrada	**Valor digital (binario)**	**Valor digital (decimal)**
0 V	00000000	0
4.88 mV	00000001	1
9.76 mV	00000010	2
14.64 mV	00000011	3
4.9902 V	11111110	1022
4.9951 V	11111111	1023

Cuadro 1: Ejemplo de resultados de una conversión A/D de 10 bits con $V_{ref+} = 5$V, $V_{ref-} = 0$V y justificado hacia la derecha.

Convertidor A/D

El resultado de la conversión se graba en los registros

- ADRESH
- ADRESL

ADRESH	A/D Result Register High Byte
ADRESL	A/D Result Register Low Byte

Ambos registros son de 8 bits.

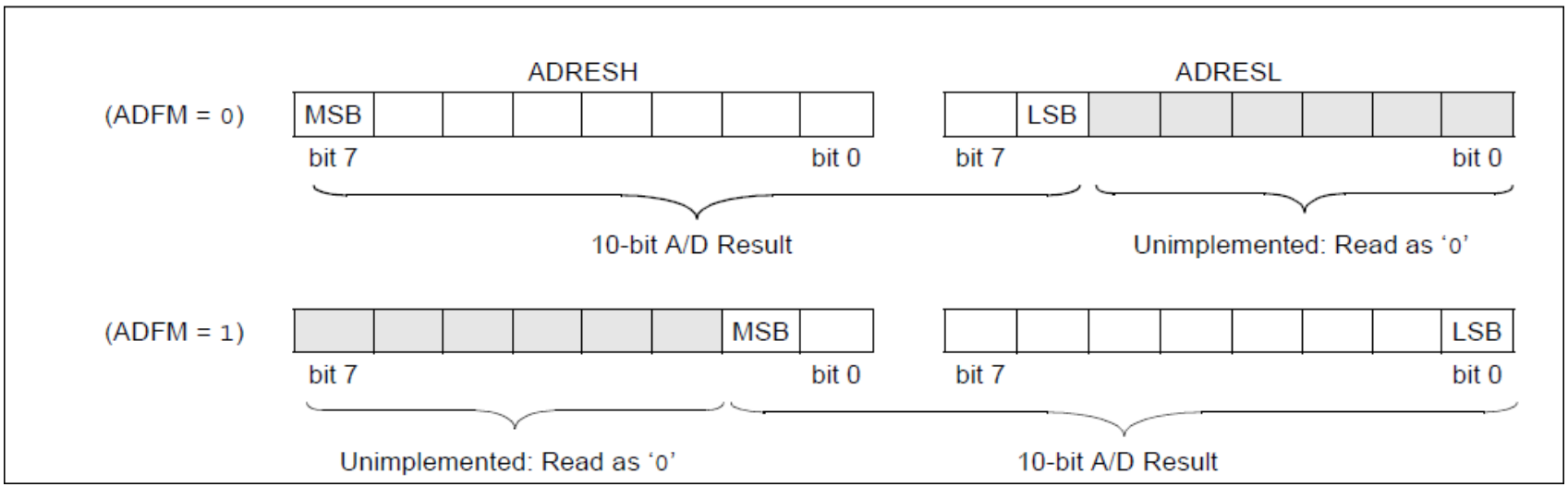
¿Con qué justificación se graba el resultado de 10 bits en estos registros? ¿Izquierda o derecha?

Convertidor A/D

El bit ADFM del registro **ADCON2** selecciona el formato del resultado de la conversión:

- Justificación hacia la izquierda (ADFM = 0)
- Justificación hacia la derecha (ADFM = 1)

FIGURE 9-3: 10-BIT A/D CONVERSION RESULT FORMAT



Convertidor A/D

Los pines que actuarán como entradas analógicas se seleccionan por medio del registro **ADCON1**.

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG0	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG0:** Voltage Reference Configuration bit (V_{REF-} source)
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Voltage Reference Configuration bit (V_{REF+} source)
1 = V_{REF+} (AN3)
0 = V_{DD}

Figura 18: Registro ADCON1.

Convertidor A/D

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Figura 19: Registro ADCON1 (continuación).

Convertidor A/D

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as '0'

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

Figura 20: Registro ADCON2.

Convertidor A/D

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits

111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Figura 21: Registro ADCON2 (continuación).

A/D Conversion Clock (T_{AD})

T_{AD} es el tiempo que dura la conversión de cada bit.

Para el PIC18F4550, el valor mínimo es de 0.7 μs.

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D Clock Period	PIC18FXXXX	0.7	25.0(1)	µs	TOSC based, VREF ≥ 3.0V
			PIC18LFXXXX	1.4	25.0(1)	µs	VDD = 2.0V, TOSC based, VREF full range
			PIC18FXXXX	TBD	1	µs	A/D RC mode
			PIC18LFXXXX	TBD	3	µs	VDD = 2.0V, A/D RC mode
131	TCNV	Conversion Time (not including acquisition time)(2)		11	12	TAD	
132	TACQ	Acquisition Time(3)		1.4 TBD	— —	µs µs	-40°C to +85°C 0°C ≤ to ≤ +85°C
135	TSWC	Switching Time from Convert → Sample		—	(Note 4)		
137	TDIS	Discharge Time		0.2	—	µs	

Legend: TBD = To Be Determined

Note 1: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

2: ADRES registers may be read on the following TCY cycle.

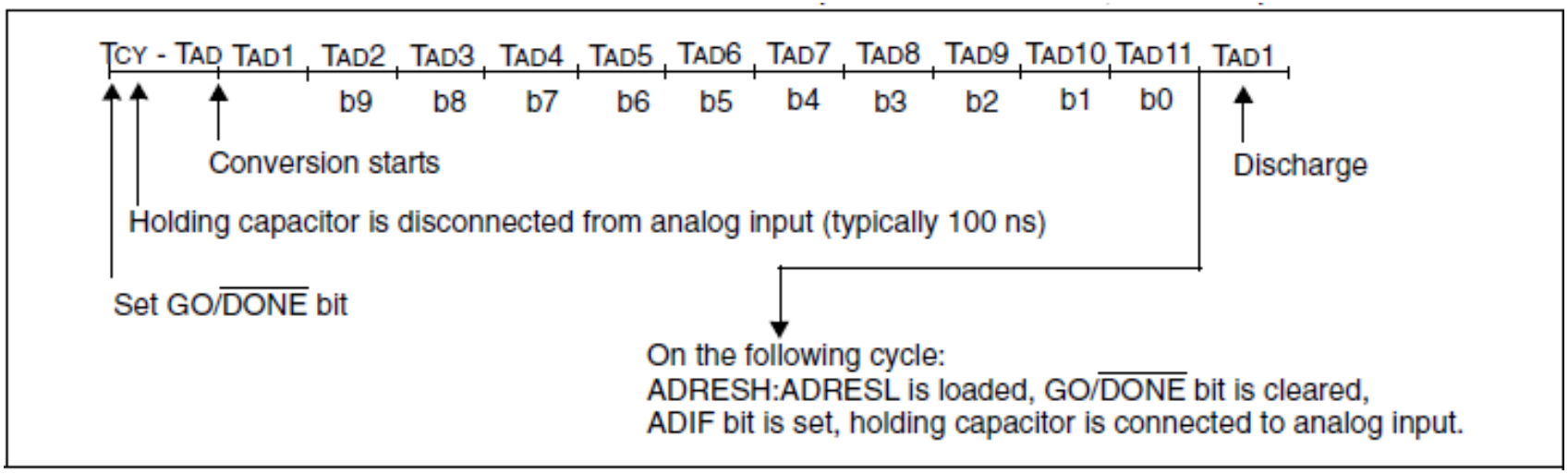
3: The time for the holding capacitor to acquire the "New" input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (*Rs*) on the input channels is 50Ω.

4: On the following cycle of the device clock.

Figura 22: Requisitos para la conversión A/D.

A/D Conversion Clock (T_{AD})

Para conversiones de 10 bits, es necesario esperar al menos un tiempo de $11 \cdot T_{AD}$.



Un tiempo mínimo de espera de 3 $\cdot T_{AD}$ es necesario antes de realizar la siguiente adquisición.

A/D Conversion Clock (T_{AD})

El T_{AD} seleccionado debe ser mayor que el valor de T_{AD} mínimo (0.7 μs) pero tan pequeño como sea posible.

Para $F_{OSC} = 48$ MHz:

$$\begin{aligned} T_{AD} &= \frac{64}{F_{OSC}} \\ &= 64 T_{OSC} \\ &= 64(2.08333 \times 10^{-8}) \\ &= 1.333\ \mu s \end{aligned}$$

1.333 μs es mayor que 0.7 μs.

A/D Conversion Clock (T_{AD})

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18FXXXX	PIC18LFXXXX(4)
2 TOSC	000	2.86 MHz	1.43 MHz
4 TOSC	100	5.71 MHz	2.86 MHz
8 TOSC	001	11.43 MHz	5.72 MHz
16 TOSC	101	22.86 MHz	11.43 MHz
32 TOSC	010	45.71 MHz	22.86 MHz
64 TOSC	110	48.0 MHz	45.71 MHz
RC(3)	x11	1.00 MHz(1)	1.00 MHz(2)

Note 1: The RC source has a typical TAD time of 4 ms.

2: The RC source has a typical TAD time of 6 ms.

3: For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.

4: Low-power devices only.

Figura 23: Valores que toma T_{AD} según la programación de ADCS1:ADCS0 y la frecuencia de funcionamiento del microcontrolador.

Convertidor A/D: Tiempo de adquisición

El tiempo de adquisición es el tiempo que tarda en cargarse por completo el capacitor C_{HOLD} cuando se activa el bit **GO/$\overline{\text{DONE}}$**.

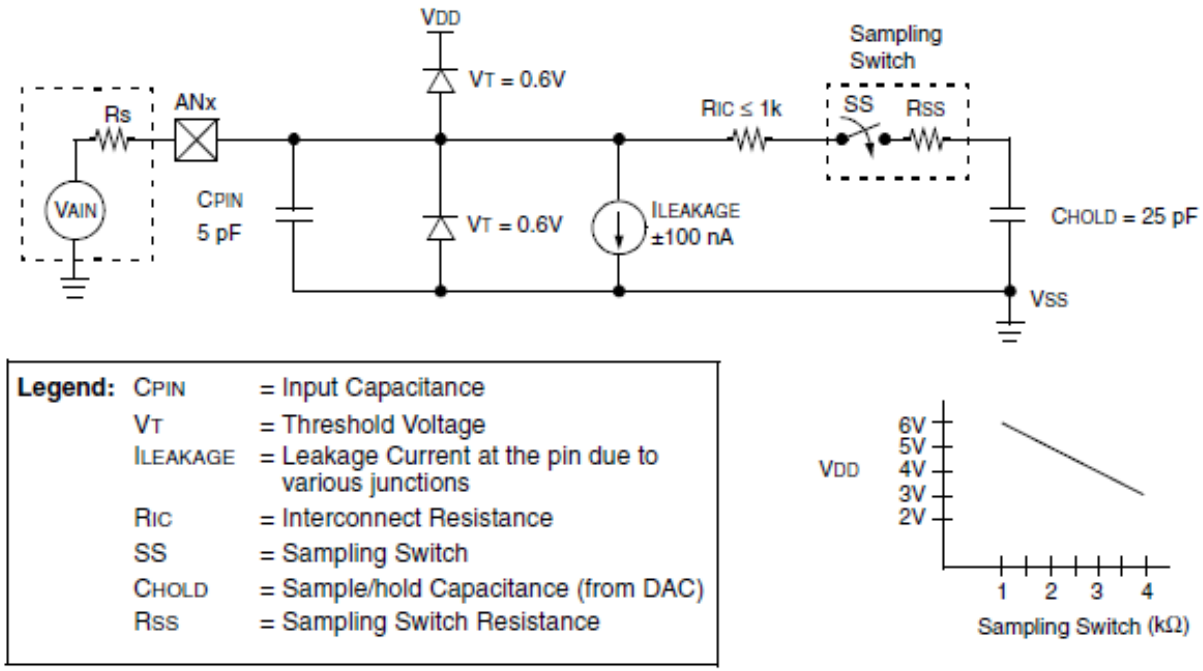


Figura 24: Modelo de la entrada analógica.

Depende de varios factores: Valor de C_{HOLD}, impedancia de la fuente, VDD, temperatura, etc.

Convertidor A/D: Tiempo de adquisición

$$
\begin{aligned}
T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\
&= T_{AMP} + T_C + T_{COFF}
\end{aligned}
$$

Figura 25: Ecuación para el cálculo del tiempo de adquisición (T_{ACQ}).

Convertidor A/D: Tiempo de adquisición

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2048)$$

Figura 26: Ecuación para el cálculo del tiempo mínimo de carga (T_C).

Convertidor A/D: Tiempo de adquisición

T_{ACQ}	=	$T_{AMP} + T_C + T_{COFF}$
T_{AMP}	=	0.2 µs
T_{COFF}	=	(Temp – 25°C)(0.02 µs/°C) (85°C – 25°C)(0.02 µs/°C) 1.2 µs

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, T_{COFF} = 0 ms.

T_C	=	-(C_{HOLD})(R_{IC} + R_{SS} + R_S) ln(1/2048) µs -(25 pF) (1 kΩ + 2 kΩ + 2.5 kΩ) ln(0.0004883) µs 1.05 µs
T_{ACQ}	=	0.2 µs + 1.05 µs + 1.2 µs 2.45 µs

Figura 27: Cálculo del tiempo de adquisición mínimo requerido.

Convertidor A/D: Tiempo de adquisición

Para $F_{OSC} = 48$ MHz ($T_{AD} = 1.333\ \mu s$):

111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

Valores válidos para el tiempo de adquisición.

Convertidor A/D

1. Configurar el pin ANx como entrada.
2. Definir al pin ANx como entrada analógica.
3. Configurar el módulo ADC:
   - Seleccionar el reloj de conversión del ADC.
   - Configurar el voltaje de referencia.
   - Seleccionar el canal de entrada del ADC.
   - Seleccionar el formato del resultado.
   - Prender el módulo ADC.
4. Configurar la interrupción del ADC (opcional).
5. Esperar el tiempo de adquisición.
6. Prender el bit **GO/#DONE** para comenzar la conversión.
7. Esperar por la conversión:
   - Checar constantemente el bit **GO/#DONE**.
   - Esperar por la interrupción.
8. Leer el resultado.
9. Borrar la bandera de interrupción por ADC.

Convertidor A/D

Práctica 4.

a) Prender un LED cuando el valor del voltaje de entrada se encuentre entre 2 y 3.5 volts. Utilizar $V_{REF+} = VDD$ y $V_{REF-} = VSS$. Utilice interrupciones.

b) Dadas dos entradas analógicas v_1 y v_2, prender un LED cuando el valor de voltaje se encuentre entre 2 y 3.5 volts. Hay dos LEDs, uno para cada entrada analógica. Utilizar $V_{REF+} = 4V$ y $V_{REF-} = 1V$. No utilizar interrupciones.

Pantalla de cristal líquido

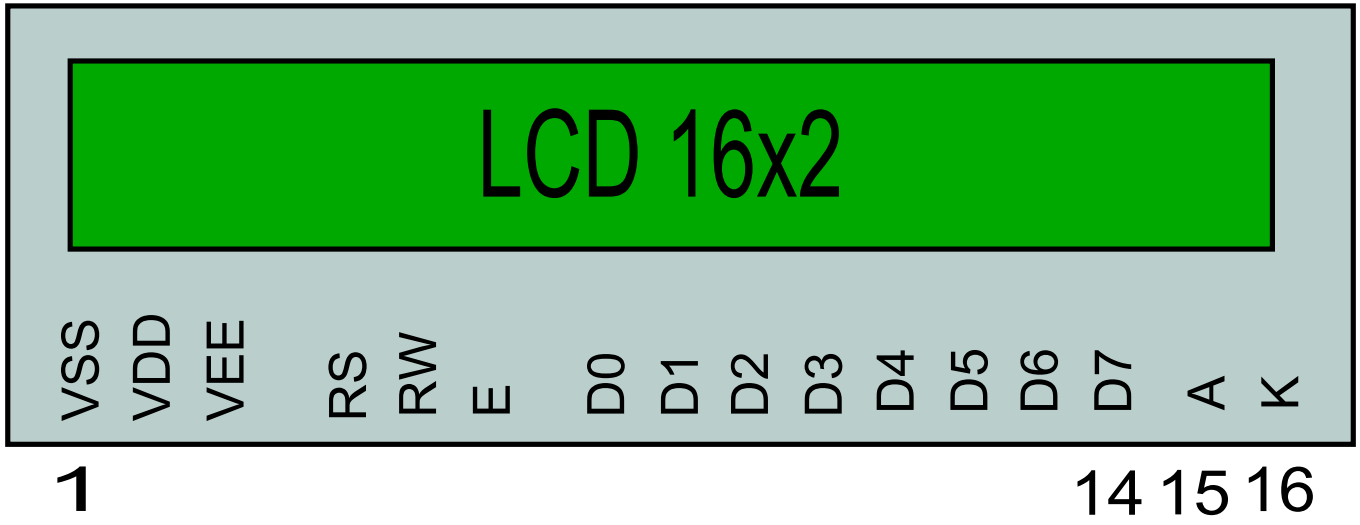


Figura 28: Diagrama de pines del LCD de 16x2.

Pantalla de cristal líquido

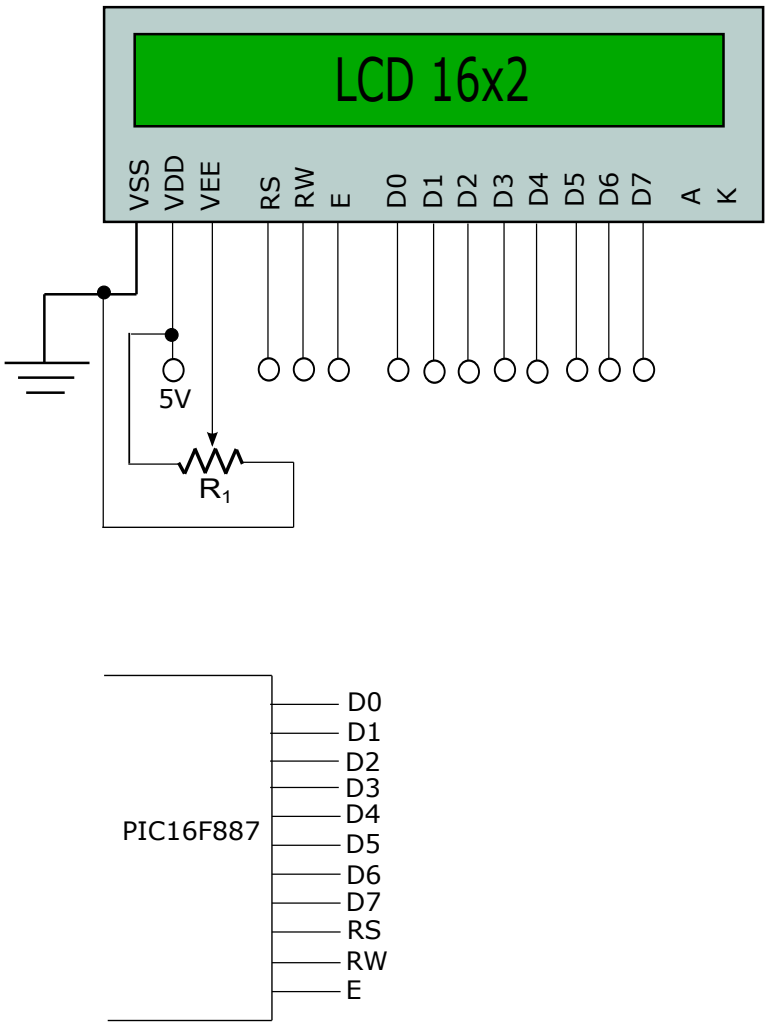


Figura 29: Diagrama de conexión del LCD de 16x2.

Inicialización del LCD (8 bits)

1. Poner en bajo todo el puerto de datos.
2. Configurar el puerto de datos como salida.
3. Configurar todas las señales de control como salida.
4. Poner en bajo todas las señales de control.
5. Espera al menos 15 ms para que se energice el LCD.
6. Configurar: 8-bit, 1-line, 5x7.
7. Espera 4.167 ms.
8. Configurar: 8-bit, 1-line, 5x7.
9. Espera 0.8333 ms.
10. Configurar: 8-bit, 1-line, 5x7.
11. Esperar hasta que el LCD se desocupe.
12. Configurar: 8-bit, 2-line, 5x7.

Inicialización del LCD (8 bits)

13. Esperar hasta que el LCD se desocupe.
14. Configurar: display, cursor y parpadeo desactivado.
15. Esperar hasta que el LCD se desocupe.
16. 13. Configurar: display, cursor y parpadeo activado.
17. Esperar hasta que el LCD se desocupe.
18. Limpiar pantalla.
19. Esperar hasta que el LCD se desocupe.
20. Enviar al cursor al inicio.
21. Esperar hasta que el LCD se desocupe.

Escribir comando (8 bits)

1. Poner en el puerto de datos el valor del comando.
2. Configurar el puerto de datos como salida.
3. Señal de control RW = ESCRIBIR.
4. Señal de control RS = COMANDO.
5. Esperar 1.5 us.
6. Señal de control EN = HABILITADO.
7. Esperar 1.5 us.
8. Señal de control EN = DESHABILITADO.
9. Esperar 1.5 us.
10. Configurar el puerto de datos como entrada.

Escribir dato (8 bits)

1. Poner en el puerto de datos el valor del dato.
2. Configurar el puerto de datos como salida.
3. Señal de control RW = ESCRIBIR.
4. Señal de control RS = DATO.
5. Esperar 1.5 us.
6. Señal de control EN = HABILITADO.
7. Esperar 1.5 us.
8. Señal de control EN = DESHABILITADO.
9. Señal de control RS = COMANDO.
10. Configurar el puerto de datos como entrada.

while(BusyLCD())

1. Señal de control RW = LEER.
2. Señal de control RS = COMANDO.
3. Esperar 1.5 us.
4. Señal de control EN = HABILITADO.
5. Esperar 1.5 us.
6. Checar la bandera de "ocupado":

    Si el LCD está ocupado,

        Señal de control EN = DESHABILITADO;

        Señal de control RW = ESCRIBIR;

        return 1;

    Si el LCD está deocupado,

        Señal de control EN = DESHABILITADO;

        Señal de control RW = ESCRIBIR;

        return 0;

Delay 1.5 us

Checar la bandera de ocupado:

$$\text{if(DATA\_PORT } \& \text{ 0x80)}$$

1: LCD ocupado

0: LCD desocupado

Delay 1.5 us

Escribir: 0

Leer: 1

Dato: 1

Comando: 0

Delay 15ms

```
#include <delays.h>

Delay10KTCYx(18);          // Delay 15 ms.
```

Cálculo del número de ciclos de instrucción:

((Fosc) / 4) / ((1/15ms)) =

(48MHz / 4) / (66.666) =

12MHz / 66.666 = 180,000 ciclos de instrucción.

Delay 1.5 us

```
#include <delays.h>

DelayFor18TCY();          // Delay 1.5 us.
```

Cálculo del número de ciclos de instrucción:

((Fosc) / 4) / ((1/1.5us)) =

(48MHz / 4) / (666666.666) =

12MHz / 0.666MHz = 18 ciclos de instrucción.

Pantalla de cristal líquido

Práctica 4.

a) Mostrar en la primera fila del LCD la frase “Hola mundo”.

b) Mostrar en la primera fila del LCD la frase “Time (sec.):” y en la segunda fila un segundero cíclico que vaya de 00 a 15 (después del 15 regresa a 00 y vuelve a contar).

c) Dibujar una secuencia de “Ping - Pong” con marquesina. Ver video.

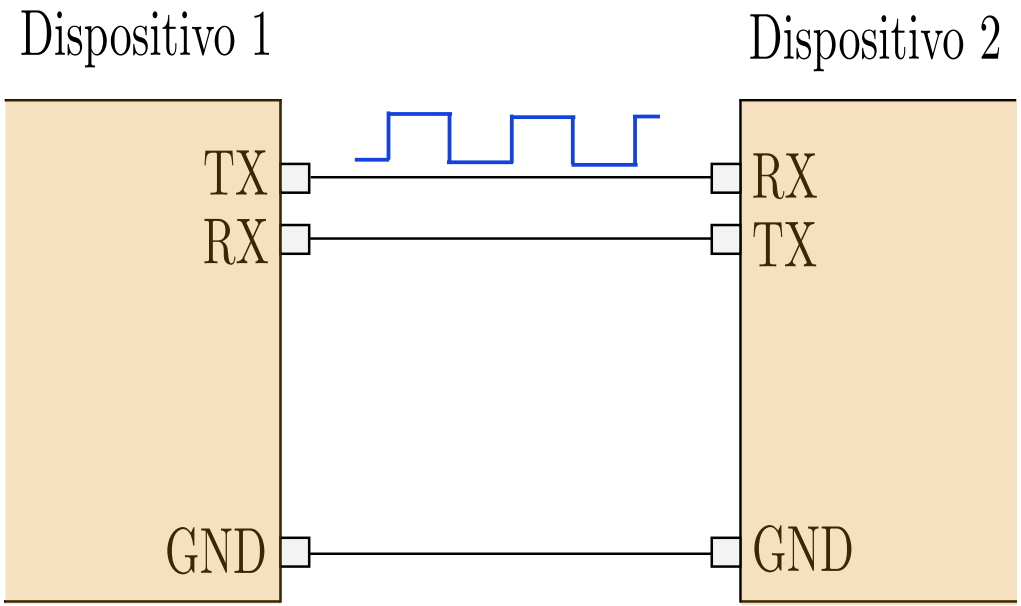
Ayuda: Checar la función *sprintf*.

EUSART

EUSART = Enhanced Universal Synchronous/Asynchronous Receiver/Transmitter

(Transmisor/Receptor Universal Síncrono/Asíncrono Mejorado)

El PIC18F4550 contiene un módulo EUSART que permite comunicar varios dispositivos usando una comunicación serial.



EUSART

El módulo USART puede trabajar de tres maneras:

- Asícrona (Full duplex, bidireccional)
- Síncrona Maestro (Half duplex, unidireccional)
- Síncrona Esclavo (Half duplex, unidireccional)

Asíncrono:

- Los bits entran y salen por dos líneas: TX y RX.
- La frecuencia de los bits es controlada internamente por el USART.

Síncrono:

- La comunicación requiere dos líneas: CK y DT.
- Los bits en la línea DT se trasladan a la frecuencia de los impulsos de reloj que viajan sobre la línea CK.

EUSART

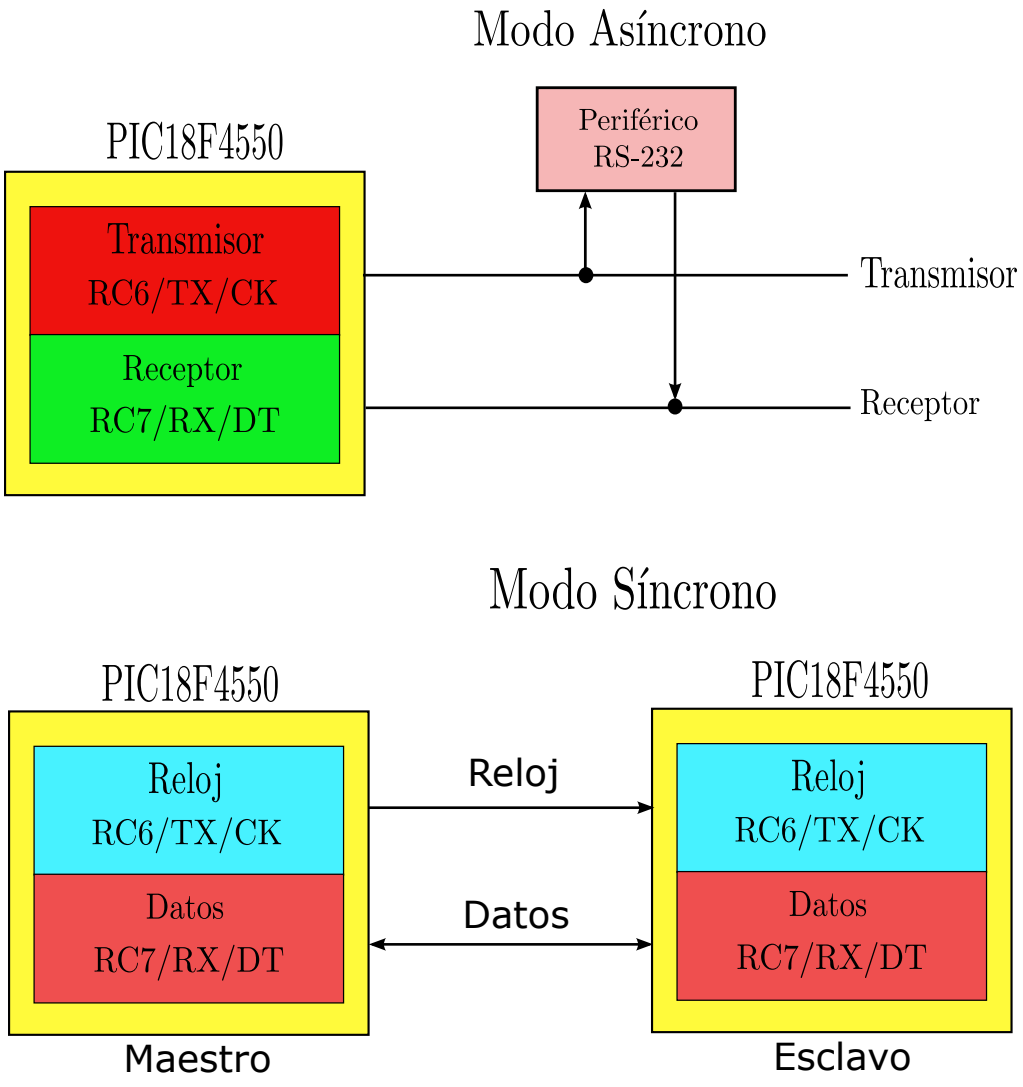


Figura 30: Síncrono vs asíncrono

RS-232-C

El RS-232-C es la norma más popular de comunicación serial asíncrona.

Características:

- Cada palabra de información es independiente de las demás.
- Cada palabra suele ser de 8 o 9 bits.
- Cada palabra está precedida por un bit de INICIO y sucedida por un bit de PARADA.
- Los bits se transfieren a una frecuencia fija y normalizada.

RS-232-C

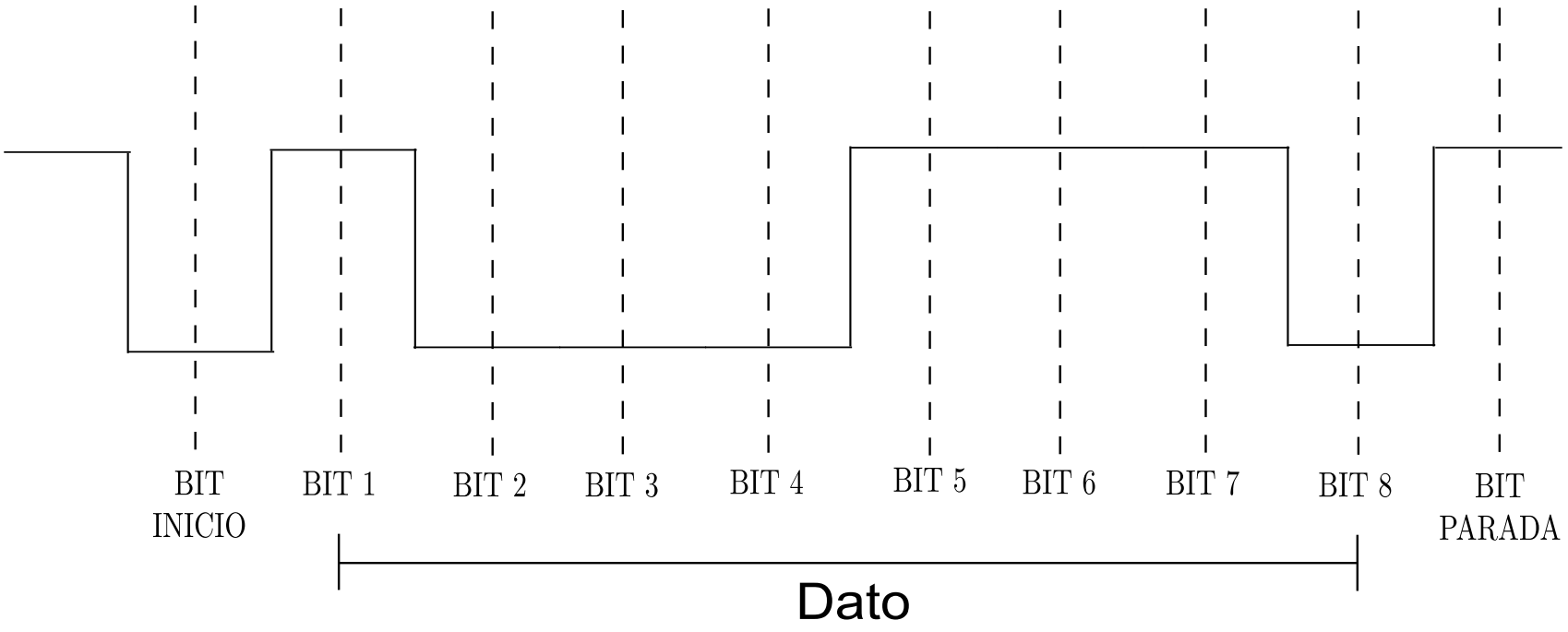


Figura 31: Ejemplo de una cadena de información 8-N-1, correspondiente a la norma RS-232-C.

Bloques principales del USART

Los bloques principales de la arquitectura USART, en modo asincrono, son:

- Circuito de muestreo
- Generador de baudios
- Transmisor asíncrono
- Receptor asíncrono

Generador de baudios

El USART dispone de un generador de frecuencia en baudios, BRG.

En el protocolo asíncrono RS-232-C, la frecuencia en baudios (bits por segundo) se realiza a un valor normalizado:

- 330
- 1200
- 2400
- 9600
- 19200
- 57600
- 115200

Generador de baudios

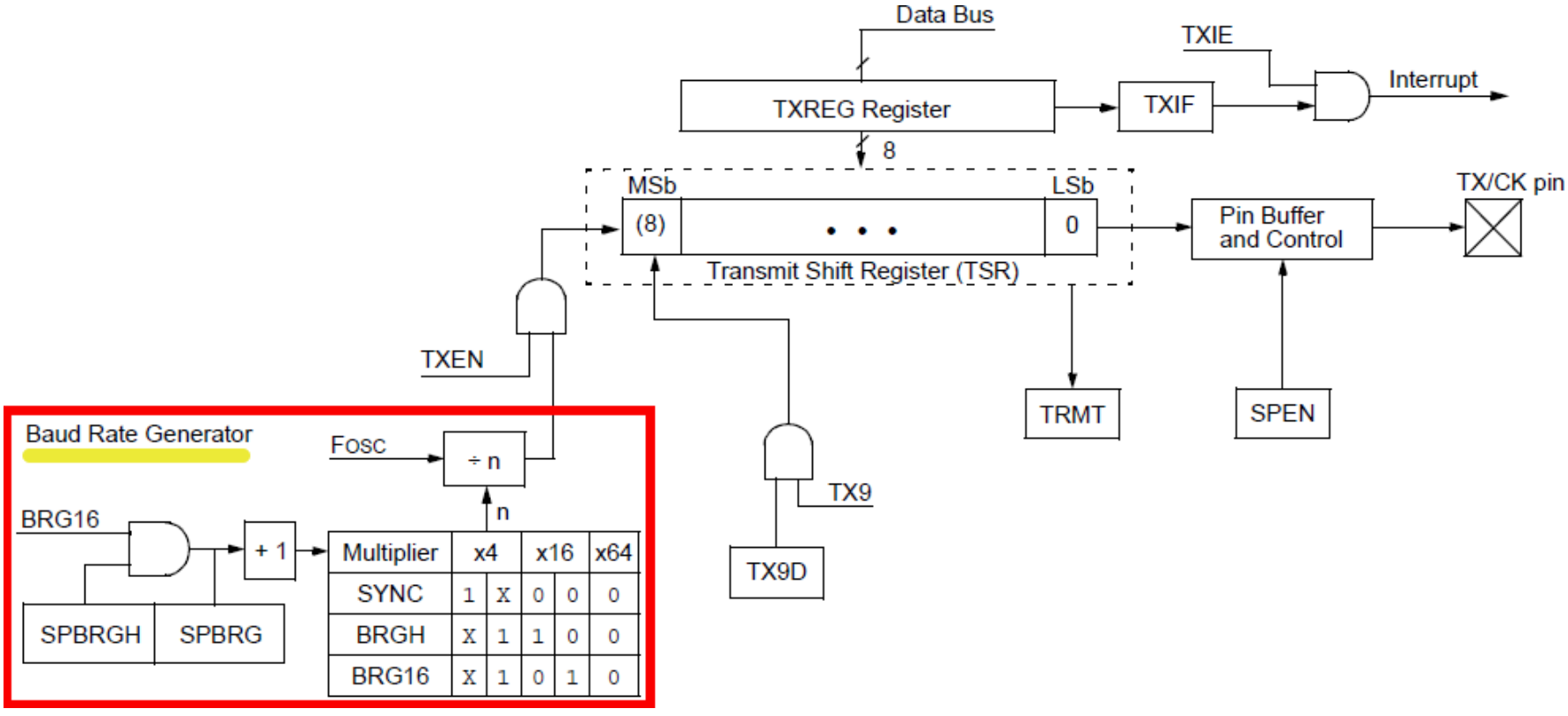


Figura 32: Generador de baudios del EUSART.

Generador de baudios

La frecuencia en baudios de BRG se controla a través de

- el registro SPBRGH,
- el registro SPBRG
- y el bit BRGH del registro TXSTA$< 2 >$.

El generador de baudios BRG puede ser 8 y de 16 bits.

Si BRG es de 16 bits, el valor se carga en SPBRGH:SPBRG.

Si BRG es de 8 bits, el valor se carga en SPBRG.

Generador de baudios

El valor X que debe cargarse en SPBRGH:SPBRG o SPBRG se calcula a partir de:

$$\text{Frecuencia en baudios deseada} = \frac{F_{osc}}{K \cdot (X + 1)} \tag{2}$$

donde X es el valor cargado en el registro SPBRG.

Si BRGH $= 0$ (baja velocidad), K $= 64$

Si BRGH $= 1$ (alta velocidad), K $= 16$

Generador de baudios

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

Figura 33: Fórmulas para la generación de baudios

Generador de baudios

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

Desired Baud Rate = Fosc/(64 ([SPBRGH:SPBRG] + 1))

Solving for SPBRGH:SPBRG:

X = ((Fosc/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615

Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%

Transmisor Asíncrono

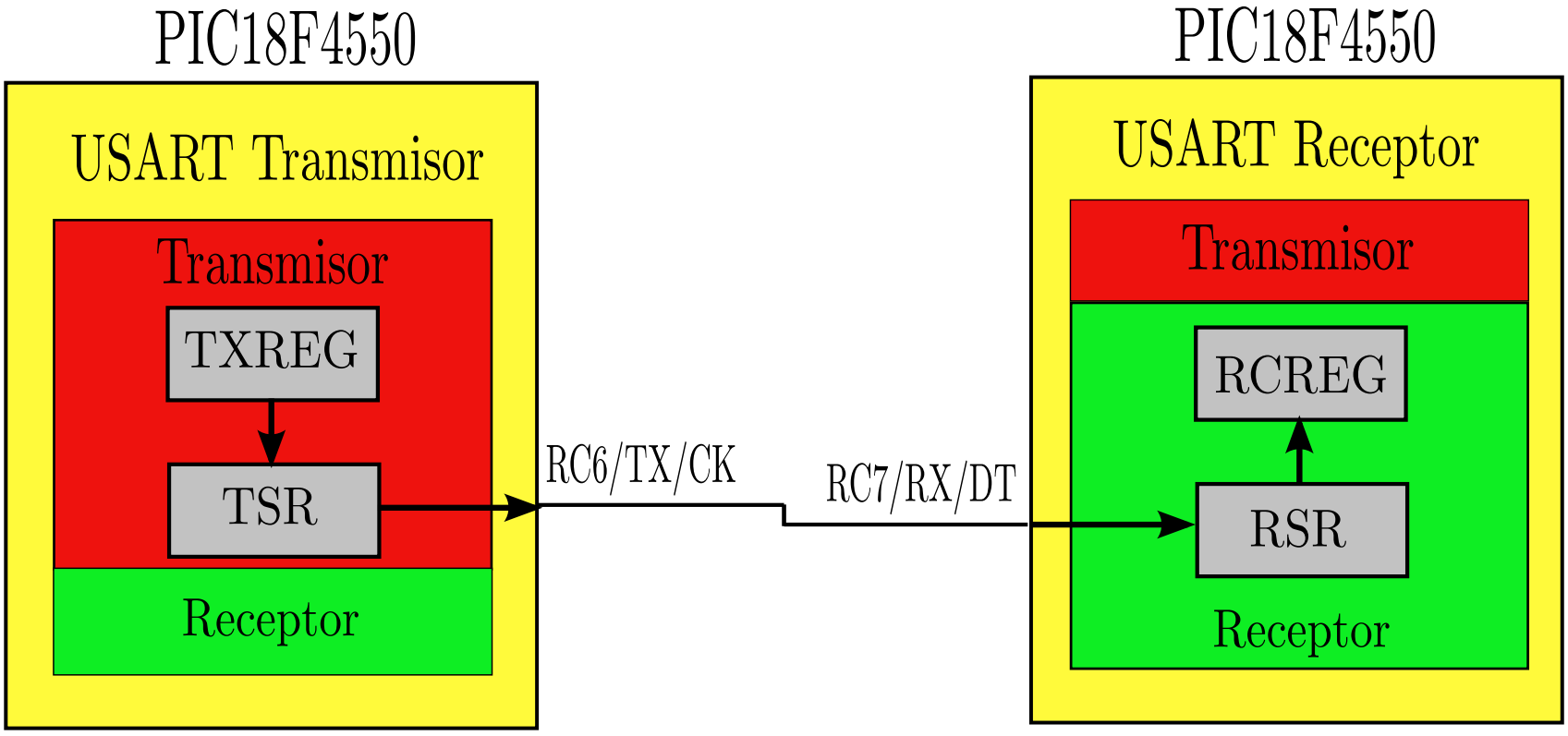


Figura 34: Conexión entre dos módulos USART en modo asíncrono.

Transmisor Asíncrono

USART Transmisor

TXREG recibe el dato que se desea transmitir.

El dato en TXREG se traspasa al registro de desplazamiento TSR, de donde sale bit por bit.

El primer dato en salir es el de menos peso.

Los bits de INICIO y de PARADA se le incluyen al dato que se desea transmitir.

Los bits salen de TSR a la frecuencia establecida.

Transmisor Asíncrono

USART Receptor

El USART receptor recibe, uno a uno, los bits, eliminando los dos de control.

Los bits de información se almacenan en RSR conforme van llegando.

Cuando RSR está lleno, el dato se traslada al registro RCREG quedando disponibles para ser usados.

Transmisor Asíncrono

La bandera TXIF se pone a 1 cuando el dato en TXREG se pasa a TSR.

TXIF se pone a 0 cuando se escribe otro dato sobre TXREG.

El bit TRMT indica vale 1 cuando el registro TSR está vacío.

Transmisor Asíncrono

Implementación de una transmisión 8-N-1 en el USART:

1. Configurar RC6 como salida y RC7 como entrada.
2. Se habilita el puerto serial poniendo SPEN = 1. 3. Poner SYNC = 0 para trabajar en modo asíncrono.
4. Poner TXIE = 0 si no se desea trabajar con interrupción.
5. Poner el bit TX9 = 0 para trabajar con datos de 8 bits.
6. Configurar el generador de baudios, cargando el valor X adecuado a SPBRGH:SPBRG y configurando el bit BRGH.
7. Habilitar la transmisión con TXEN = 1.
8. Poner TXIF = 0.
9. Cargar el dato en TXREG. Comienza a transmitir.

Receptor Asíncrono

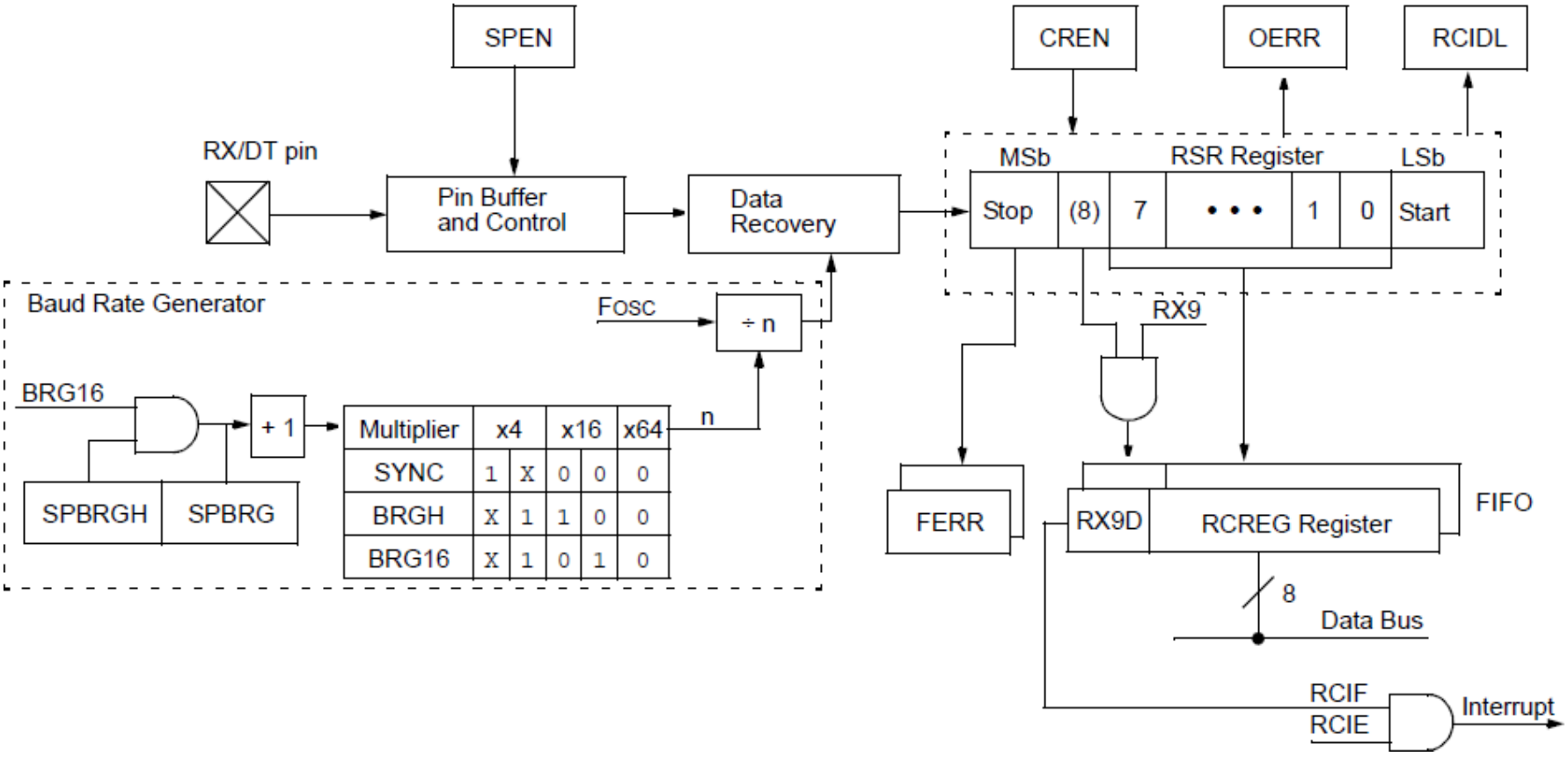


Figura 35: Diagrama de bloques del circuito receptor USART

Receptor Asíncrono

Implementación de una recepción 8-N-1 en el USART:

1. Configurar el generador de baudios, cargando el valor X adecuado a SPBRGH:SPBRG y configurando el bit BRGH.

2. Poner SYNC = 0 para trabajar en modo asíncrono.

3. Se pone RCIE = 1 para interrumpir cuando llegue un nuevo dato.

4. Poner RX9 = 0 para trabajar con datos de 8 bits.

5. Poner CREN = 1 para habilitar la recepción. Inicializar la bandera RCIF = 0.

- Al recibir un dato, el bit RCIF se pone a 1 produciendo una interrupción en caso de haberse permitido.

- Leer los 8 bits del registro RCREG.

Receptor Asíncrono

Revisión de errores:

Leer los bits **FERR** y **OERR** del registro RCSTA.

- FERR: Framing error (no coinciden los protocolos o se tienen distintas velocidades).
- OERR: Overrun error (sobreescritura de datos en el receptor: Se lee más lento de lo que se escribe).

En caso de haberse producido algún error se puede poner CREN = 0, pero lo mejor siempre será detectar la fuente del error y corregirlo.

EUSART

Práctica 5.

a) Enviar y recibir datos usando el protocolo de comunicación RS-232 TTL (se puede utilizar un único PIC). Al apretar un pulsador se envía el caracter 'a'; al recibirse este caracter, se cambia el estado de un LED (ON/OFF).

b) Enviar y recibir datos usando el protocolo de comunicación RS-232 TTL (se puede utilizar un único PIC). Al apretar un pulsador se cambia el estado de un LED (ON/OFF) cumpliendo las siguientes reglas:

- Se envía el string "ON" si se quiere encender el LED,
- se envía el string "OFF" si se quiere apagar el LED
- el LED comienza apagado.

Módulo CCP

El PIC18F4550 tiene dos módulos de Captura/Comparación/PWM o CCP: CCP1 y CCP2.

Estos módulos realizan tres funciones principales:

- **Modo captura**: Temporización de eventos.
- **Modo comparación**: Genera un evento externo al transcurrir un tiempo determinado.
- **Modo modulación de ancho de pulsos (PWM)**: Generación de una señal PWM de frecuencia y ciclo de trabajo variable.

Módulo CCP

Tanto CCP1 como CCP2 se manejan principalmente por medio de dos registros:

- Registro de control (CCPxCON)
- Registro de datos (CCPRx)

El registro de datos, a su vez, está compuesto de dos registros de 8 bits:

- CCPRxL (byte bajo)
- CCPRxH (byte alto)

Módulo CCP

El registro **CCPXCON** controla la función que realizará el módulo CCP.

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.

Figura 36: Registro CCP1CON.

Módulo CCP

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 CCP2 match (CCPxIF bit is set)
11xx = PWM mode

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

Figura 37: Bits del registro CCP1CON.

Módulo CCP

Los registros **CCPRXH** y **CCPRXL** forman el registro de trabajo de 16 bits del módulo CCPX.

CCPR1L	Capture/Compare/PWM Register 1 Low Byte (LSB)
CCPR1H	Capture/Compare/PWM Register 1 High Byte (MSB)
CCPR2L	Capture/Compare/PWM Register 2 Low Byte (LSB)
CCPR2H	Capture/Compare/PWM Register 2 High Byte (MSB)

Figura 38: Registros de trabajo de CCP1 y CCP2.

Módulo CCP

Los módulos CCP utilizan los Timers 1, 2 o 3.

CCP/ECCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2

Figura 39: Timers empleados por los módulos CCP.

Módulo CCP

La asignación de un timer en particular se realiza a través del registro T3CON.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
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 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6, 3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the capture/compare clock source for both CCP modules
01 = Timer3 is the capture/compare clock source for CCP2;
Timer1 is the capture/compare clock source for CCP1
00 = Timer1 is the capture/compare clock source for both CCP modules

Figura 40: Registro T3CON.

Módulo CCP

bit 5-4 **T3CKPS1:T3CKPS0:** Timer3 Input Clock Prescale Select bits

11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **T3SYNC:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)

When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input

When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit

1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR3ON:** Timer3 On bit

1 = Enables Timer3
0 = Stops Timer3

Figura 41: Bits del registro T3CON.

Módulo CCP

Los pines RC1 y RB3 pueden asignarse al módulo CCP2.

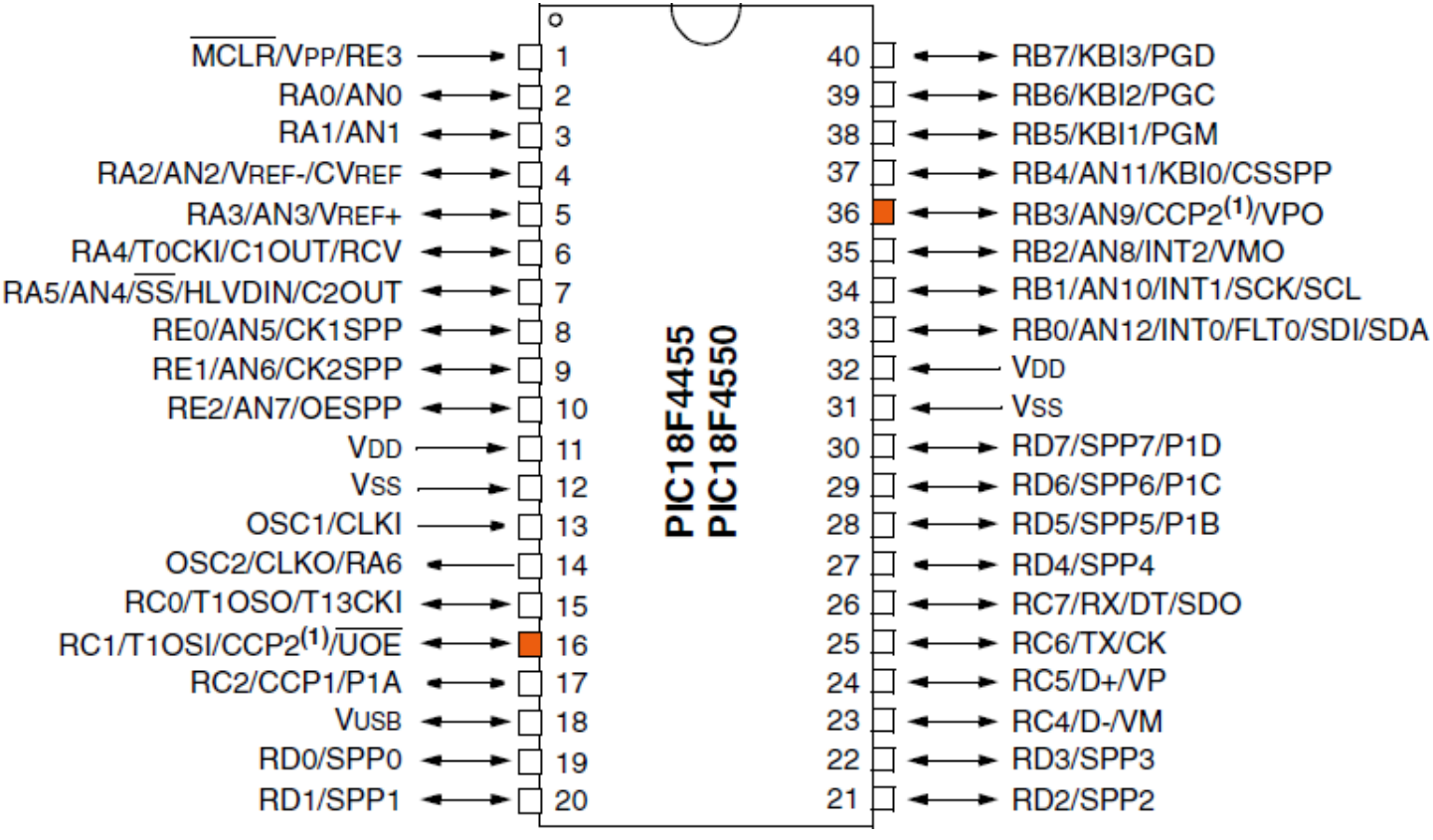


Figura 42: Bits del registro CONFIG3H.

Módulo CCP

La asignación se realiza a través del bit CCP2MX del registro CONFIG3H.

R/P-1	U-0	U-0	U-0	U-0	R/P-0	R/P-1	R/P-1
MCLRE	—	—	—	—	LPT1OSC	PBADEN	CCP2MX
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed		u = Unchanged from programmed state

bit 7 **MCLRE:** $\overline{\text{MCLR}}$ Pin Enable bit
1 = $\overline{\text{MCLR}}$ pin enabled, RE3 input pin disabled
0 = RE3 input pin enabled, $\overline{\text{MCLR}}$ pin disabled

bit 6-3 **Unimplemented:** Read as '0'

bit 2 **LPT1OSC:** Low-Power Timer1 Oscillator Enable bit
1 = Timer1 configured for low-power operation
0 = Timer1 configured for higher power operation

bit 1 **PBADEN:** PORTB A/D Enable bit
(Affects ADCON1 Reset state. ADCON1 controls PORTB<4:0> pin configuration.)
1 = PORTB<4:0> pins are configured as analog input channels on Reset
0 = PORTB<4:0> pins are configured as digital I/O on Reset

bit 0 **CCP2MX:** CCP2 MUX bit
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RB3

Figura 43: Bits del registro CONFIG3H.

Modo de captura

El modo de captura registra la ocurrencia de un evento a través del pin CCPx.

En modo de captura, cuando se registra un evento en el pin CCPx, los registros CCPXH y CCPXL capturan el valor de 16 bits del registro TMR1 o TMR3.

Una captura puede ser detonada

- cada flanco de bajada,
- cada flanco de subida,
- cada cuarto flanco de subida, o
- cada décimo sexto flanco de subida.

Modo de captura

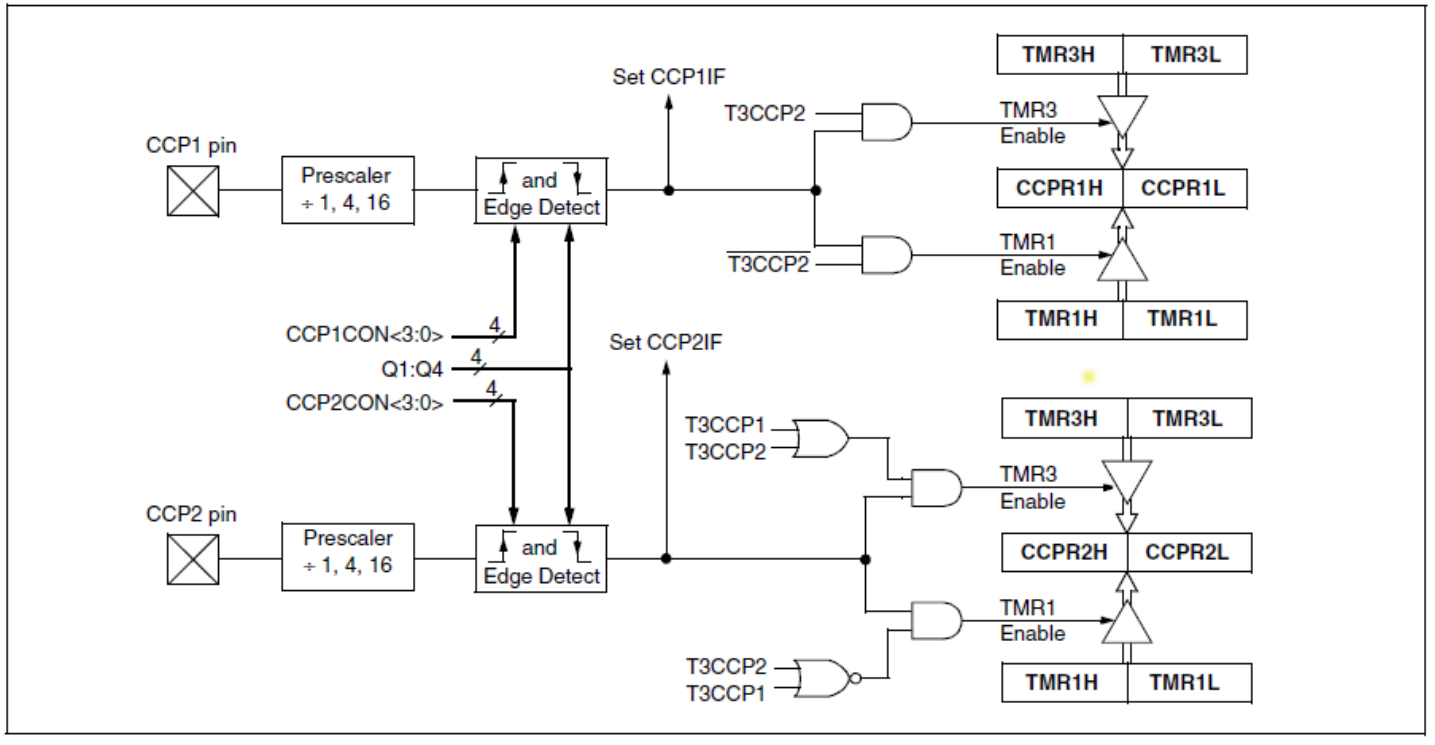


Figura 44: Diagrama de bloques del módulo de captura.

Modo de captura

El evento se selecciona a través de los bits CCPMx3:CCPxM0 del registro CCPxCON.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

bit 7-6 **P1M<1:0>:** PWM Output Configuration bits
If CCP1M<3:2> = 00, 01, 10:
xx = P1A assigned as Capture/Compare input; P1B, P1C, P1D assigned as port pins
If CCP1M<3:2> = 11:
00 = Single output; P1A modulated; P1B, P1C, P1D assigned as port pins
01 = Full-Bridge output forward; P1D modulated; P1A active; P1B, P1C inactive
10 = Half-Bridge output; P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins
11 = Full-Bridge output reverse; P1B modulated; P1C active; P1A, P1D inactive

bit 5-4 **DC1B<1:0>:** PWM Duty Cycle Least Significant bits
Capture mode:
Unused.
Compare mode:
Unused.
PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPR1L.

bit 3-0 **CCP1M<3:0>:** ECCP Mode Select bits
0000 = Capture/Compare/PWM off (resets ECCP module)
0001 = Unused (reserved)
0010 = Compare mode, toggle output on match (CCP1IF bit is set)
0011 = Unused (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, set output on match (CCP1IF bit is set)
1001 = Compare mode, clear output on match (CCP1IF bit is set)
1010 = Compare mode, generate software interrupt on match (CCP1IF bit is set, CCP1 pin is unaffected)
1011 = Compare mode, trigger special event (CCP1IF bit is set; CCP1 resets TMR1 or TMR2
1100 = PWM mode; P1A, P1C active-high; P1B, P1D active-high
1101 = PWM mode; P1A, P1C active-high; P1B, P1D active-low
1110 = PWM mode; P1A, P1C active-low; P1B, P1D active-high
1111 = PWM mode; P1A, P1C active-low; P1B, P1D active-low

Figura 45: Bits del registro CCPxCON que sirven para seleccionar el evento que detona una captura.

Modo de captura

Cuando una captura se realiza, la bandera CCPxIF se pone a 1.

Si ocurre una nueva captura antes de leer el registro CCPRx, el valor anterior será sobreescrito por el nuevo valor.

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Figura 46: Bit CCP1IF del registro PIR1.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIF	CMIF	USBIF	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF
bit 7							bit 0

Figura 47: Bit CCP2IF del registro PIR2.

Modo de captura

Modo	**pin CCPx**
Captura	Entrada
Comparación	Salida
PWM	Salida

Cuadro 2: Estado del pin CCPx de acuerdo al modo de trabajo del módulo CCP.

Modo de captura

El Timer1 debe correr en modo de temporizador o como contador síncrono.

Cuando se cambia el modo de operación, puede llegar a ocurrir una interrupción falsa.

En estos casos, conviene desactivar las interrupciones o el módulo CCP y borrar la bandera CCPxIF después de cualquier cambio en el modo de operación.

Al desactivar el módulo CCP, el contador del prescalador se borra.

Cambiar de un prescalador a otro no borra el valor del contador y puede generar una interrupción falsa. Se debe desactivar el módulo antes de cambiar el prescalador.

Modo captura

Práctica 7. Obtener la frecuencia y ciclo de trabajo de una señal cuadrada.

Modo comparación

En este modo, los registros CCPRxH-L comparan continuamente su contenido con el valor del TMR1.

Cuando ambos valores coinciden, la patita CCPx realiza uno de los siguientes eventos:

- Cambia su estado (toggle).
- Pasa a nivel alto.
- Pasa a nivel bajo.
- Genera un disparo especial.
- No cambia su estado pero se produce una interrupción.

En este modo, la patita CCPx debe configurarse como salida.

El TMR1 debe trabajar en modo temporizador o contador síncrono.

Modo comparación

Sin importar que modo de comparación se elija, la bandera CCPxIF se pone a 1 al coincidir el valor de TMR1 con el de CCPR1H-L.

Si el bit de permiso de interrupción está a 1, se genera una petición de interrupción.

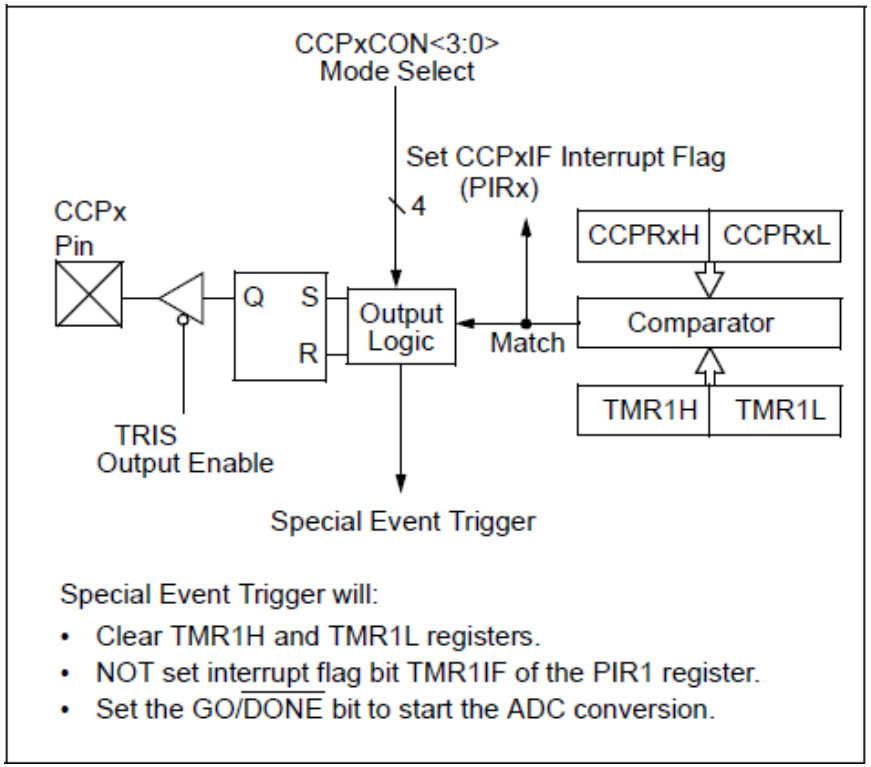


Figura 48: Diagrama de bloques del módulo de comparación.

Modo de comparación

Cuando se escoge el evento de "disparo especial", el módulo CCPx realiza lo siguiente:

- Pone a 0 el TMR1
- Comienza una conversión ADC si el ADC está habilitado.

El estado de la patita CCPx no se modifica cuando se genera un disparo especial.

El disparo especial sirve para realizar conversiones de analógico a digital de manera periódica sin el control del programa de instrucciones.

Modo de comparación

Práctica

1) Generar una señal cuadrada con frecuencia de 1 kHz y ciclo de trabajo del 60 %.

2) Realiza una conversión de analógico a digital de manera periódica usando el disparo especial del módulo de comparación.

Modo PWM

En modo PWM, el pin CCPx proporciona una señal cuadrada con ancho de pulso modulado.

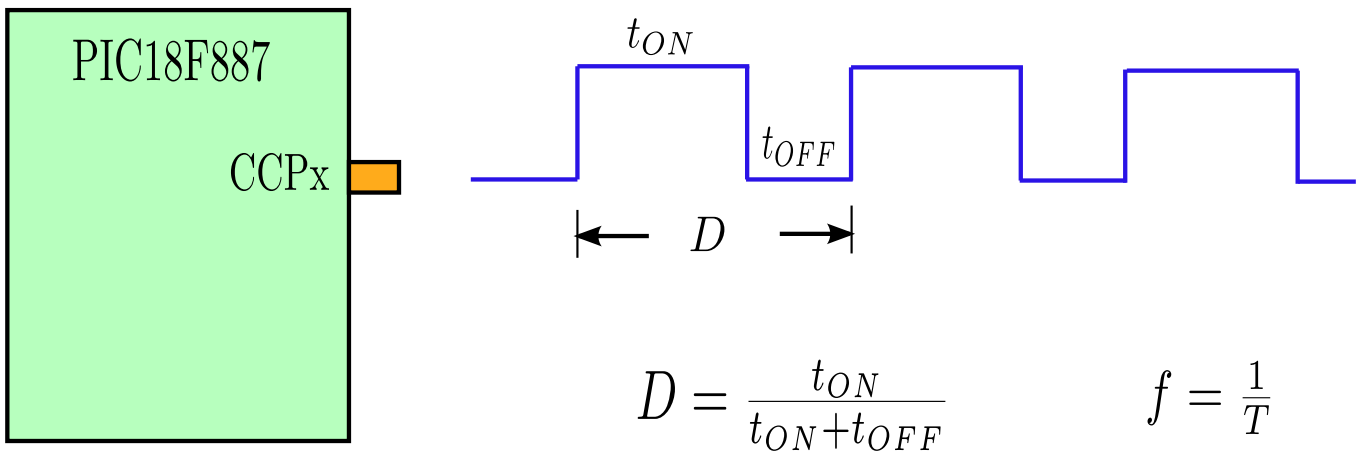


Figura 49: El módulo CCP cuenta con la posibilidad de generar señales de ancho de pulso modulado.

Modo PWM

El periodo, el ciclo de trabajo y la resolución se configuran a través de los siguientes registros:

- PR2
- T2CON
- CCPRxL
- CCPxCON

La resolución se relaciona con el número de bits que pueden usarse para indicar el valor de PWM.

Por ejemplo, con 10 bits, podemos escoger hasta 2^10 posibles valores de PWM.

Periodo del PWM

La forma de calcular la frecuencia de la señal PWM es la siguiente[1]:

$$\text{PWM Frec.} = \frac{F_{\text{OSC}}}{(PR2 + 1) \cdot 4 \cdot \text{Presc.}} \tag{3}$$

PWM Frec. = Frecuencia de la señal PWM (Hz).

F_{OSC} = Frecuencia de reloj.

PR2 = Registro de 8 bits del PIC18F4550.

Presc. = Prescalador del TIMER 2.

[1] El postescalador de TMR2 no afecta el periodo del PWM.

Periodo del PWM

Ejemplo. Use la ec. (3) para calcular el valor que debe tener el registro PR2 si se desea que la frecuencia del PWM sea de 10 KHz, el prescalador del TIMER 2 es igual a 16, $F_{OSC} = 48$ MHz.

$$PR2 = \frac{F_{OSC}}{4 \cdot \text{Presc} \cdot \text{PWM Frec.}} - 1$$

$$= \frac{48 \times 10^6}{4 \cdot 16 \cdot 10 \times 10^3} - 1$$

$$= 75 - 1$$

$$= 74$$

$$= 0\text{x}4\text{A}$$

Ciclo de trabajo

El ciclo de trabajo se define a través de los registros CCPRxL y los bits CCPxCON<5:4>.

Dichos registros pueden ser escritos en cualquier momento.

Esto debido a que el valor en dichos registros no se carga en el registro CCPRxH sino hasta que se complete el periodo.

$$t_{ON} = \frac{(\text{CCPRxL:CCPxCON} < 5{:}4 >) \cdot (\text{TMR2 Prescaler})}{F_{OSC}} \tag{4}$$

$$D = \frac{\text{CCPRxL:CCPxCON} < 5{:}4 >}{4 \cdot (\text{PR2} + 1)} \tag{5}$$

Ciclo de trabajo

Ejemplo. Use la ec. (4) para calcular el valor que deben tener los registros (CCPRxL:CCPxCON $<$ 5:4 $>$) si el ciclo de trabajo es del 40 %, el prescalador del TIMER 2 es igual a 16, $F_{OSC} = 48$ MHz y la frecuencia de la señal PWM es igual a 10 kHZ.

Solución.

Primero encontramos el valor de t_{ON}:

$$\begin{aligned} t_{ON} &= D * (t_{ON} + t_{OFF}) \\ &= D * \text{PWM Period} \\ &= 0.4 * \left(\frac{1}{10000}\right) \\ &= \frac{0.4}{10000} = 40\mu s \end{aligned}$$

Ciclo de trabajo

Ya con el valor de t_{ON}, despejamos de la ecuación (4) el valor que debe cargársele a los registros (CCPRxL:CCPxCON $<$ 5:4 $>$):

$$(\text{CCPRxL:CCPxCON} < 5{:}4 >) = \frac{t_{ON} \cdot F_{OSC}}{\text{TMR2 Prescaler}}$$

$$= \frac{(40\mu s) \cdot (48MHz)}{16}$$

$$= \frac{1920}{16}$$

$$= 120$$

Configuración para modo PWM

1. **OpenTimer2(Timer2Config)**

a. Configura el prescalador del TIMER 2.

b. Enciende el temporizador.

2. **OpenPWM1(unsigned char PR2Value)**

a. Configurar el registro CCP1CON para usar P1A para generar la señal PWM.

b. Configurar el pin CCPx como salida.

c. Detener el TIMER 2.

d. Cargar la frecuencia de la señal PWM a través del registro PR2.

e. Comenzar el TMR2.

3. **SetDCPWM1(unsigned int DC)**

a. Cargar el ciclo de trabajo de la señal PWM en los registros CCPR1L y CCP1CON<5:4>.

Modo PWM

REGISTER 16-1: CCP1CON: ECCP CONTROL REGISTER (40/44-PIN DEVICES)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
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 **P1M1:P1M0:** Enhanced PWM Output Configuration bits

If CCP1M3:CCP1M2 = 00, 01, 10:

xx = P1A assigned as Capture/Compare input/output; P1B, P1C, P1D assigned as port pins

If CCP1M3:CCP1M2 = 11:

00 = Single output: P1A modulated; P1B, P1C, P1D assigned as port pins

01 = Full-bridge output forward: P1D modulated; P1A active; P1B, P1C inactive

10 = Half-bridge output: P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins

11 = Full-bridge output reverse: P1B modulated; P1C active; P1A, P1D inactive

Figura 50: Registro CCP1CON empleado para configurar el módulo PWM del PIC18F4550.

Modo PWM

bit 5-4 **DC1B1:DC1B0**: PWM Duty Cycle Bit 1 and Bit 0

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are fou
in CCPR1L.

bit 3-0 **CCP1M3:CCP1M0**: Enhanced CCP Mode Select bits

0000 = Capture/Compare/PWM off (resets ECCP module)
0001 = Reserved
0010 = Compare mode, toggle output on match
0011 = Capture mode
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 CCP1 pin low, set output on compare match (set CCP1IF)
1001 = Compare mode, initialize CCP1 pin high, clear output on compare match (set CCP1IF)
1010 = Compare mode, generate software interrupt only, CCP1 pin reverts to I/O state
1011 = Compare mode, trigger special event (CCP1 resets TMR1 or TMR3, sets CCP1IF bit)
1100 = PWM mode: P1A, P1C active-high; P1B, P1D active-high
1101 = PWM mode: P1A, P1C active-high; P1B, P1D active-low
1110 = PWM mode: P1A, P1C active-low; P1B, P1D active-high
1111 = PWM mode: P1A, P1C active-low; P1B, P1D active-low

Figura 51: Bits del registro CCP1CON empleado para configurar el módulo PWM del PIC18F4550.

PWM

```
#define USE_AND_MASKS

#include <plib/timers.h>
#include <plib/pwm.h>

void main(void)
{
    unsigned char Timer2Config = T2_PS_1_16;
    OpenTimer2(Timer2Config);
    OpenPWM1(0xAA);
    SetDCPWM1(511);

    while(1);
}
```

PWM

Práctica.

a) Generar una señal PWM de 5 KHz con DC $= 60\ \%$.

b) Generar una señal PWM de 2 KHz cuyo ciclo de trabajo aumente cada medio segundo de 0 a 100 en intervalos de 5. Al llegar a 100, el ciclo de trabajo debe disminuir con la misma tasa de cambio hasta llegar nuevamente a cero. El proceso debe repetirse indefinidamente.

c) Tocar las notas Do - Do# - Re - Re# - Mi - Fa - Fa# - Sol - Sol# - La - La# - Si con 12 pulsadores diferentes, uno para cada nota.

d) Controlar la posición de un servomotor con el PIC18F4550.

TSOP1738

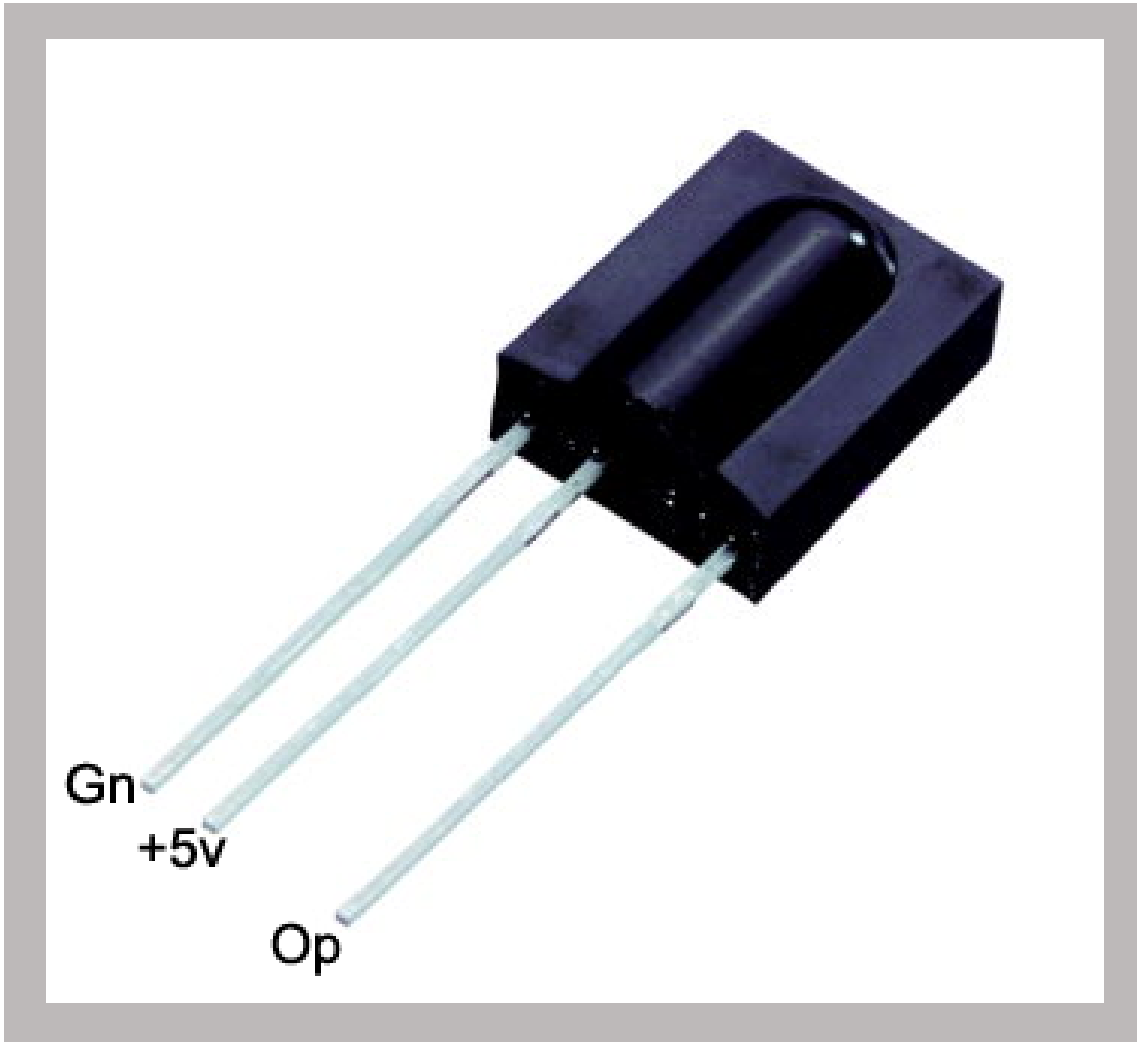


Figura 52: TSOP1738 (pines).

TSOP1738

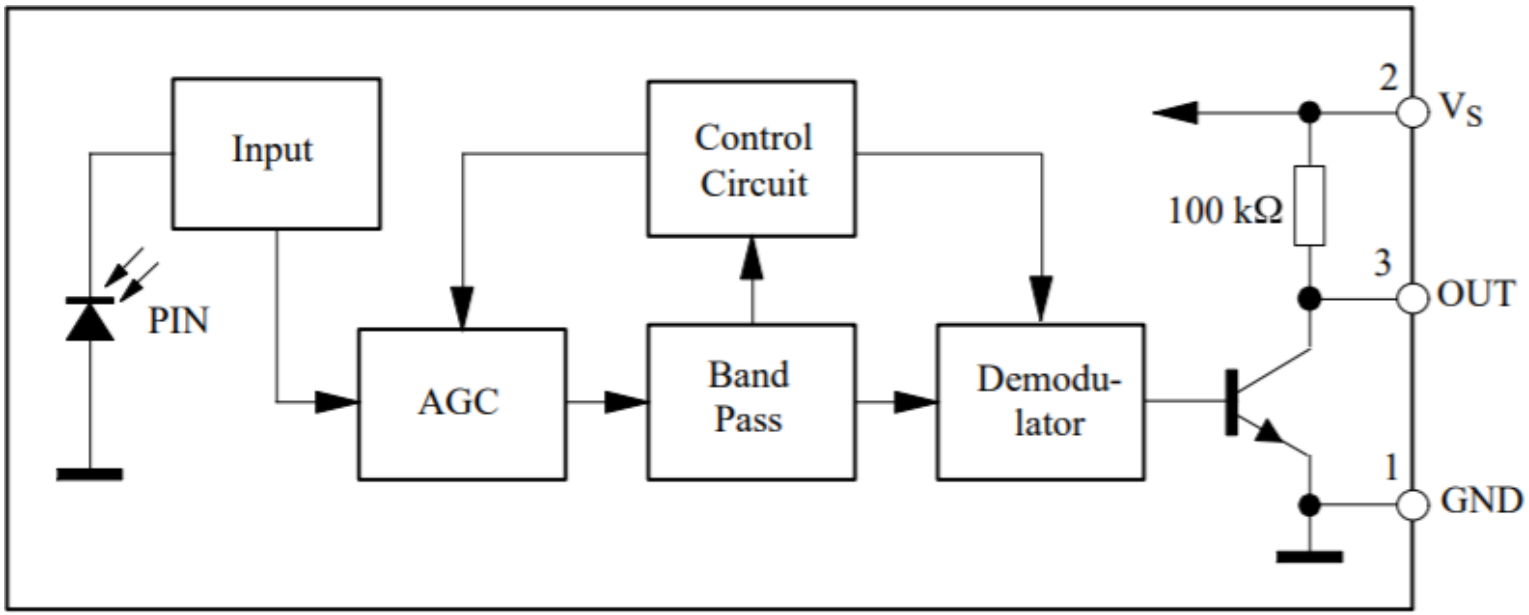


Figura 53: Diagrama de bloques del TSOP1738.

TSOP1738

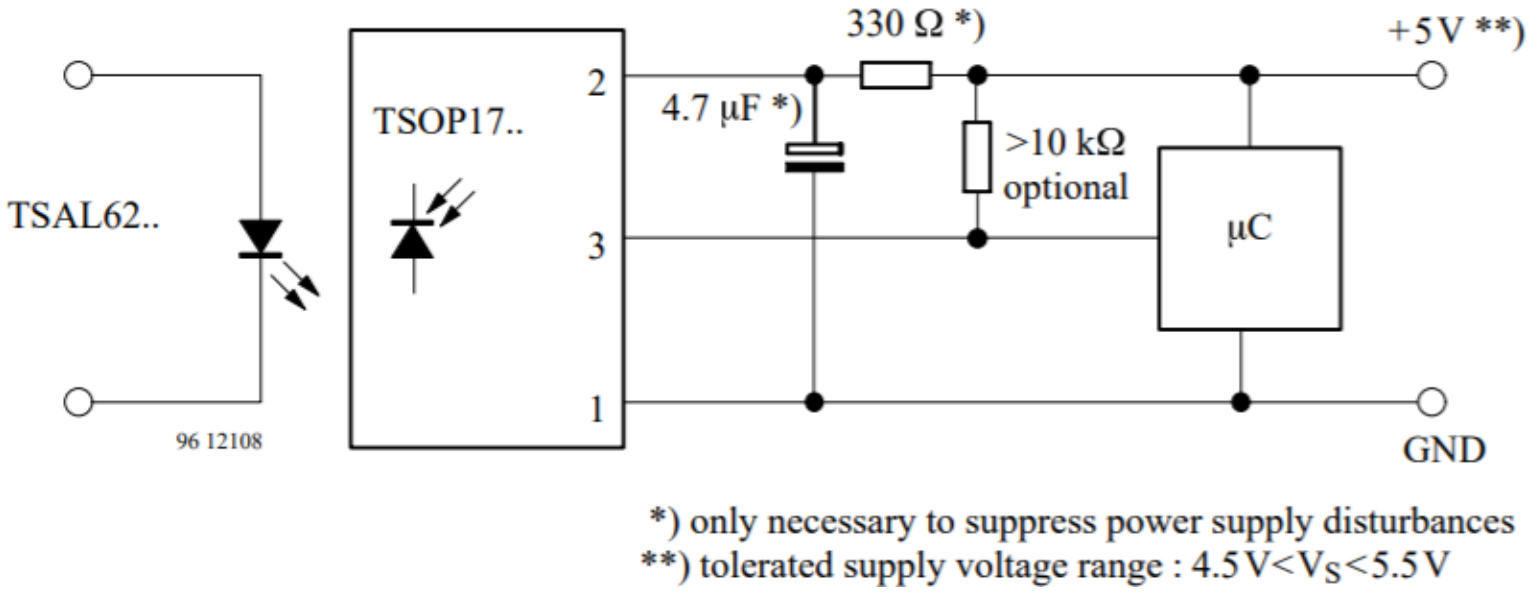


Figura 54: Diagrama eléctrico sugerido para el TSOP1738.

TSOP1738

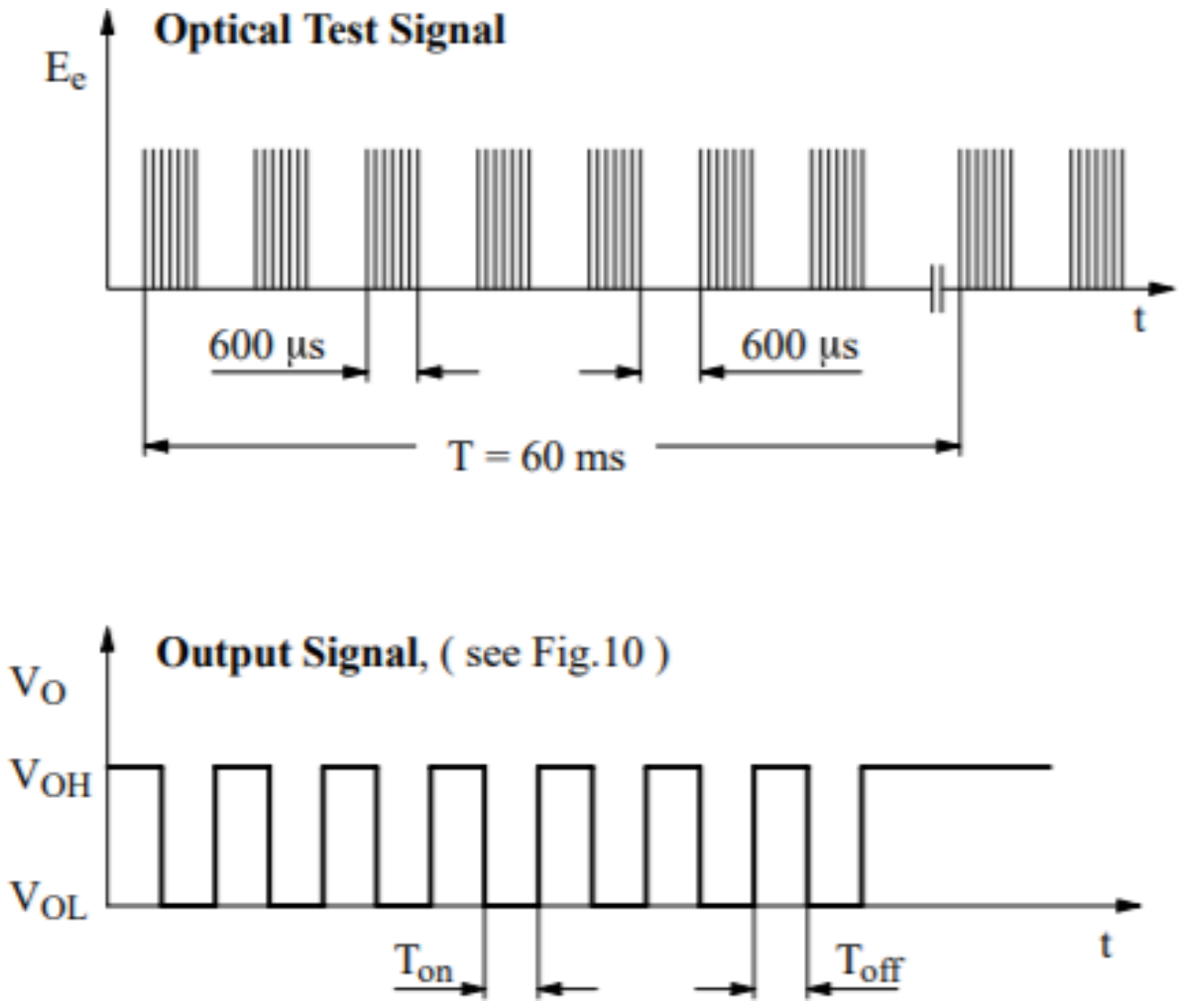


Figura 55: Salida del TSOP1738 para una señal IR de entrada.

I2C: Conexión al bus

El bus I2C permite crear una red de dispositivos, algunos de ellos funcionarán como *maestros* y otros como *esclavos*.

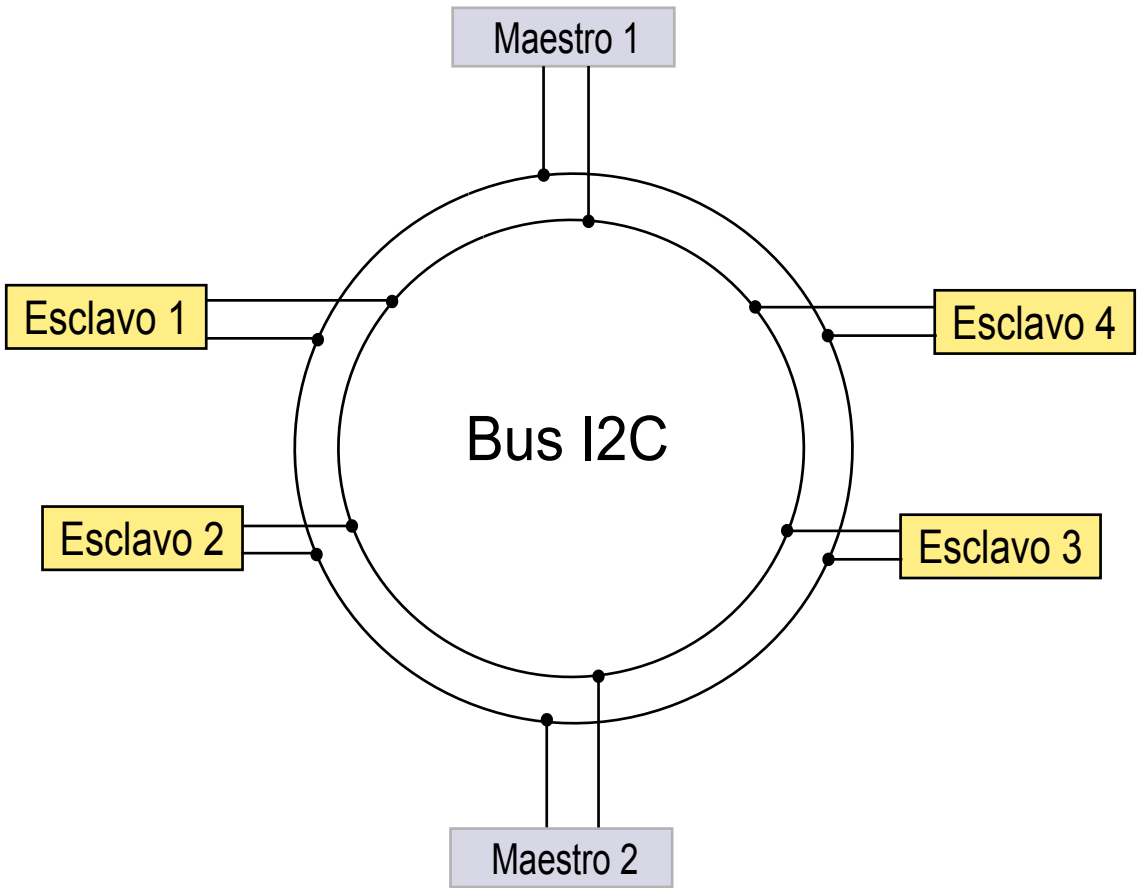


Figura 56: Conexión al bus I2C.

I2C: Conexión al bus

El bus I2C permite crear una red de dispositivos, algunos de ellos funcionarán como *maestros* y otros como *esclavos*.

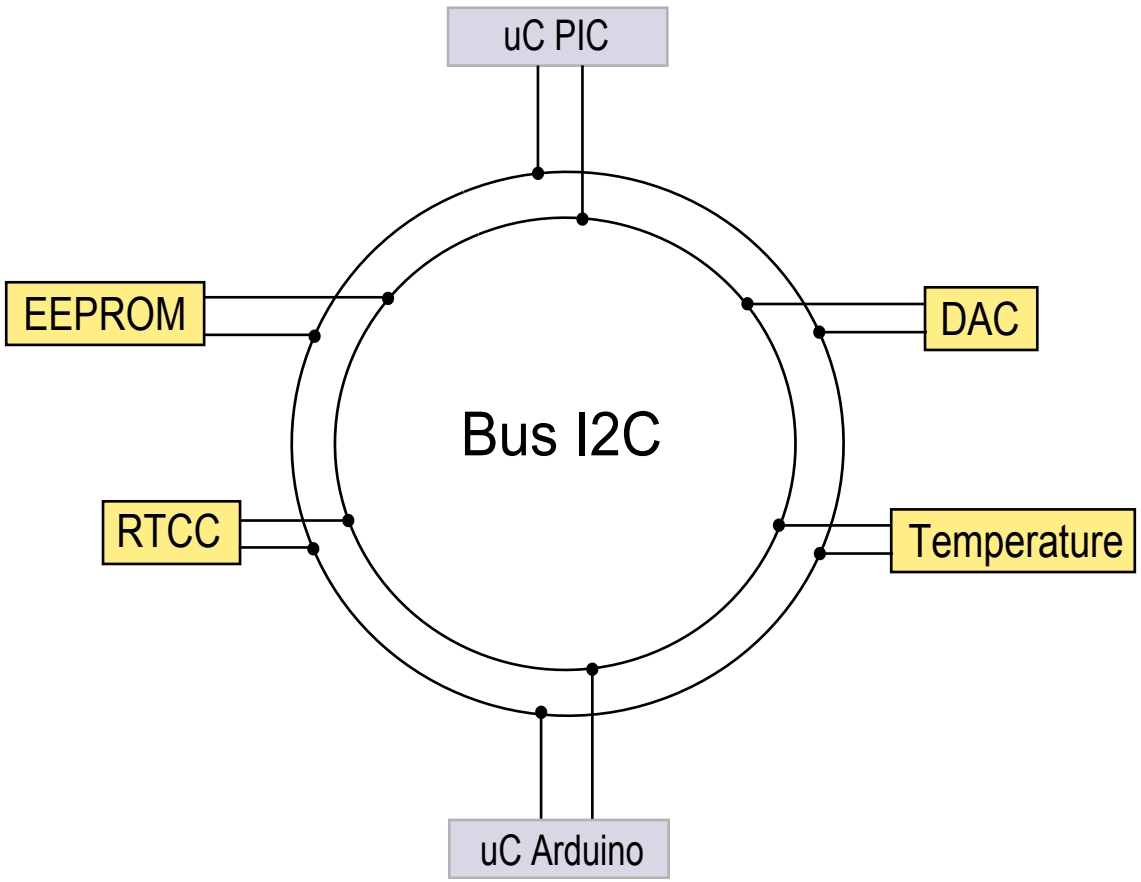


Figura 57: Ejemplo de conexión al bus I2C.

I2C: Conexión al bus

Dos líneas, SDA (datos) y SCL (reloj), son las que transportan la información entre los diferentes dispositivos conectados al bus.

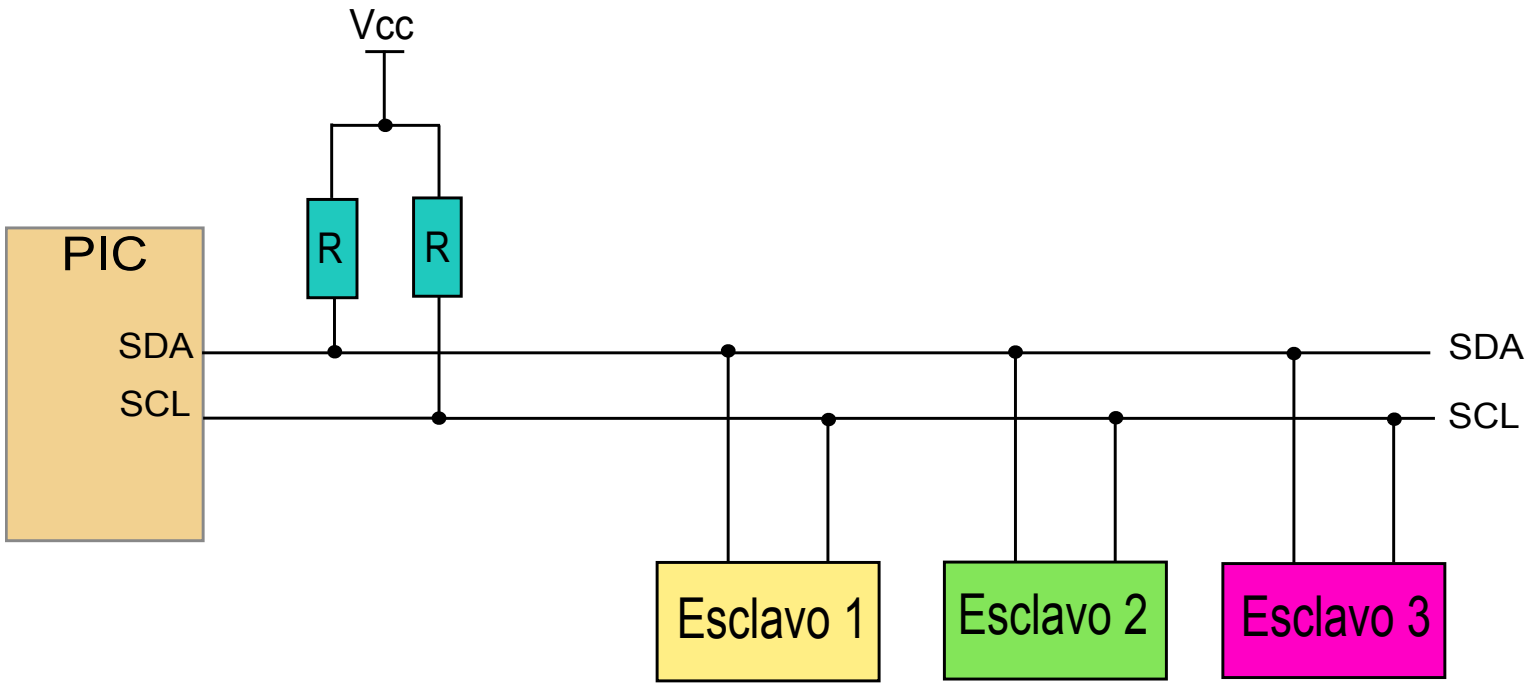


Figura 58: Red I2C.

I2C: Conexión al bus

Las líneas SDA y SCL son bidireccionales y se conecta a $+V_{dd}$ mediante resistencias de *pull-up*.

Cuando el bus está libre, ambas líneas están a nivel lógico "1".

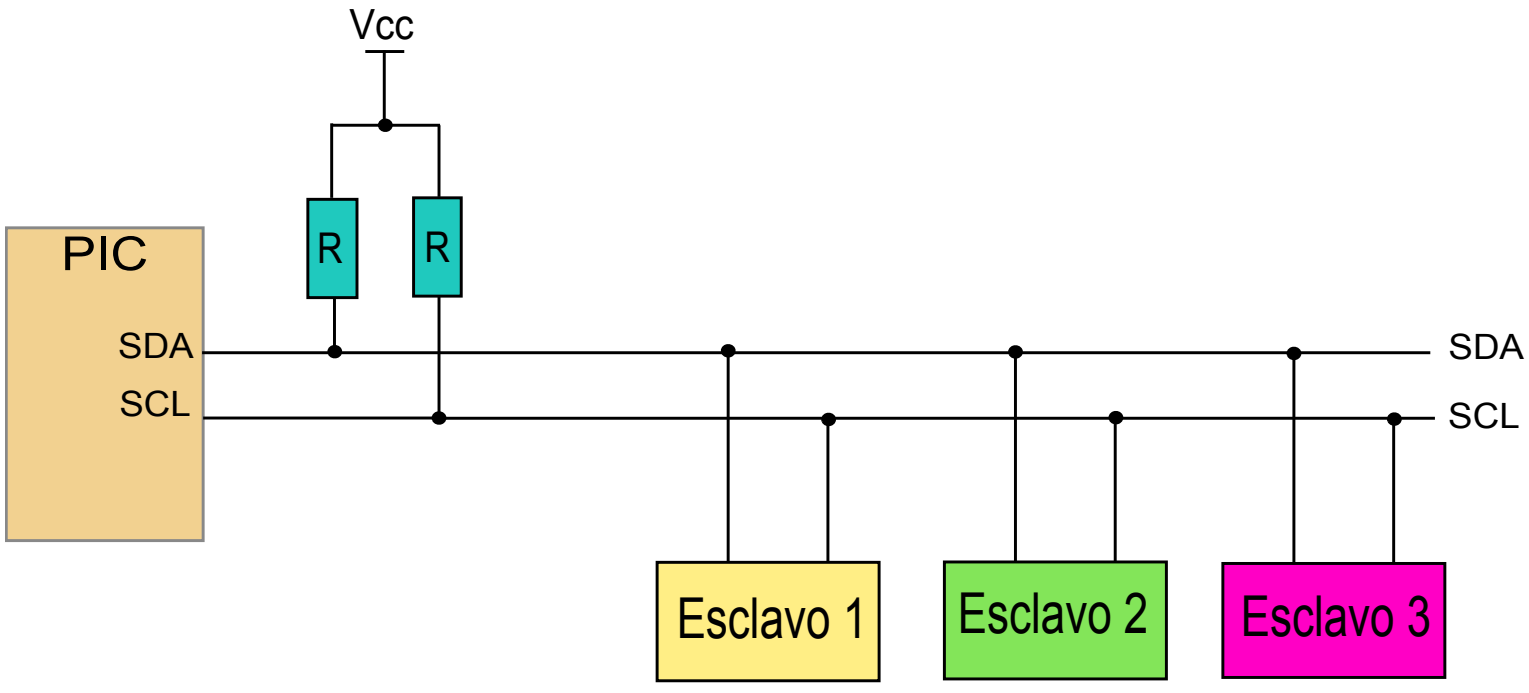


Figura 59: Conexión de SDA y SCL al bus.

I2C: Conexión tipo AND del bus

Todos los dispositivos del bus mantienen una conexión tipo AND entre sí.

Todos los dispositivos en el bus tienen que usar pines de colector abierto.

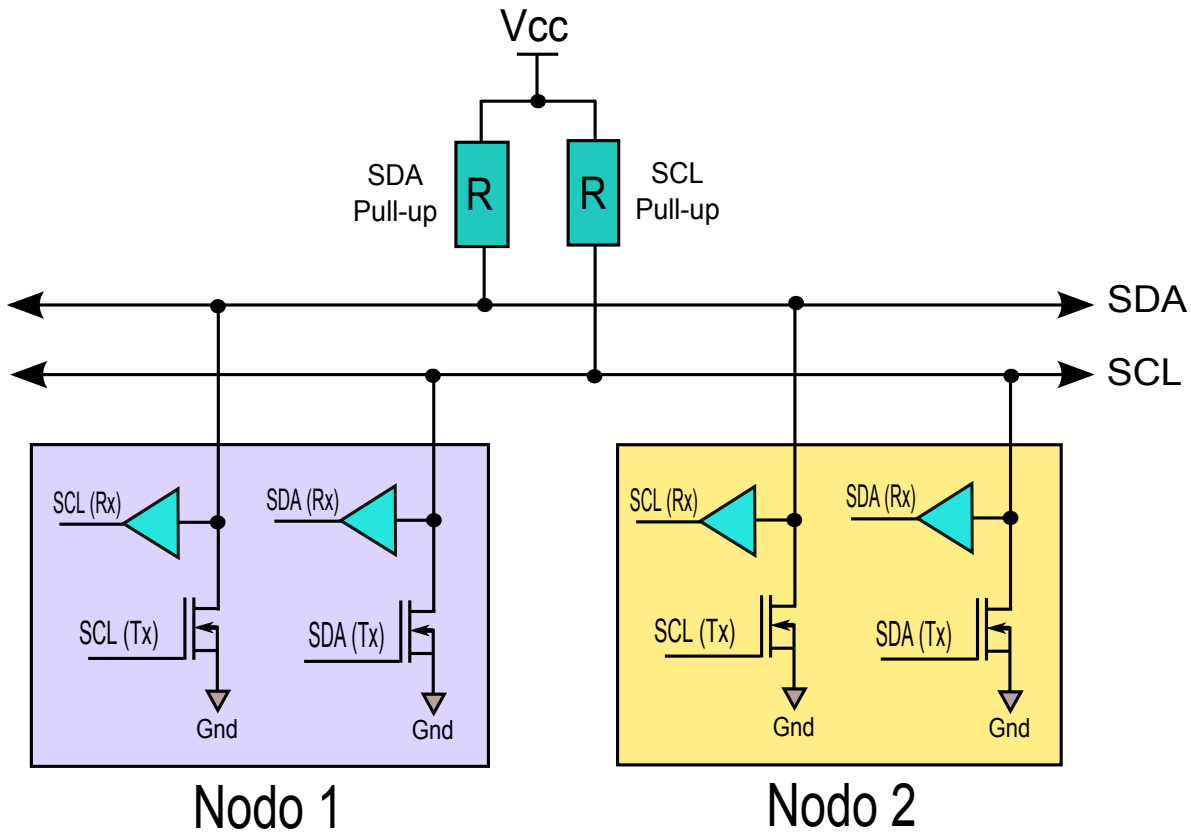


Figura 60: Conexión tipo AND del bus.

I2C: Conexión tipo AND del bus

Para una salida a colector abierto como la mostrada en la figura **??**, la salida será alta siempre, excepto cuando la entrada es 5 V.

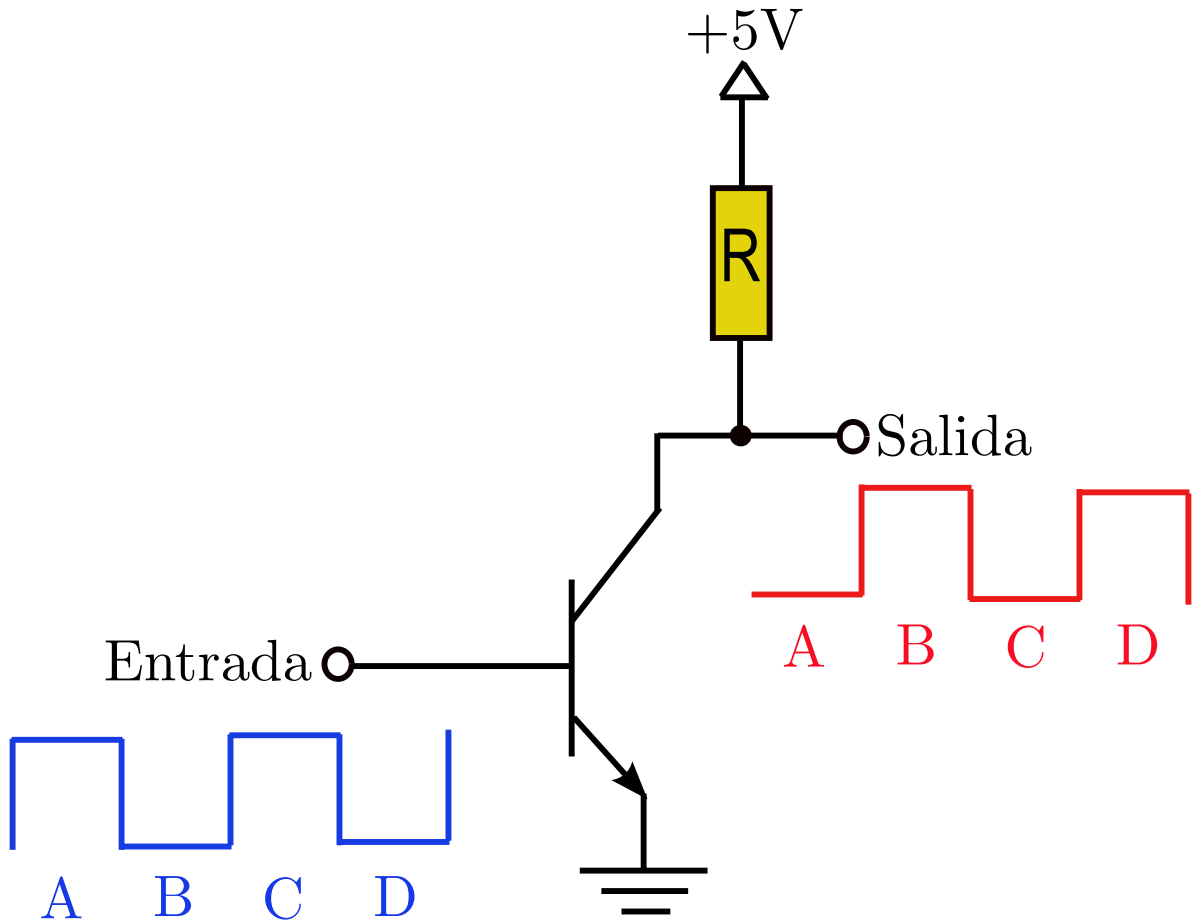


Figura 61: Salida a colector abierto.

I2C: Conexión tipo AND del bus

En un bus con conexión tipo AND, sólo si todos los transistores están apagados, la línea se mantiene en alto.

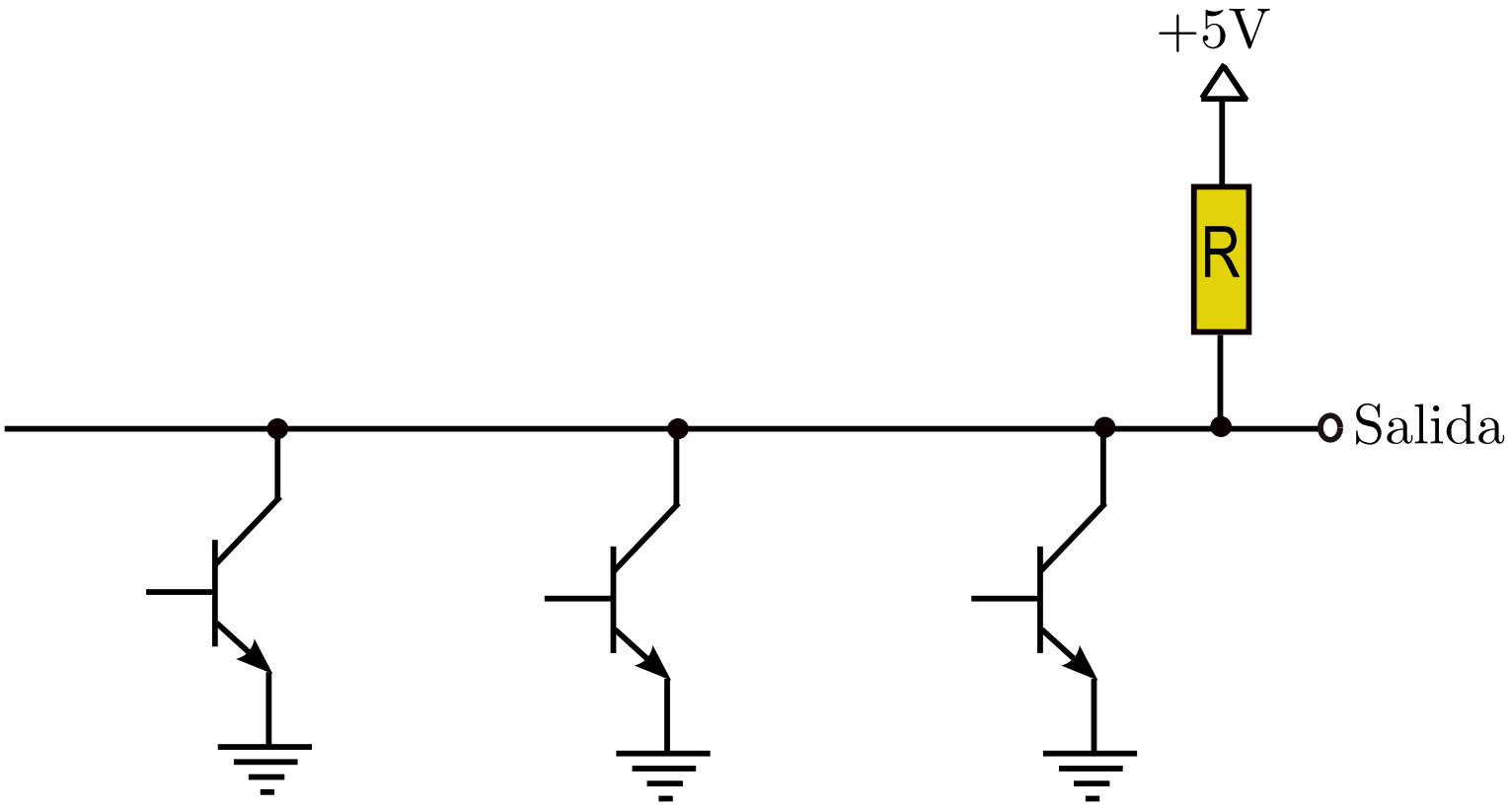


Figura 62: Bus conexión tipo AND.

Un dispositivo *maestro* es el que inicia la transferencia de datos y genera la señal de reloj.

Cualquier dispositivo direccionado por un *maestro* es considerado *esclavo*.

1er caso: A, maestro, envía información a B, esclavo:

a) A direcciona a B.
b) A envía el dato a B.
c) A termina la transferencia.

2do caso: A, maestro, recibe información desde B, esclavo:

a) A direciona a B.
b) A recibe el dato desde B.
c) A termina la transferencia.

Figura 63: Maestro A escribe y lee datos del esclavo B.

I2C: Transferencia y validación del bit

Cada pulso que se transfiere por la línea SDA debe ir acompañado de un pulso de reloj por la línea SCL.

El bit de datos por la línea SDA debe mantenerse estable durante el período alto ("1" lógico) de SCL.

La línea SDA sólo puede cambiar de estado durante el período bajo de SCL.

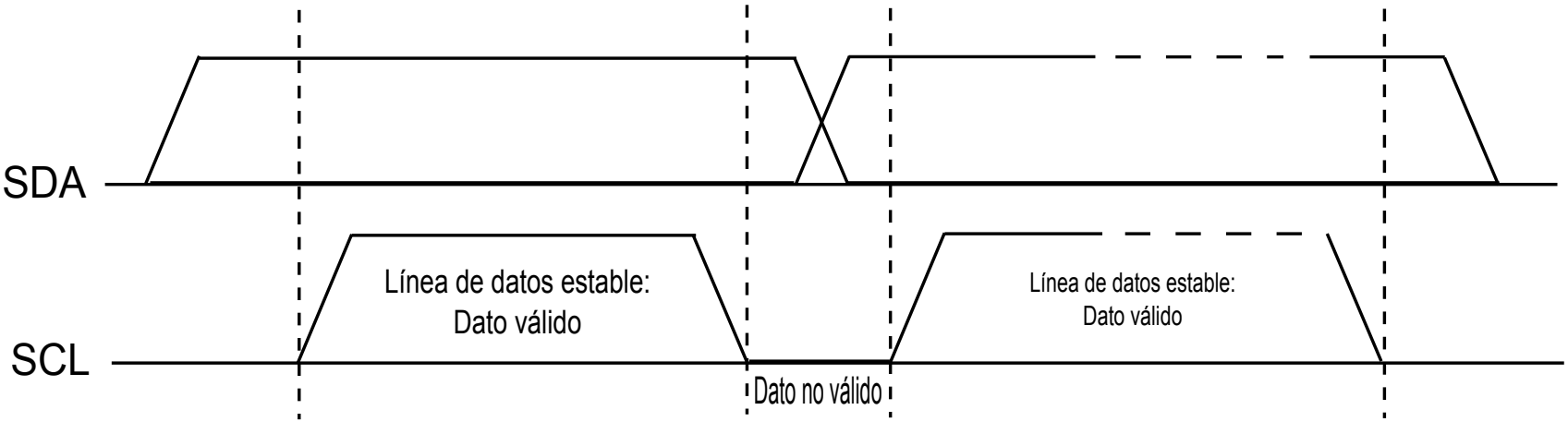


Figura 64: Validación del bit de datos.

I2C: Condiciones de inicio y de parada

La condiciones de inicio y de parada son eventos que determinan el inicio y final de una transferencia de datos entre el maestro y el esclavo.

Condición de inicio: Flanco de bajada en SDA mientras SCL está a nivel lógico “1”.

Condición de parada: Flanco de subida en SDA mientras SCL permanece a nivel “1”.

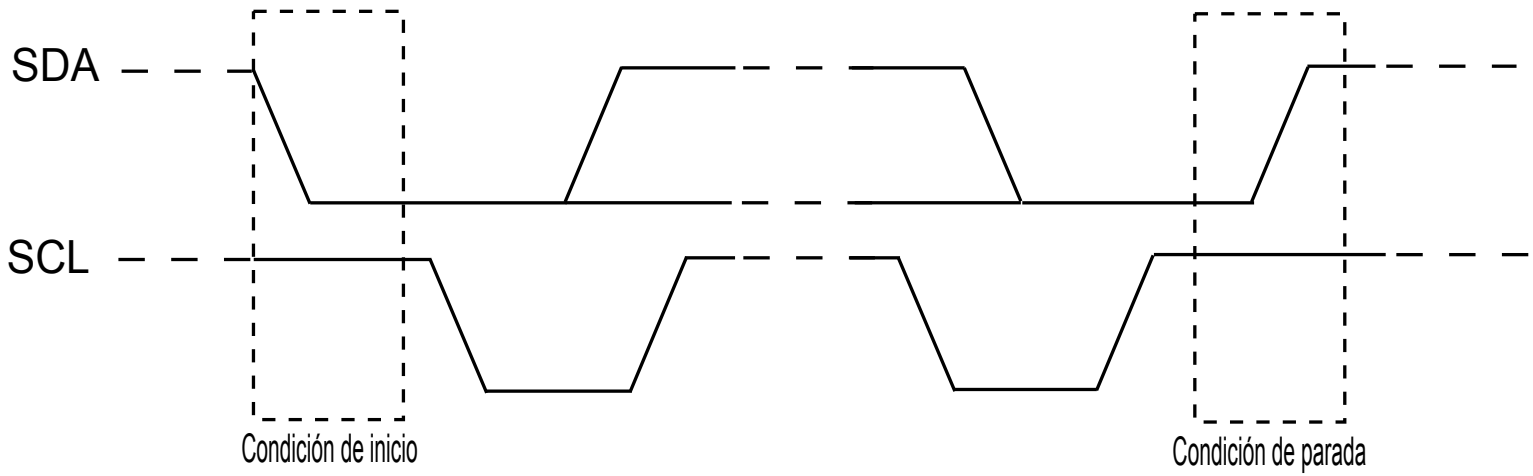


Figura 65: Condiciones de inicio y de parada.

I2C: Condiciones de inicio y de parada

Las condiciones de inicio y de parada son siempre generadas por el maestro.

El bus se considera ocupado (BUSY) después de la condición de inicio, y se considera libre, cierto tiempo después de la condición de parada.

I2C: Transferencia de datos

Todos los bytes colocados sobre SDA deben constar de 8 bits, comenzando por el bit de más peso (MSB).

Cada byte debe ir seguido de un bit de reconocimiento, ACK.

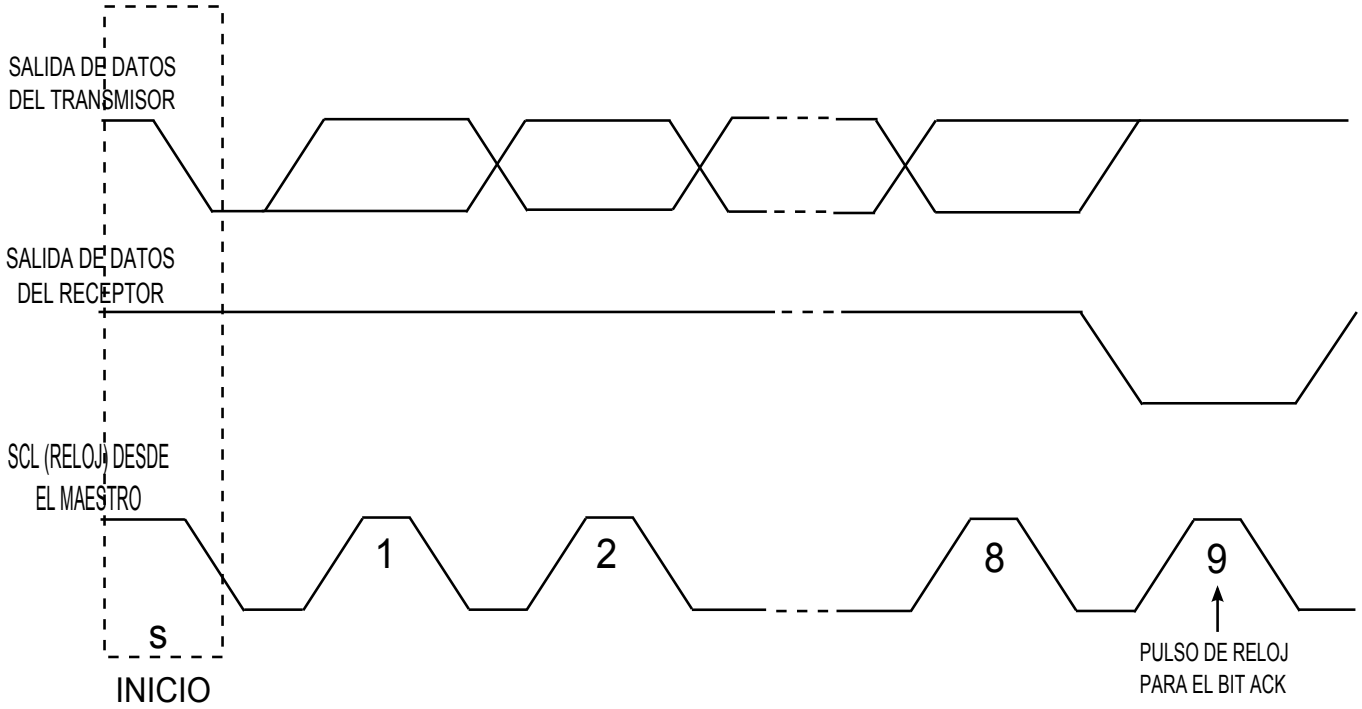


Figura 66: Formato del byte.

I2C: Bit de reconocimiento

El bit ACK es obligatorio en la transferencia de cada byte.

El pulso de reloj asociado al bit ACK lo genera el maestro.

El transmisor pone la línea SDA a “1” durante dicho pulso de reloj.

El receptor pone a “0” la línea SDA durante el pulso de reloj correspondiente al 9° bit.

Debido a la conexión tipo AND en el bus, en SDA prevalece el nivel “0”.

Si el esclavo no genera el bit de reconocimiento ACK al ser direccionado, por ejemplo si está ocupado realizando una función interna:

- el esclavo mantiene la línea SDA en “1” (NACK),
- esto lo detecta el maestro que genera una condición de parada,
- la transferencia es así abortada.

I2C: Formato

Un mensaje I2C estándar consiste de cutro partes:

- Condición de inicio;
- dirección de 7 bits del esclavo más bit de transmisión R/W#;
- trasferencia de datos; y
- condición de parada.

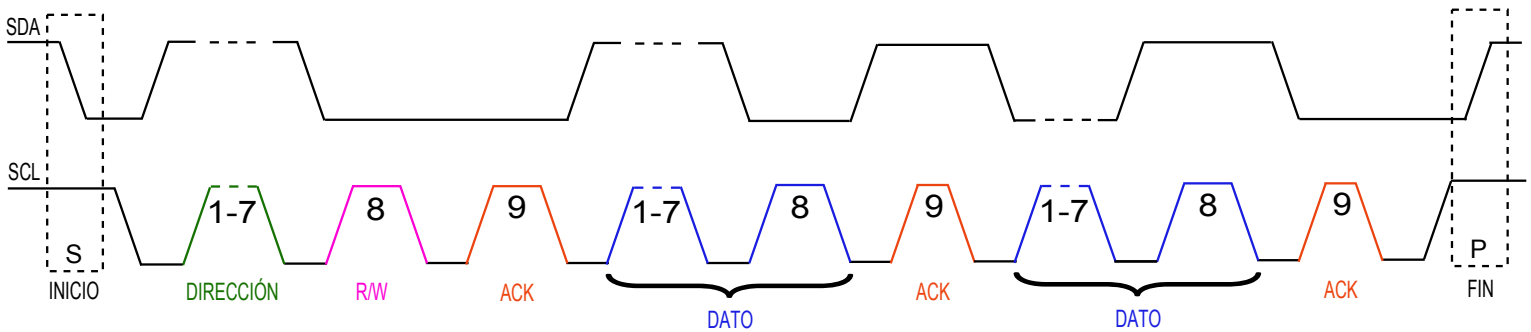


Figura 67: Formato de los datos transferidos.

I2C: Formato

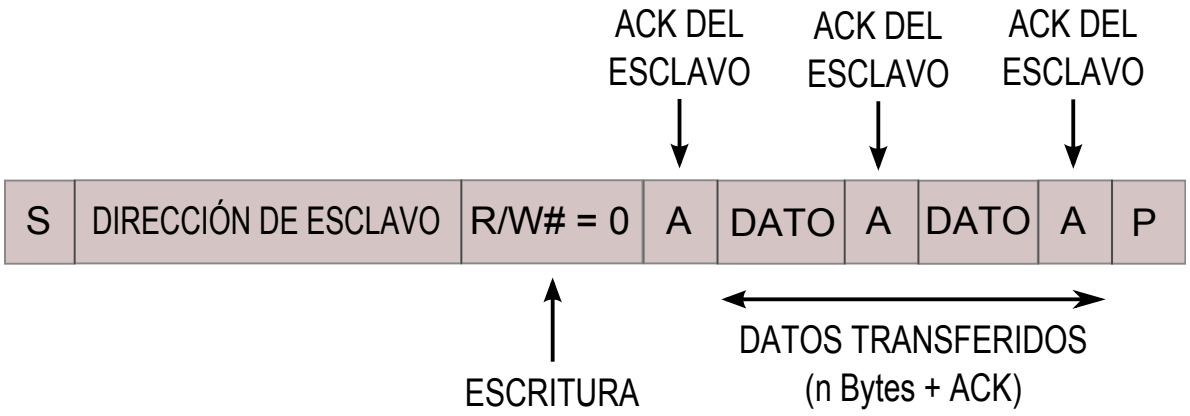


Figura 68: Ejemplo de escritura al esclavo.

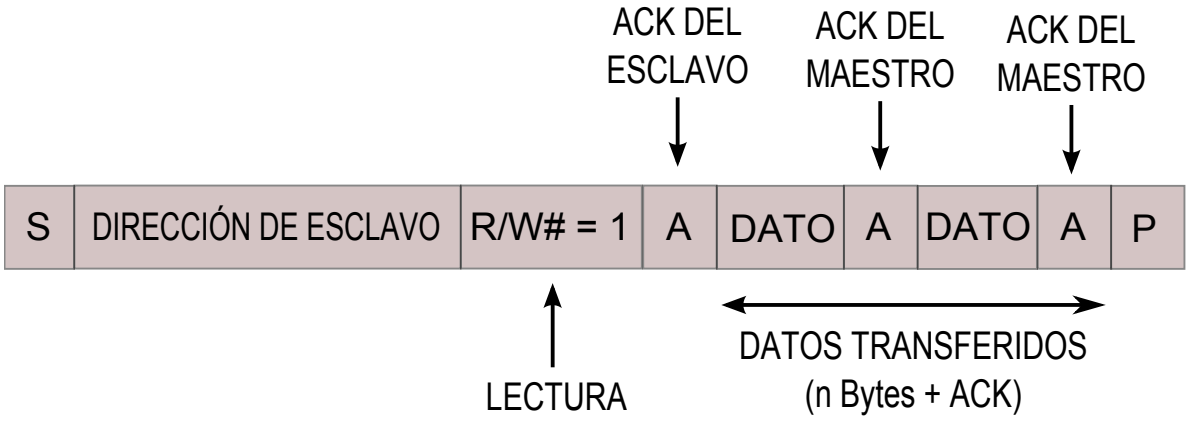


Figura 69: Ejemplo de lectura al esclavo.

I2C: Formato

Si un maestro aún desea comunicar por el bus, éste puede generar otra condición de inicio y direccionar a otro esclavo sin generar previamente la condición de parada.

Este formato también permite seleccionar un esclavo para leerlo, por ejemplo, y luego para escribirlo (dispositivo EEPROM).

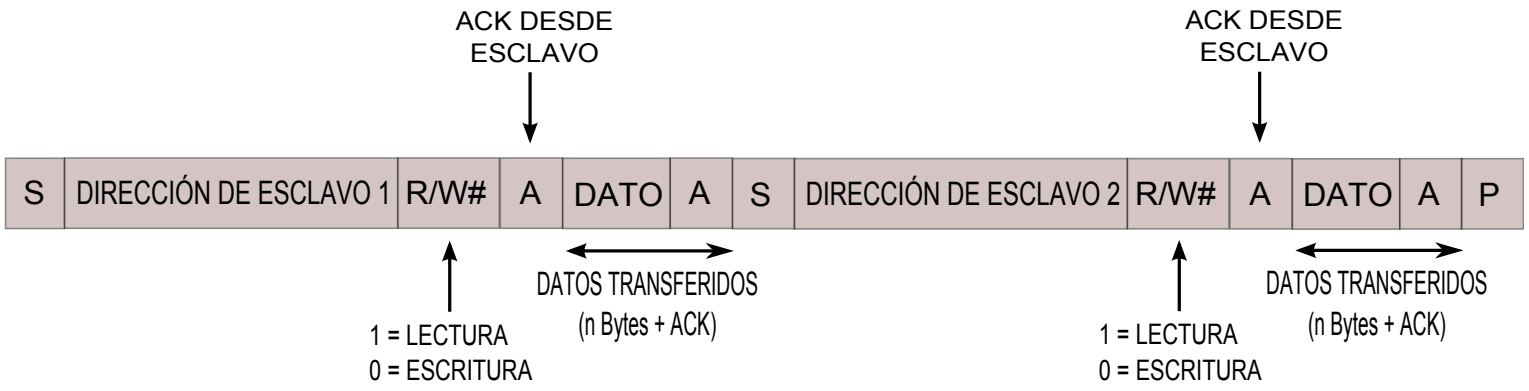
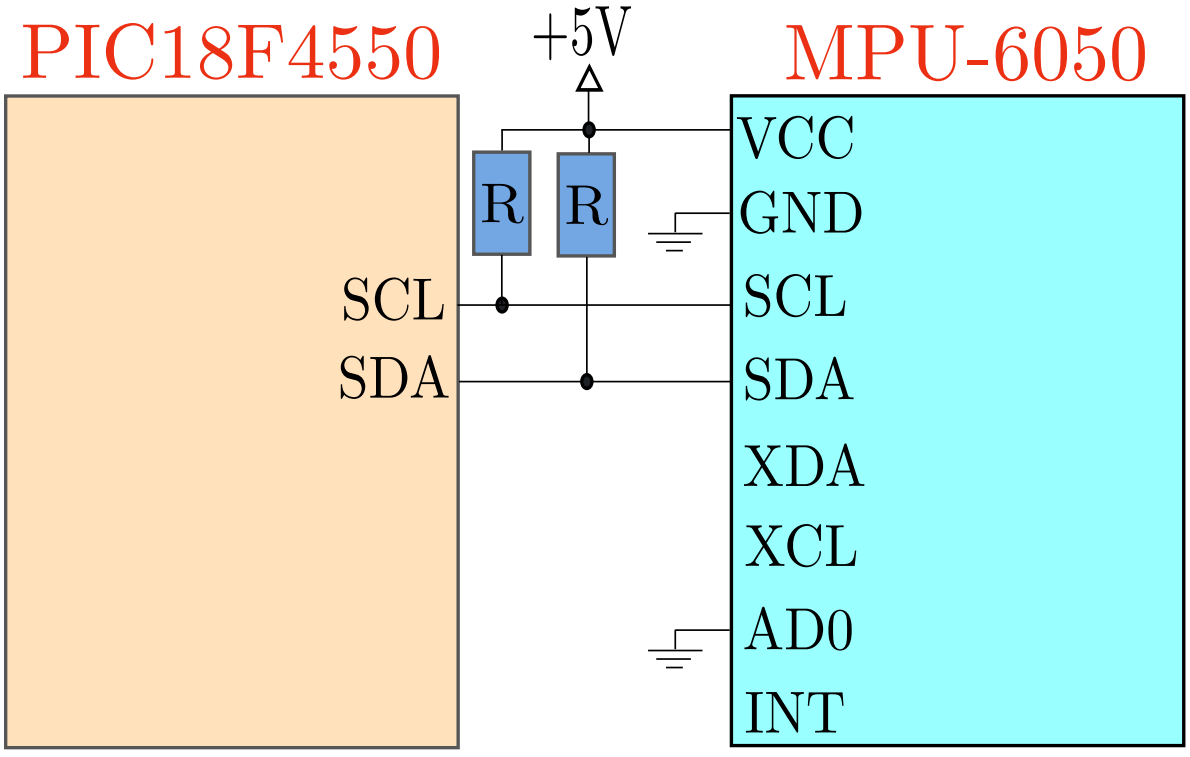


Figura 70: El maestro cambia el byte de dirección y selecciona un esclavo distinto.

I2C: MPU-6050



$\mathrm{R} = 4.7\ \mathrm{K}\Omega$

Figura 71: Diagrama de conexión entre el PIC y el Giroscopio - Acelerómetro MPU-6050.

I2C: MPU-6050

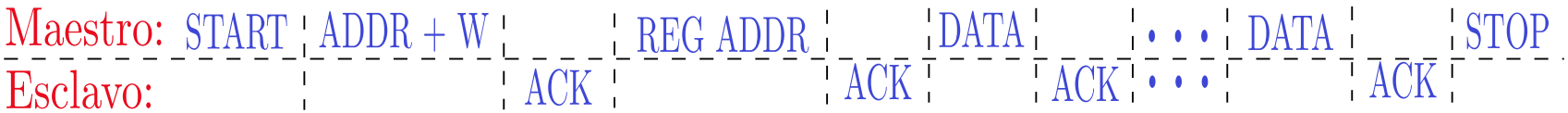


Figura 72: I2C: Secuencia de escritura típica.

```
// @Brief: Write a bunch of data to an I2C device.
// @Input:
//    - slave_address: I2C address of the device to be written.
//    - reg: Register where data will be written.
//    - data: Bunch of data that will be written.
//    - length: Number of bytes that will be written.
// @Output: none

void writeI2CDevice(unsigned char slave_address, unsigned char reg,
unsigned char* data, unsigned char length)
```

I2C: MPU-6050

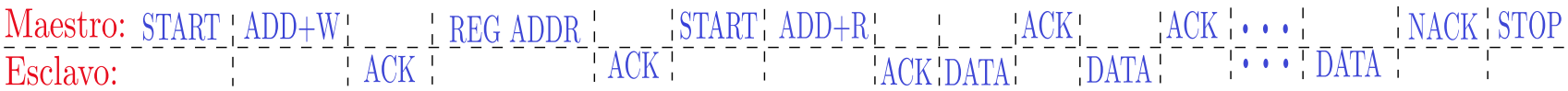


Figura 73: I2C: Secuencia de lectura típica.

```
// @Brief: Read one or more bytes out of an I2C slave device.
// @Input:
//    - slave_address: I2C address of the device to be written.
//    - data: Array where read data will be stored.
//    - length: Number of bytes that will be stored.
// @Output: none
```

```
void readI2CDevice(unsigned char slave_address, unsigned char reg,
unsigned char* data, unsigned char length)
```

I2C: MPU-6050

Comandos de la libería i2c.h de MPLABX:

```
IdleI2C();              // Espera hasta que el bus se desocupe.
StartI2C();             // Envia la condición de inicio.
WriteI2C( slave_address & 0xfe ); // Envía la dirección del esclavo con el
                                  // bit R/W en cero para escritura.
WriteI2C( slave_address | 0x01 ); // Envía la dirección del esclavo con el
                                  // bit R/W en uno para lectura.
WriteI2C(reg);          // Envía la dirección del registro a escribir o leer.
WriteI2C( data_byte );  // Envía el dato que se va a escribir.
data_byte = ReadI2C();  // Lee un byte de dato
AckI2C();               // Envía un ACK para continuar leyendo.
NotAckI2C();            // Envía un NACK para detener la lectura.
StopI2C();              // Envía la condición de parada.
```

I2C: MPU-6050

Abrir I2C:

```
SSPSTAT &= 0x3F; // power on state
SSPCON1 = 0x00;  // power on state
SSPCON2 = 0x00; // power on state
SSPCON1 |= 0b00001000; // select serial mode
SSPSTAT |= 0b10000000; // slew rate on/off

I2C_SCL_PIN = 1; // PORTBbits.RB1
I2C_SDA_PIN = 1; // PORTBbits.RB0
SSPCON1 |= SSPENB; // enable synchronous serial port

SSPADD = 0x13;
```

I2C: MPU-6050

writeI2CDevice

```
IdleI2C();              // Wait until the bus is idle
StartI2C();             // Send START condition
IdleI2C();              // Wait for the end of the START condition
WriteI2C( slave_address & 0xfe );       // Send address with R/W#
                                        // cleared for write
IdleI2C();                  // Wait for ACK
WriteI2C( reg );            // Send address of register where data will
                            // be written.
IdleI2C();                  // Wait for ACK
for(i=0; i<length; i++) {
    WriteI2C(data[i]);          // Write first byte of data
    IdleI2C(); }                // Wait for ACK
StopI2C();              // Hang up, send STOP condition
```

I2C: MPU-6050

readI2CDevice

```
IdleI2C();              // Wait until the bus is idle
StartI2C();             // Send START condition
IdleI2C();              // Wait for the end of the START condition
WriteI2C(slave_address&0xfe);   // Send address with R/W# cleared for write
IdleI2C();              // Wait for ACK
WriteI2C(reg);          // Send address of register to be read.
IdleI2C();              // Wait for ACK

StartI2C();             // Send START condition
IdleI2C();              // Wait for the end of the START condition
WriteI2C(slave_address|0x01);    // Send address with R/W set for read
IdleI2C();              // Wait for ACK
```

I2C: MPU-6050

```
for(i=0; i<length; i++)
{
    data[i] = ReadI2C();        // Read byte of data

    if(i != (length-1))
    {
        AckI2C();               // Send ACK if we are going to continue reading.
    }
    else
    {
        NotAckI2C();            // Send NACK if we won't continue reading.
    }
}

StopI2C();                      // Hang up, send STOP condition
```

I2C: MPU-6050

IdleI2C

```
while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F)); //Transmisión en progreso.
```

I2C: MPU-6050

StartI2C

```
IdleI2C();
SEN = 1; // Lleva a cabo la condición de inicio.
```

I2C: MPU-6050

StopI2C

```
IdleI2C();
PEN = 1; // Lleva a cabo la condición de parada.
```

I2C: MPU-6050

WriteI2C

```
IdleI2C();
SSPBUF = d; //Write data to SSPBUF
```

I2C: MPU-6050

ReadI2C

```
IdleI2C();
RCEN = 1; // Enable Receive mode for I2C.
IdleI2C();
return SSPBUF; //Read data from SSPBUF
```

I2C: MPU-6050

AckI2C

```
SSPCON2bits.ACKDT=0; // Will acknowledge.
SSPCON2bits.ACKEN=1; // Send the acknowledge bit.
```

I2C: MPU-6050

NotAckI2C

```
SSPCON2bits.ACKDT=1; // Will not acknowledge.
SSPCON2bits.ACKEN=1; // Send the not acknowledge bit.
```