

Laboratorio USB-PIC'School

Manual de usuario con tutorial y ejemplos para PIC16F88X

Rev. Enero 2018



MK Electrónica

C/ Ipergorta, 11 - 3ºA
48410 OROZKO - BIZKAIA
Te.: 34-944 04 80 89

email: info@mkelectronica.com
www.mkelectronica.com

INDICE GENERAL

TEMA 1: EL HARDWARE

1.1	INTRODUCCION	1-1
1.2	CARACTERISTICAS	1-2
1.3	ARQUITECTURA DE USB-PIC'SCHOOL	1-4
1.3.1	La fuente de alimentación	1-5
1.3.2	El oscilador	1-6
1.3.3	Los microcontroladores	1-6
1.3.4	Entradas analógicas	1-9
1.3.5	Entradas digitales	1-9
1.3.6	El generador lógico	1-10
1.3.7	El teclado	1-11
1.3.8	Salidas digitales	1-12
1.3.9	Salidas digitales de alta corriente	1-13
1.3.10	Interfase serie RS-232	1-14
1.3.11	Display de 7 segmentos	1-15
1.3.12	Pantalla LCD	1-16
1.3.13	Dispositivos I2C y 1-wire	1-17
1.3.14	Sección de Depuración/Grabación	1-18

TEMA 2: EL DEPURADOR/GRABADOR

2.1	INTRODUCCION	2-1
2.2	DESCRIPCION	2-1
2.3	ENTORNO DE DESARROLLO MPLAB-IDE EMPLEADO PARA LA GRABACION	2-2
2.3.1	Instalación	2-3
2.3.2	Selección de dispositivo	2-4
2.3.3	Seleccionando el grabador	2-5
2.3.4	Ficheros .HEX y buffers de memoria	2-5
2.3.5	Funciones más comunes del grabador	2-6
2.3.6	Iconos de grabación	2-8
2.4	EL SOFTWARE DE GRABACION PICKIT-2	2-9
2.4.1	Instalación	2-9
2.4.2	Selección de un dispositivo	2-10
2.4.3	Importar un fichero .HEX	2-11
2.4.4	Grabación del dispositivo	2-12
2.4.5	Verificando el dispositivo	2-12
2.4.6	Lectura de un dispositivo	2-13
2.4.7	Borrado del dispositivo	2-13
2.4.8	Verificación del borrado	2-13
2.4.9	Procedimientos automáticos de lectura/escritura	2-13
2.5	EL MODO DE DEPURACION	2-14
2.5.1	Seleccionando el dispositivo	2-15
2.5.2	Seleccionando el depurador	2-15
2.5.3	Funciones más comunes en la depuración	2-16
2.5.4	Iconos empleados en la depuración	2-23

TEMA 3: TUTORIAL Y EJEMPLOS

3.1	INTRODUCCION	3-1
3.2	TUTORIAL	3-3
3.2.1	Creación de un proyecto	3-3
3.2.2	El programa fuente	3-8
3.2.3	Preparando el hardware de la aplicación	3-9
3.2.5	Grabación del dispositivo	3-10
3.2.6	Comprobando el funcionamiento de la aplicación	3-11
3.2.7	Depuración del ejemplo anterior	
3.3	EJEMPLOS	3-13
3.3.1	Ejemplo 1: Simple manejo de las entradas y salidas	3-14
3.3.2	Ejemplo 2: Función lógica NOT	3-14
3.3.3	Ejemplo 3: Decodificador combinacional	3-14
3.3.4	Ejemplo 4: Automatismo secuencial	3-15
3.3.5	Ejemplo 5: Automatismo secuencial	3-16
3.3.6	Ejemplo 6: Automatismo, máquina de envasar	3-17
3.3.7	Ejemplo 7: Automatismo, máquina de envasar (versión mejorada)	3-17
3.3.8	Ejemplo 8: Juego de luces	3-18
3.3.9	Ejemplo 9: La interrupción del TMR0	3-19
3.3.10	Ejemplo 10: La interrupción externa RB0/INT y el modo SLEEP	3-20
3.3.11	Ejemplo 11: Interrupciones por cambio de estado en RB2:RB0	3-21
3.3.12	Ejemplo 12: El Watch Dog Timer (WDT)	3-21
3.3.13	Ejemplo 13: El display de 7 segmentos	3-22
3.3.14	Ejemplo 14: Decodificador BCD- 7 segmentos	3-23
3.3.15	Ejemplo 15: Barrido de displays	3-23
3.3.16	Ejemplo 16: Contador decimal UP/DOWN	3-24
3.3.17	Ejemplo 17: Seguimos El TMR0 como contador de eventos externos	3-25
3.3.18	Ejemplo 18: La memoria EEPROM de datos, la máquina "Su Turno"	3-26
3.3.19	Ejemplo 19: Introducción al manejo de la pantalla LCD	3-27
3.3.20	Ejemplo 20: Seguimos con la pantalla LCD	3-28
3.3.21	Ejemplo 21: Visualizando más mensajes	3-28
3.3.22	Ejemplo 22: El teclado	3-29
3.3.23	Ejemplo 23: El teclado y la pantalla LCD	3-30
3.3.24	Ejemplo 24: Interrupción por cambio de estado	3-31
3.3.25	Ejemplo 25: Teclado alfanumérico	3-31
3.3.26	Ejemplo 26: El TMR1 en modo temporizador, interrupción cada cierto tiempo	3-32
3.3.27	Ejemplo 27: El TMR1 como contador de eventos externos	3-33
3.3.28	Ejemplo 28: El TMR1 como contador de eventos externos con control de disparo	3-33
3.3.29	Ejemplo 29: El TMR1 como divisor de frecuencia	3-34
3.3.30	Ejemplo 30: Frecuencímetro	3-35
3.3.31	Ejemplo 31: El TMR2, interrupción periódica	3-35
3.3.32	Ejemplo 32: El TMR2, temporizaciones ajustables	3-36
3.3.33	Ejemplo 33: El oscilador interno	3-37
3.3.34	Ejemplo 34: El módulo CVREF, generar una tensión analógica variable	3-38
3.3.35	Ejemplo 35: El módulo CVREF, generando una señal en diente de sierra	3-39
3.3.36	Ejemplo 36: El módulo comparador, comparando dos tensiones	3-41
3.3.37	Ejemplo 37: El módulo comparador, comparando una tensión analógica con otra fija de referencia	3-41
3.3.38	Ejemplo 38: El módulo comparador, comparador con tensión de referencia ajustable	3-42
3.3.39	Ejemplo 39: El módulo convertidor ADC	3-44
3.3.40	Ejemplo 40: El módulo convertidor ADC, voltímetro digital	3-45
3.3.41	Ejemplo 41: El módulo convertidor ADC, multiplexando entradas analógicas	3-45
3.3.42	Ejemplo 42: Capturas con el módulo CCP, midiendo periodos	3-47
3.3.43	Ejemplo 43: Capturas con el módulo CCP, midiendo la anchura de un pulso	3-47

3.3.44	Ejemplo 44: Capturas con el módulo CCP, midiendo tiempo entre un pulsos	3-48
3.3.45	Ejemplo 45: Modo comparación del CCP1, contador de eventos	3-48
3.3.46	Ejemplo 46: Comparación con el módulo CCP1, contador de eventos ajustable	3-49
3.3.47	Ejemplo 47: Modulación de anchura de pulsos (PWM) mediante el módulo CCP1	3-50
3.3.48	Ejemplo 48: PWM mediante el módulo CCP1, variando la anchura	3-50
3.3.49	Ejemplo 49: Modulación PWM mediante el convertidor ADC	3-51
3.3.50	Ejemplo 50: Modulación PWM, modo "Pulse Steering"	3-52
3.3.51	Ejemplo 51: Modulación PWM, modo "Half-Bridge"	3-54
3.3.52	Ejemplo 52: Modulación PWM, modo "Full-Bridge"	3-55
3.3.53	Ejemplo 53: Modulación PWM, modo "Full-Bridge" y "Auto-Shutdown"	3-57
3.3.54	Ejemplo 54: El USART, transmitiendo caracteres	3-58
3.3.55	Ejemplo 55: El USART, recepción/transmisión de caracteres	3-58
3.3.56	Ejemplo 56: Auto Detección del USART	3-59
3.3.57	Ejemplo 57: El USART, auto detección y transmisión de mensajes	3-60
3.3.58	Ejemplo 58: El USART, monitorización remota	3-60
3.3.59	Ejemplo 59: El USART, Control remoto	3-61
3.3.60	Ejemplo 60: Protocolo y comunicación 1-wire	3-62
3.3.61	Ejemplo 61: Comunicación 1-wire, medir la temperatura con el DS18S20	3-63
3.3.62	Ejemplo 62: Comunicación 1-wire, termostato	3-64
3.3.63	Ejemplo 63: Protocolo y comunicación I2C, un secuenciador en RAM	3-65
3.3.64	Ejemplo 64: Comunicación I2C, el reloj del DS1307	3-69
3.3.65	Ejemplo 65: Comunicación I2C, Reloj/Calendario con el DS1307	3-69
3.3.66	Ejemplo 66: Aplicación práctica, cartel publicitario	3-71
3.3.67	Ejemplo 67: Aplicación práctica, Data Logger	3-71
3.3.68	Ejemplo 68: Aplicación práctica, juego "Adivina el número"	3-72
3.3.69	Ejemplo 69: Aplicación práctica, control de accesos	3-73
3.3.70	Ejemplo 70: Aplicación práctica, Programa de test de USB-PIC'School	3-74

BIBLIOGRAFIA

- 1.- Microcontroladores PIC, Teoría y Práctica. Ed. Creaciones Copyright
- 2.- Microcontroladores PIC, diseño práctico de aplicaciones. Primera parte. Ed. Mac Graw-Hill
- 3.- Microcontroladores PIC, diseño práctico de aplicaciones. Segunda parte. Ed. Mc Graw-Hill
- 4.- Diseño práctico con microcontroladores. Ed. Thomoson
- 5.- MPASM User's guide. Microchip
- 6.- MPLAB-IDE User's guide. Microchip
- 7.- PIC16F88X Data Sheet. Microchip
- 8.- PICKit-2 User Guide. Microchip

DIRECCIONES DE INTERES

MK Electrónica: www.mkelectronica.com
Microchip. www.microchip.com
Custom Computer Services inc. (compiladores C) www.ccsinfo.com
Topway (pantallas LCD) www.topwaydisplay.com y www.topwaysz.com
Dallas Semiconductor (dispositivos 1-wire e I2C) www.maxim-ic.com



TEMA 1: EL HARDWARE

1.1 INTRODUCCION

MK Electrónica. tiene el placer de *poner* en tus manos la nueva versión de nuestro popular laboratorio PIC'School, la herramienta ideal para el aprendizaje y diseño de aplicaciones basadas en los populares microcontroladores PIC de Arizona Microchip. Figura 1-1.

USB-PIC'School es un laboratorio de carácter didáctico. Con él podremos trabajar con la mayor parte de los dispositivos PIC más representativos de las familias 12F, 16F y 18F, en 8, 18, 28 y 40 patillas, pudiéndose desarrollar aplicaciones tanto a nivel de software como de hardware. Dispone de un buen número de los periféricos más empleados en las aplicaciones reales, así como un módulo board para el montaje sin soldadura y recuperación de componentes que permite implementar el hardware necesario en un determinado proyecto.

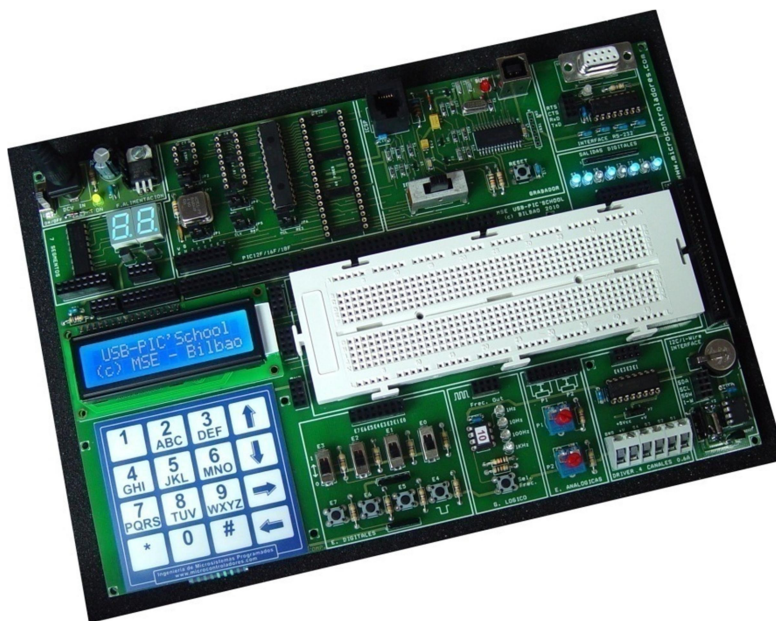


Figura 1-1. El laboratorio USB-PIC'School

A diferencia de otros equipos similares, ninguno de los periféricos contenidos en el laboratorio **USB-PIC'School** están conectados de forma predeterminada a las líneas de E/S del microcontrolador. Es el propio usuario quien, mediante un sencillo cableado sin soldadura, conecta y asocia los distintos periféricos a esas líneas de E/S según lo requiera la aplicación. Esto permite tener un contacto más directo con el hardware y un mayor control del mismo, imprescindible en la fase de aprendizaje de los microcontroladores y/o en el desarrollo de aplicaciones basadas en los mismos.

En *MK Electrónica.* somos conscientes de la necesidad de disponer de herramientas sencillas, flexibles, potentes y de bajo coste, que permitan desarrollar, depurar, grabar y evaluar de forma cómoda y rápida las aplicaciones desarrolladas por el usuario, y donde se puedan emplear y aprovechar al máximo los distintos recursos internos que poseen todos los microcontroladores PIC, desde la gama baja a la gama mejorada. Esta ha sido nuestra meta y objetivo final a la hora de presentar el laboratorio **USB-PIC'School**.

1.2 CARACTERISTICAS

A continuación se muestran las características más relevantes del laboratorio USB-PIC'School:

- Alimentación única mediante F. Alimentación externa (no incluida) de 9 a 15VDC con positivo al centro. El laboratorio incorpora los circuitos de filtrado, estabilización, piloto e interruptor ON/OFF.
- Admite dispositivos PIC de 8, 18, 28 y 40 patillas de las familias 12FXXX, 16FXXX y 18FXXX alimentados con +5V y compatibles con la herramienta PICKit-2. De serie se suministra con el moderno PIC16F886 de 28 pines (versión muy mejorada del PIC16F876).

- Oscilador integrado a cristal de cuarzo encapsulado en DIP8 para generar la frecuencia general de trabajo del sistema. El oscilador suministrado de serie es de 4MHz, se inserta sobre un zócalo de 8 pines y es fácilmente intercambiable por otros osciladores con otras frecuencias.
- Configuración mediante jumpers de las señales CLK/GP5, CLK/RA7, MCL/GP3, MCL/RA5 y MCL/RE3. Igualmente se puede configurar el tipo de reloj a emplear: Oscilador a cristal de cuarzo o red R-C (XT/RC). Es posible por tanto adaptarse a múltiples configuraciones según modelo de PIC y/o aplicación a realizar.
- Integra el hardware necesario para la depuración y/o grabación del dispositivo. Dicho hardware es compatible con el PICKIT2 de Microchip y, por tanto, se puede controlar mediante sus herramientas software originales (MPLAB y PICKIT).
- Tres modos de operación diferentes que se seleccionan fácilmente mediante un conmutador deslizante. El modo RUN se emplea cuando el dispositivo está grabado con el programa de aplicación a ejecutar. Todas las líneas de E/S están disponibles para el control de periféricos. El modo USB se emplea durante los ciclos de **Depuración/Grabación** del dispositivo. Esta Depuración/Grabación se realiza mediante **un puerto USB de un PC** dotado del conocido entorno de desarrollo **MPLAB de Microchip**. Se suministra el cable USB de tipo A a tipo B para la conexión con el PC. Finalmente el modo ICSP permite conectar el laboratorio USB-PIC'School con otras herramientas externas dotadas de las señales ICSP necesarias para la depuración y/o grabación en circuito, como pueden ser el ICD2 o el ICD3 de Microchip.
- Todos los puertos con las líneas de E/S del PIC son accesibles mediante bloques de conectores que permiten una rápida y sencilla conexión con los periféricos a controlar.
- No hay conexiones predeterminadas. Cualquier periférico se puede conectar a cualquier línea de E/S.
- Acceso a los periféricos mediante bloques de conectores que permiten realizar una rápida y fiable conexión con las líneas de E/S del PIC. Basta emplear cable rígido de conexiones de 0.6 mm de grosor.
- Interface serie estándar RS232 con conector DB9 hembra y acceso para la gestión y control de las señales TxD, RxD, CTS y RTS.
- Reloj/calendario en tiempo real (RTC) alimentado mediante pila (incluida) y sensor de temperatura. Ambos dispositivos se gestionan mediante el bus I2C y 1-Wire, lo que permite el aprendizaje y empleo de estos protocolos tan empleados en las aplicaciones de carácter comercial.
- 8 salidas digitales pilotadas mediante leds de un elegante color azul de 3 mm y alto contraste.
- Dos displays de 7 segmentos de 0.5" y ánodo común, en color azul y alto contraste. Permiten conocer las técnicas de decodificación y visualización mediante "barrido secuencial"
- 4 salidas digitales de alta corriente mediante drivers de 4 canales a 0.6A cada uno. La tensión de salida de estos canales se selecciona mediante jumper y puede ser fija de +5Vcc o bien una tensión externa de referencia de hasta 36V. Tanto las salidas como la entrada de tensión externa son accesibles mediante bornas o clemas de paso 5.08 para la conexión de cargas de consumo elevado (relés, motores DC, motores PAP, etc.)
- Pantalla LCD de 2 x 16 caracteres en blanco sobre fondo azul, y conectores que permiten el interface a 4 u 8 bits.
- Teclado matricial de membrana de 4 x 4 teclas.
- 8 entradas digitales formadas por 4 interruptores y 4 pulsadores activos por transición 1-0-1.



- 2 entradas mediante potenciómetros que permiten evaluar y simular variables analógicas.
- Generador lógico de onda cuadrada con frecuencia de salida ajustable de 1Hz, 10Hz, 100Hz y 1KHz y pilotada mediante leds.
- Módulo board de 550 contactos para el montaje sin soldadura de todo tipo de circuitos de adaptación, periféricos, etc..
- Conector de expansión PIC-BUS 2 de 40 vías que transporta todas las señales del PIC y que permite la conexión de todo tipo de tarjetas y circuitos externos. Este conector es 100% compatible con el de otros productos MKE como PIC'Control.
- Se incluye un CDROM con una colección de más de 70 ejemplos de aplicación y programación, escritos tanto en ensamblador como en lenguaje C. Con estos ejemplos se trata de utilizar todos los recursos internos del potente dispositivo PIC16F886 así como todas las posibilidades y prestaciones del propio laboratorio USB-PIC'School. En el CDROM también se adjunta el presente manual y tutorial en castellano así como abundante información técnica.
- El laboratorio USB-PIC'School se presenta totalmente montado, comprobado y listo para funcionar, en un útil maletín de plástico para su transporte y/o almacenamiento. Se incluye el cable para la conexión USB con el PC, el microcontrolador de gama media PIC16F886 y el cable de conexiones para realizar los diferentes ejemplos propuestos.

1.3 ARQUITECTURA DE USB-PIC'School

En este apartado se hace una explicación exhaustiva del laboratorio USB-PIC'School cuyas secciones se muestran en el plano de serigrafía de la figura 1-2. Mediante los correspondientes esquemas el usuario podrá comprender la conexión de los diferentes periféricos disponibles así como las consideraciones que se deben tener sobre los mismos. Estas explicaciones pueden servir de ayuda al usuario para realizar sus propias aplicaciones a medida.



1.3.1 La fuente de alimentación

Se encarga de obtener la tensión general de +5Vcc con la que se alimenta el laboratorio USB-PIC'School. En la figura 1-3 se muestra su ubicación.

Figura 1-3. La fuente de alimentación



El esquema eléctrico se muestra en la figura 1-4. A través del conector CN1 se aplica la tensión de 9 a 15VDC procedente de un alimentador estándar con el positivo al centro. El interruptor SW7 conecta la alimentación del sistema. El diodo D1 evita la polarización inversa y con ello los daños que se pudieran ocasionar al equipo. El regulador U1 estabiliza la tensión a +5Vcc 1A máximo con la que trabaja el equipo. Dicha tensión, junto con GND están disponibles en el conector AP1 y AP2 respectivamente para alimentar a los circuitos y prototipos del usuario. En el conector AP3 está disponible también la tensión de entrada sin estabilizar +Vin para aquellos circuitos que la requieran. El diodo led D3 indica la presencia de tensión +5Vcc.

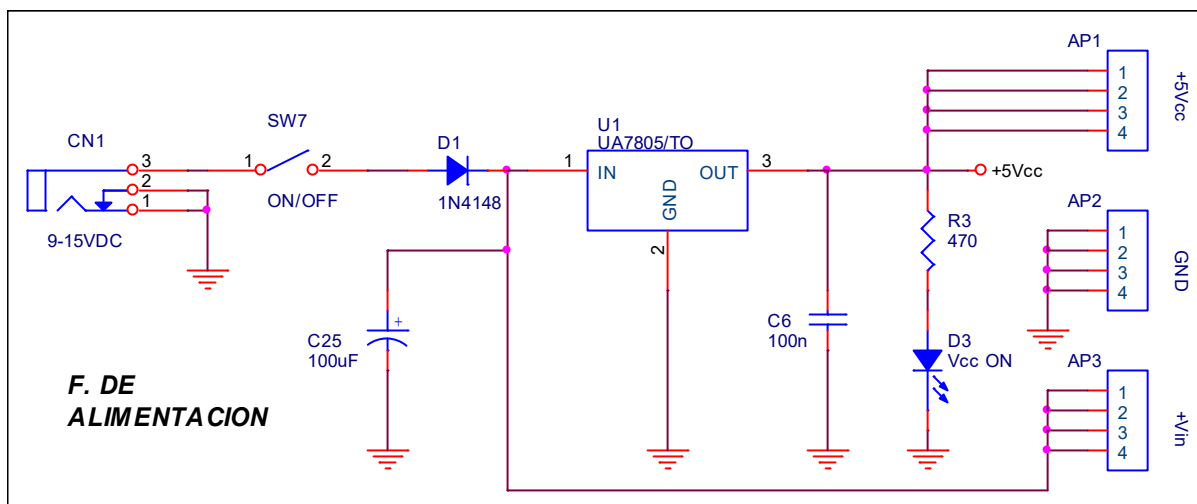
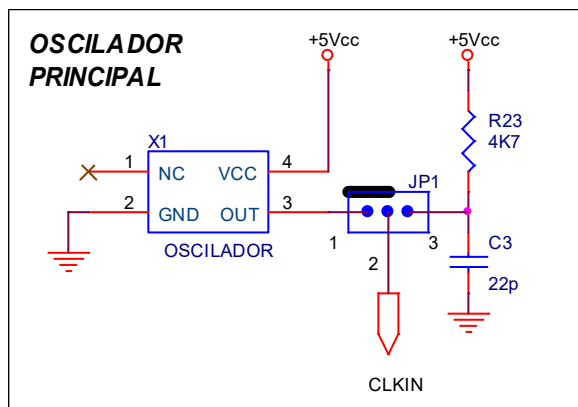


Figura 1-4. Esquema de la fuente de alimentación

1.3.2 El oscilador

Se encarga de generar la frecuencia principal de trabajo del microcontrolador. Su esquema se muestra en la figura 1-5. Está formado por el dispositivo integrado X1 y la red RC formada por R23 y C3. Mediante el jumper JP1 se obtiene la señal de reloj CLKIN. Cerrando 1-2 dicha señal la proporciona el oscilador X1. Cerrando 2-3 la señal se obtiene de la red RC. El valor de estos componentes determina la frecuencia de la misma (consultar parámetros proporcionados por Microchip)

Figura 1-5. Circuito oscilador





El dispositivo X1 contiene en su interior un oscilador preciso controlado por cuarzo y todo ello en una cápsula metálica en formato DIP8. Ver la figura 1-6. Este oscilador, también llamado *"Canned Oscillator"*, se puede adquirir en diferentes frecuencias según nuestras necesidades. Basta simplemente cambiar uno por otro para que nuestra aplicación se ejecute a más o menos velocidad.

Figura 1-6. *"Canned Oscillator"*

1.3.3 Los microcontroladores

Es la sección más importante del laboratorio USB-PIC'School y donde se inserta el dispositivo PIC con el que se va a trabajar. Básicamente consta de cuatro zócalos donde se pueden insertar los dispositivos de 8, 18, 28 y 40 patillas tal y como se muestra en la fotografía de la figura 1-7, USB-PIC'School incorpora de serie el potente controlador PIC16F866 de gama media. Se trata de un dispositivo muy mejorado respecto a su predecesor PIC16F876.

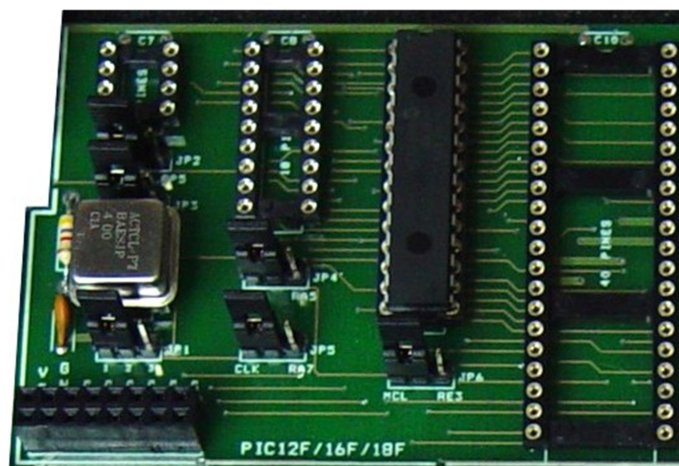


Figura 1-7. *Los microcontroladores*

En la figura 1-8 se muestran los diferentes tipos de encapsulados que se admiten. Se debe tener en cuenta que Microchip diseña y comercializa de forma constante nuevos modelos y por ello, la figura muestra la distribución de patillas y señales que deben tener las cápsulas de 8, 18, 28 y 40 patillas, para ser compatibles a nivel de hardware con el laboratorio USB-PIC'School. También se debe considerar que el software de grabación soporte un dispositivo determinado. Nosotros sugerimos el empleo de los programas de libre distribución como el MPLAB-IDE de Microchip, que se puede obtener y actualizar periódicamente con nuevas versiones en las que se soportan nuevos dispositivos PIC.

La figura 1-8 también muestra cómo algunas patillas pueden tener una doble función según el modelo de PIC que se trate. Normalmente dichas señales se pueden configurar en el propio PIC, durante el proceso de grabación y, mediante jumpers, se conectan al hardware según interese. Por ejemplo, en algunos dispositivos PIC de 18 pines, la patilla 16 se puede emplear como entrada de reloj (CLKIN) o como línea de E/S (RA7).

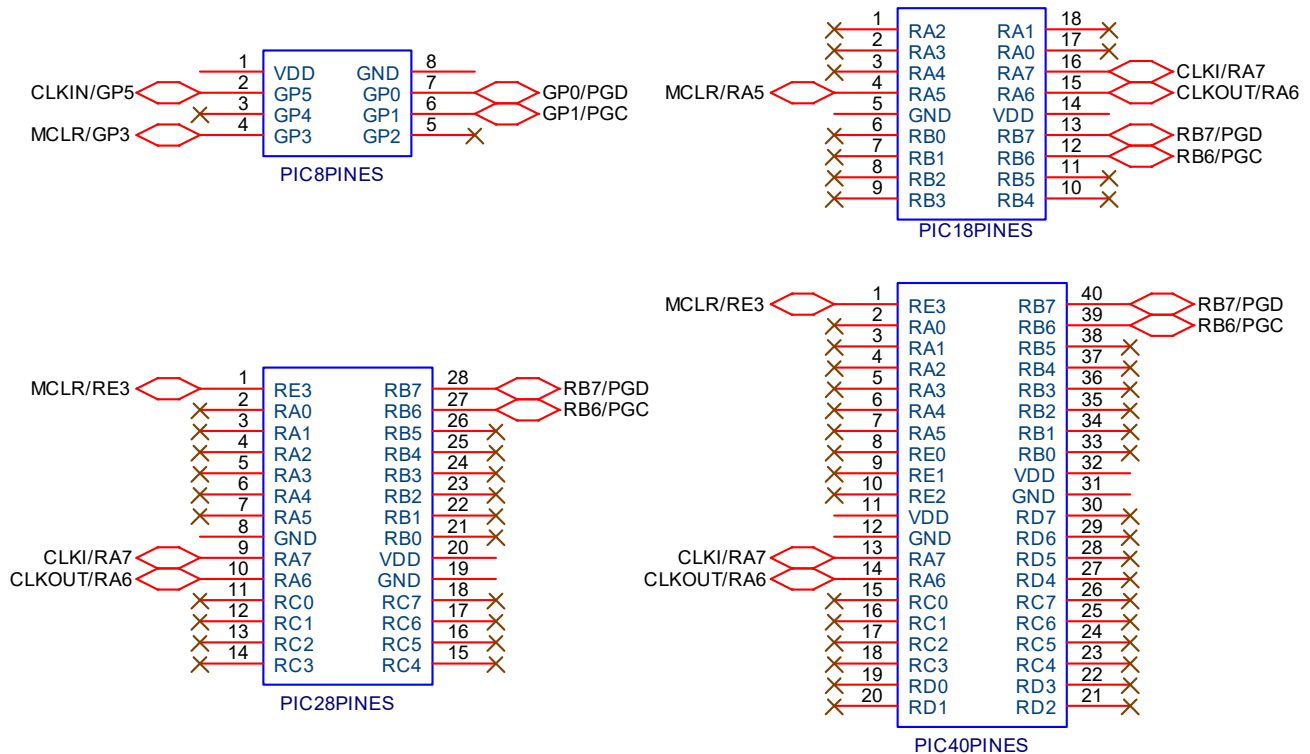
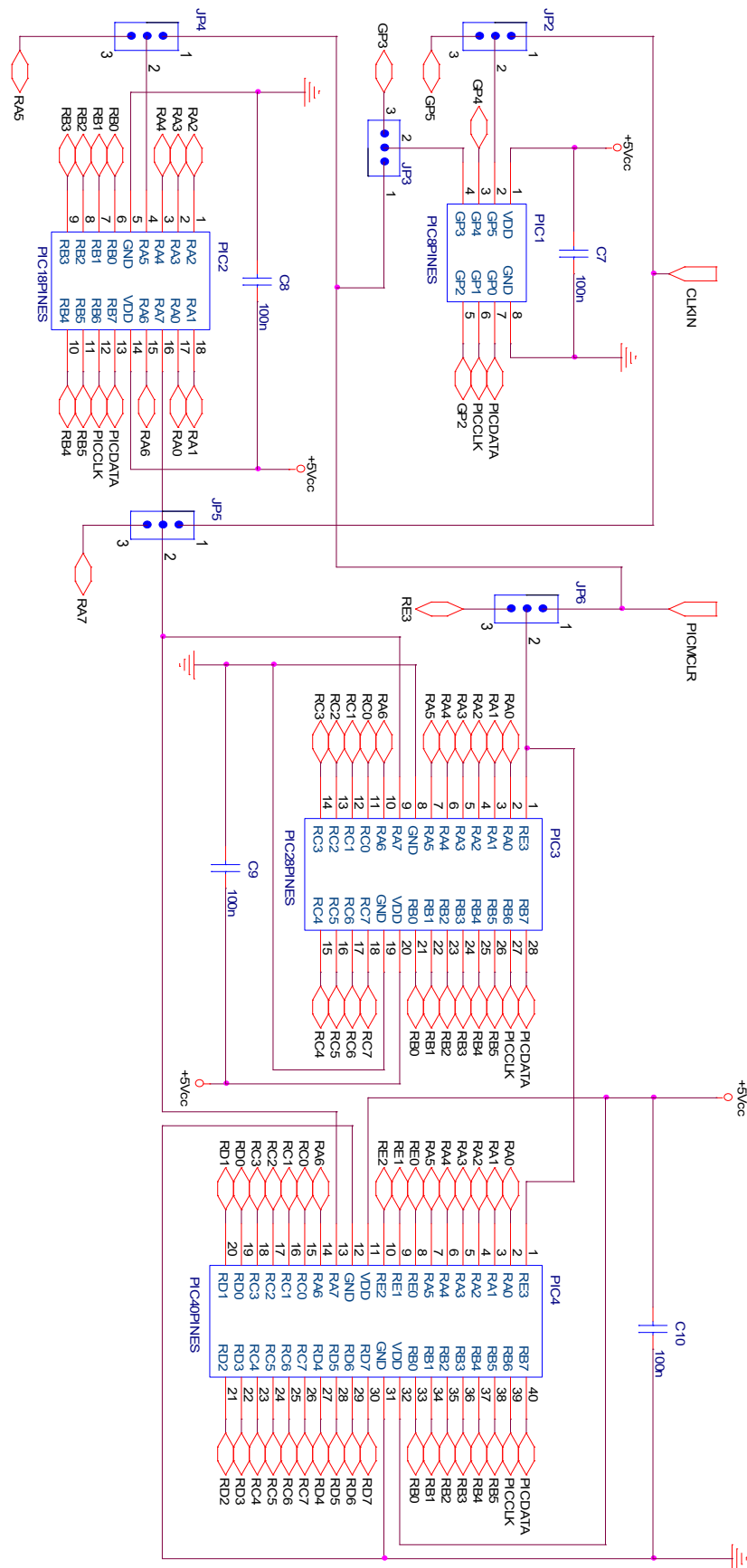


Figura 1-8. Distribución de señales según el tipo de encapsulado

El esquema eléctrico de la figura 1-9 muestra las distintas conexiones así como los diferentes jumpers de configuración disponibles en el laboratorio USB-PIC'School y que se resumen en la siguiente tabla:

CAPSULA	JUMPER	POSICION	DESCRIPCION
8 Pines	JP2	1-2	Por defecto CLKIN. Entrada de reloj externo que se obtiene desde el oscilador principal.
		2-3	GP5. Línea de E/S de propósito general si está disponible en el dispositivo.
	JP3	1-2	Por defecto PICMCLR. Entrada de reset externo. Imprescindible en los ciclos de lectura/grabación del dispositivo.
		2-3	GP3. Línea de E/S de propósito general si está disponible en el dispositivo.
18 Pines	JP4	1-2	Por defecto PICMCLR. Entrada de reset externo. Imprescindible en los ciclos de lectura/grabación
		2-3	RA5. Línea de E/S de propósito general si está disponible en el dispositivo.
	JP5	1-2	Por defecto CLKIN. Entrada de reloj externo que se obtiene desde el oscilador principal.
		2-3	RA7. Línea de E/S de propósito general si está disponible en el dispositivo.
28/40 Pines	JP5	1-2	Por defecto CLKIN. Entrada de reloj externo que se obtiene desde el oscilador principal.
		2-3	RA7. Línea de E/S de propósito general si está disponible en el dispositivo.
	JP6	1-2	Por defecto PICMCLR. Entrada de reset externo. Imprescindible en los ciclos de lectura/grabación del dispositivo.
		2-3	RE3. Línea de E/S de propósito general si está disponible en el dispositivo.



1.3.4 Entradas analógicas

Están formadas por dos potenciómetros tal y como se muestra en la fotografía de la figura 1-10 y en el esquema de la figura 1-11..

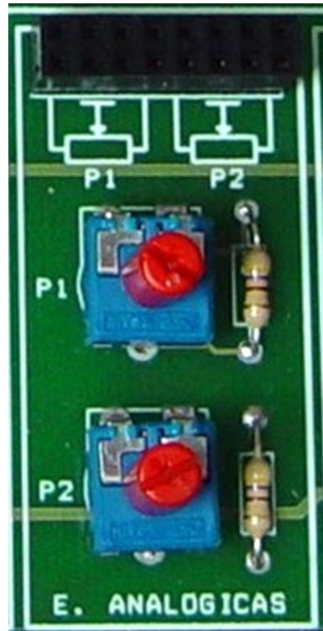


Figura 1-10. Las entradas analógicas

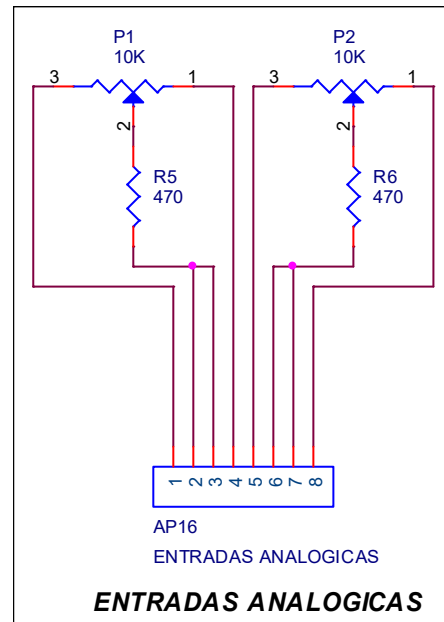


Figura 1-11. Conexiones de las entradas analógicas

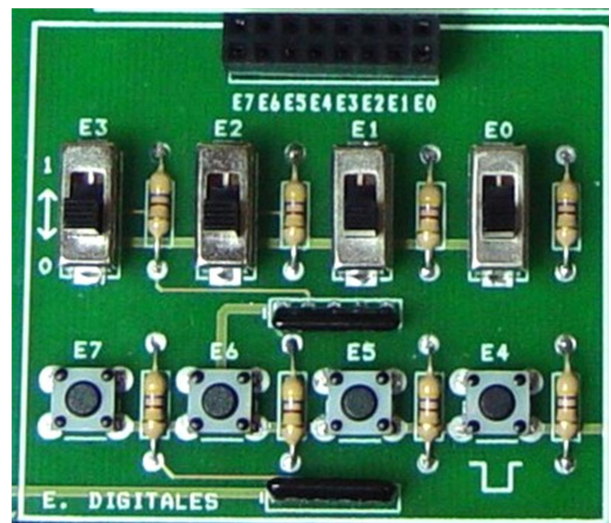
Según el esquema, las líneas de los potenciómetros son directamente accesibles desde el bloque de conexión AP16 correspondiente y se las puede conectar cualquier tipo de circuito eléctrico analógico.

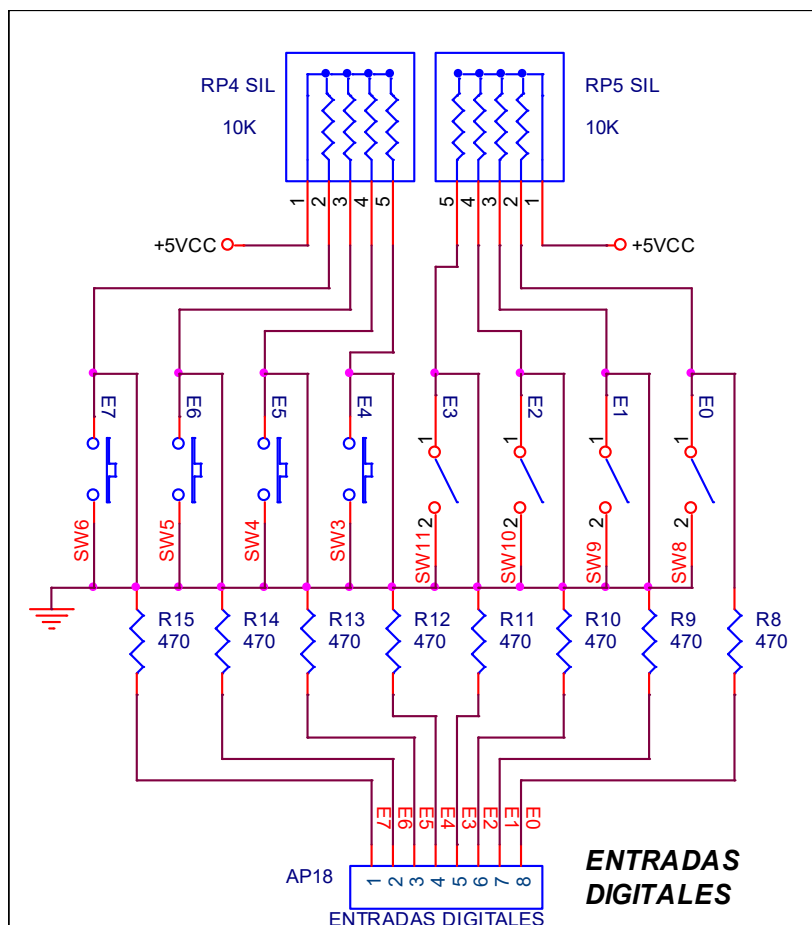
1.3.5 Entradas digitales

Se muestran en la fotografía de la figura 1-12 y están formadas por 4 interruptores deslizantes y otros tantos pulsadores.

Figura 1-12. Entradas digitales

La figura 1-13 muestra el esquema eléctrico de estas ocho entradas cuyas señales están disponibles en el bloque de conexiones AP18. Las señales E0-E3 proceden de los cuatro interruptores deslizantes. Cuando estos están abiertos (hacia arriba) proporcionan nivel lógico "1" gracias a las resistencias Pull-Up de RP5. Cuando cualquiera de ellos se cierra (hacia abajo) se conecta directamente con la señal GND por lo que proporcionan nivel "0".





Por otra parte los cuatro pulsadores en situación de reposo permanecen abiertos generando nivel lógico "1" gracias a las resistencias Pull-Up contenidas en RP4. Cuando se accionan se cierra circuito con GND y se genera nivel "0".

Tanto los interruptores como los pulsadores son compatibles con cualquiera de las entradas digitales del dispositivo PIC que en ese momento esté instalado en el laboratorio USB-PIC'School.

Figura 1-13. Esquema de las entradas digitales.

1.3.6 El generador lógico

Se muestra en la figura 1-14 y está basado en el dispositivo SYM10AA. Es capaz de generar ondas cuadradas simétricas a frecuencias de 1Hz, 10Hz, 100Hz y 1KHz. Está especialmente indicado para aplicaciones de control de tiempos y también para evaluar y ensayar con los módulos de captura y comparación (CCP) contenidos en ciertos modelos de dispositivos PIC.

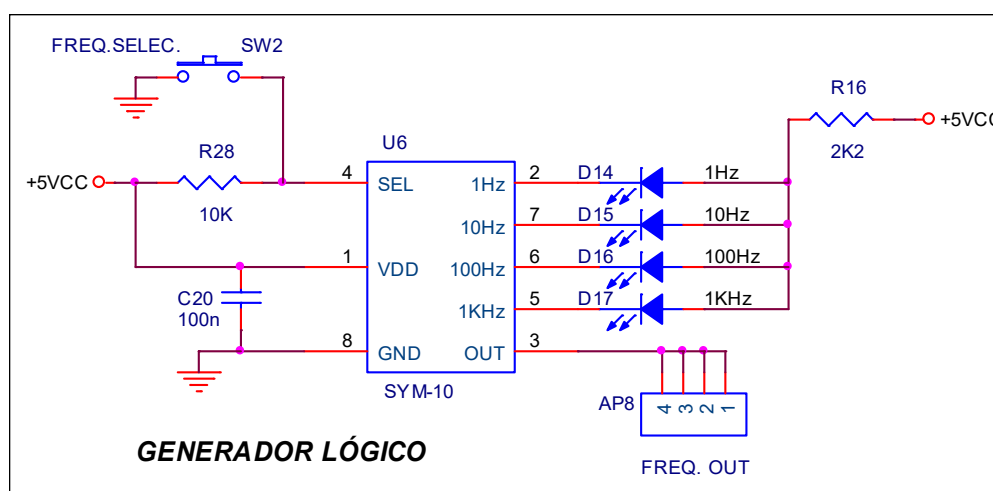


Figura 1-14. El generador lógico

Figura 1-15. Esquema del generador lógico

El esquema eléctrico de dicho generador es el que proporciona su propio fabricante y se muestra en la figura 1-15. Puede servir como base para múltiples aplicaciones. La señal de salida se obtiene mediante el bloque de conexiones AP8. Por defecto la frecuencia de salida es siempre de 1Hz. Mediante el pulsador SW2 se va seleccionando secuencialmente la frecuencia a 10Hz, 100Hz, 1KHz y vuelta a empezar con 1Hz. Los leds D14-D17 monitorizan la frecuencia de salida en todo momento.

1.3.7 El teclado



Figura 1-16. El Teclado

Tal y como se muestra en la figura 1-16, el laboratorio USB-PIC'School va provisto de una teclado matricial de membrana de 4 x 4 teclas. Se trata del periférico de entrada por excelencia que va a permitir introducir todo tipo de datos para su posterior procesamiento.

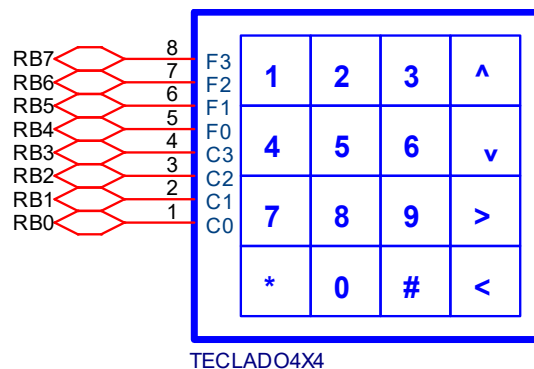
Se trata de un teclado con caracteres hexadecimales y el control del mismo va a suponer tener que aprender a utilizar una serie de técnicas y conceptos que se emplean en las más diversas aplicaciones. Manejar conceptos tales como “barrido del teclado”, “tecla pulsada”, “rebotes”, “interrupción con cada pulsación”, “wake up con cada pulsación”, “teclas multi-carácter”, etc.. darán al usuario la posibilidad de acometer ambiciosos proyectos de carácter profesional y comercial.

Un dato relevante a tener en cuenta es que a pesar de disponer de 16 teclas, tan solo son necesarias 8 líneas de E/S del microcontrolador para su total control. Ello es debido a su distribución matricial. La figura 1-17 muestra el teclado asociado al bloque de conexiones AP19. En dicha figura también se puede apreciar la relación entre fila-columna y la tecla asociada. Así pues la tecla 1 está asociada con la fila 0 (F0) y la columna 0 (C0).

En la laboratorio USB-PIC'School las 8 líneas del teclado se puede conectar a cualquier puerta del PIC, pero se sugiere conectarlo a las líneas de la puerta B. Esto es porque la mencionada puerta tiene, en la mayoría de los dispositivos PIC, dos características muy interesantes para el control de un teclado: resistencias pull-up internas para las líneas de entrada y posibilidad de interrupción cuando cualquiera de esas líneas de entrada cambie de estado.

Figura 1-17. Esquema de conexiones del teclado

Vamos a analizar las conexiones mostradas en la figura 1-17. Las cuatro filas F0-F3 se conectarán a RB4-RB7 y las cuatro columnas C0-C3 se conectarán con las líneas RB0-RB3. La intersección fila-columna da lugar a una tecla en concreto. Es decir, si se pulsa por ejemplo la tecla 4, supone unir eléctricamente la fila F1 con la columna C0, que es tanto como decir que las líneas RB0 y RB5 del PIC se han unido.



La rutina software encargada de explorar el teclado tiene que determinar qué tecla se ha pulsado. Para ello, por ejemplo, se configura las líneas RB0-RB3 (las columnas) como salidas y RB4-RN7 (las filas) como entradas. Secuencialmente se van activando cada una de las columnas al tiempo que se lee el estado de las filas. Cuando se detecta que una fila está activa es porque se pulso una tecla. Basta conocer qué columna se

activó en ese momento para sacar la relación fila-columna que define a cada tecla. Esta tarea, conocida como “barrido de teclado”, ha de repetirse de forma constante y periódica. De esta manera y, a la velocidad de trabajo del PIC, será posible detectar una pulsación en cualquier momento.

Haciendo uso de algunas de las prestaciones que ofrecen los dispositivos PIC en la puerta B, es posible desarrollar rutinas más sofisticadas y eficaces. Efectivamente, programando los registros oportunos de un PIC podemos hacer que las entradas RB4-RB7 (las filas) se conecten a las resistencias pull-up internas que hacen que esas líneas, en estado de reposo, estén a nivel “1”. Por otra parte también podemos habilitar la interrupción por cambio de estado en cualquiera de las entradas de la puerta B.

En este momento las salidas RB0-RB3 (columnas) se ponen a nivel “0” y esta situación de reposo se mantiene mientras no se pulse ninguna tecla. El microcontrolador puede dedicarse a otras tareas o quedarse en standby reduciendo el consumo, hasta que ocurra un suceso como puede ser la pulsación de cualquier tecla. Efectivamente, puesto que RB0-RB3 (columnas) están a “0” y RB4-RB7 (filas) están a “1” gracias a las resistencias pull-Up internas, cuando se pulse cualquier tecla se produce una interrupción por cambio de estado en cualquiera de las líneas RB4-RB7 (filas).

Esa interrupción provoca la inmediata atención del microcontrolador que pasa a ejecutar la rutina que se encargará de averiguar qué tecla se pulso.

1.3.8 Salidas digitales

Están formadas por 8 diodos led (S0-S7) que sirven para monitorizar el nivel lógico de las líneas a las que están conectados. Ver la figura 1-18 y el correspondiente esquema en la figura 1-19.



Figura 1-18. Salidas digitales

Son una total de 8 leds de color azul y alto contraste los que dispone el laboratorio USB-PIC'School para representar el estado lógico de las salidas digitales de una aplicación. A estos leds se accede desde el conector AP17, tal y como se muestra en el esquema de la figura 1-19, y se pueden conectar individualmente a cualquiera de las líneas del microcontrolador. Estas líneas son capaces de suministrar del orden de 25mA por lo que no es necesario ningún circuito adicional de amplificación excepto las resistencias de absorción contenidas en el pack RP2.

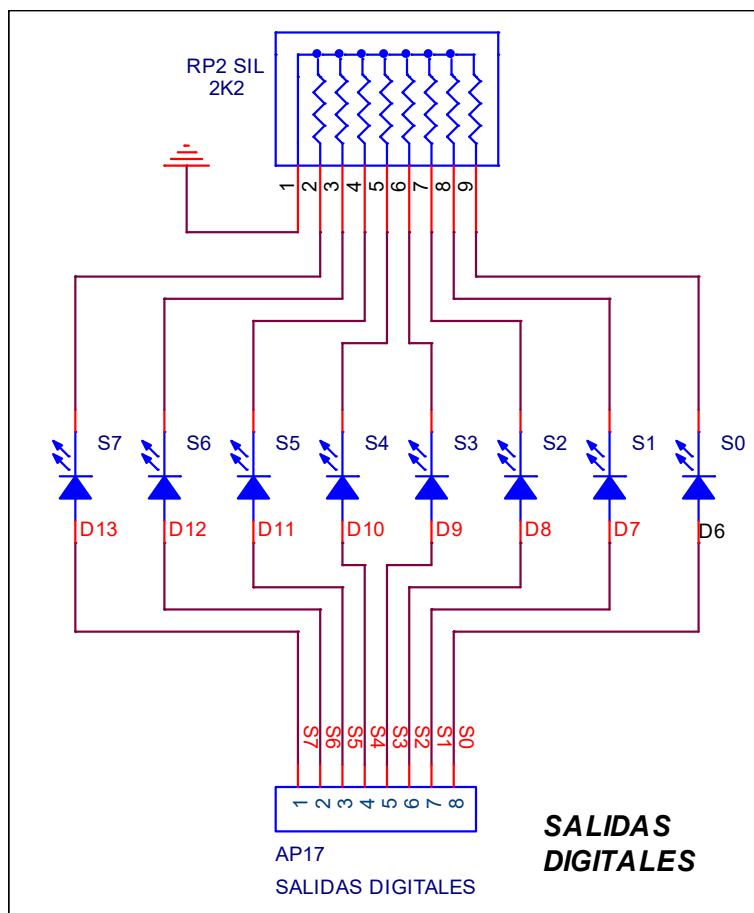
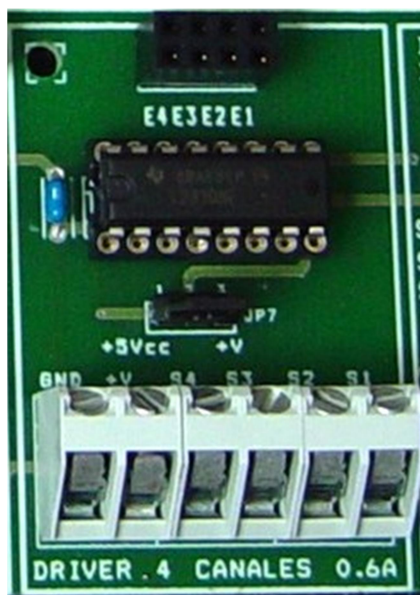


Figura 1-19. Esquema de las salidas digitales

Un nivel lógico “1” por cualquiera de esas líneas provoca el encendido del led correspondiente. Un nivel “0” lo apaga. Es una forma muy simple y económica de reflejar el estado binario de las líneas de salida, donde cada led simula la carga que se desea controlar.

1.3.9 Salidas digitales de alta corriente

El laboratorio USB-PIC'School dispone de un driver de 4 canales formado por el dispositivo L293D y capaz de proporcionar salidas de alta corriente. Ver la fotografía de la figura 1-20 .



a 1-20. Las salidas de alta corriente

Figur

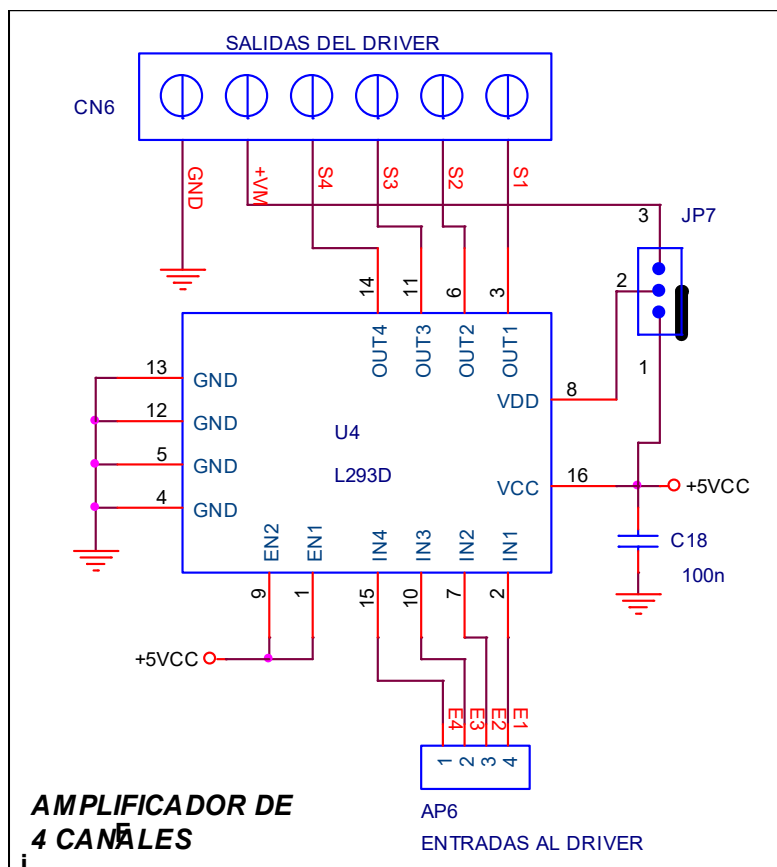


Figura 1-21. Esquema de las salidas de alta corriente.

En la figura 1-21 se puede apreciar el esquema eléctrico. El dispositivo integrado L293D dispone de 4 canales amplificadores o drivers capaces de proporcionar salidas de hasta 0.6 A a 36V cada uno, a partir de señales TTL de baja corriente.

Mediante el conector AP6 se conectan las señales E1-E4 a amplificar y que proceden de sendas salidas del microcontrolador. El conector CN6 está formado por un conjunto de bornas o clemas donde se obtienen las salidas S1-S4 debidamente amplificadas y se realizan las conexiones con las cargas a controlar.

Mediante el jumper JP7 se selecciona la tensión de salida para las cargas. En la posición 1-2 (por defecto) dicha tensión es de +5Vcc. En la posición 2-3 la tensión que alimenta a las cargas se aplica externamente a través de la borna +VM.

Mediante estas salidas de alta corriente el laboratorio USB-PIC'School puede controlar cargas de consumo elevado como pueden ser lámparas incandescentes, relés, motores DC, motores paso a paso, etc..

Cabe indicar que el dispositivo L293D integra en su interior sendos diodos para la absorción de corrientes inversas provocadas por cargas inductivas.

1.3.10 Interface serie RS-232



Su ubicación dentro del laboratorio se muestra en la figura 1-22. Consiste en el clásico interface serie que permite adaptar los niveles lógicos del microcontrolador a niveles RS-232.

Figura 1-22. Canal serie RS-232

Según el esquema eléctrico mostrado en la figura 1-23 el interface está formado por el popular adaptador de niveles MAX-232. Mediante el conector AP5 disponemos de las señales de transmisión y recepción (TxD y RxD) así como las de control de flujo CTS y RTS.

Estas señales proceden del microcontrolador. El conector CN5 es un conector DB9 hembra estándar, que permite realizar la conexión con el periférico serie. Mediante un led bicolor (D5) se monitoriza la transmisión y/o recepción.

Este interface permite realizar todo tipo de comunicaciones serie entre el laboratorio USB-PIC'School y cualquier otro equipo mediante el protocolo estándar RS-232. La velocidad de transferencia irá en función del tipo de microcontrolador empleado, su velocidad de trabajo y de su configuración.

Se recuerda que algunos dispositivos PIC incorporan en su interior una UART completa que se encarga de realizar la mayor parte de los procedimientos propios de la comunicación según los protocolos tanto RS-232 como RS-485. En este último caso basta con montar el sencillo circuito de adaptación correspondiente, sobre el módulo board de montaje sin soldadura con que cuenta USB-PIC'School.

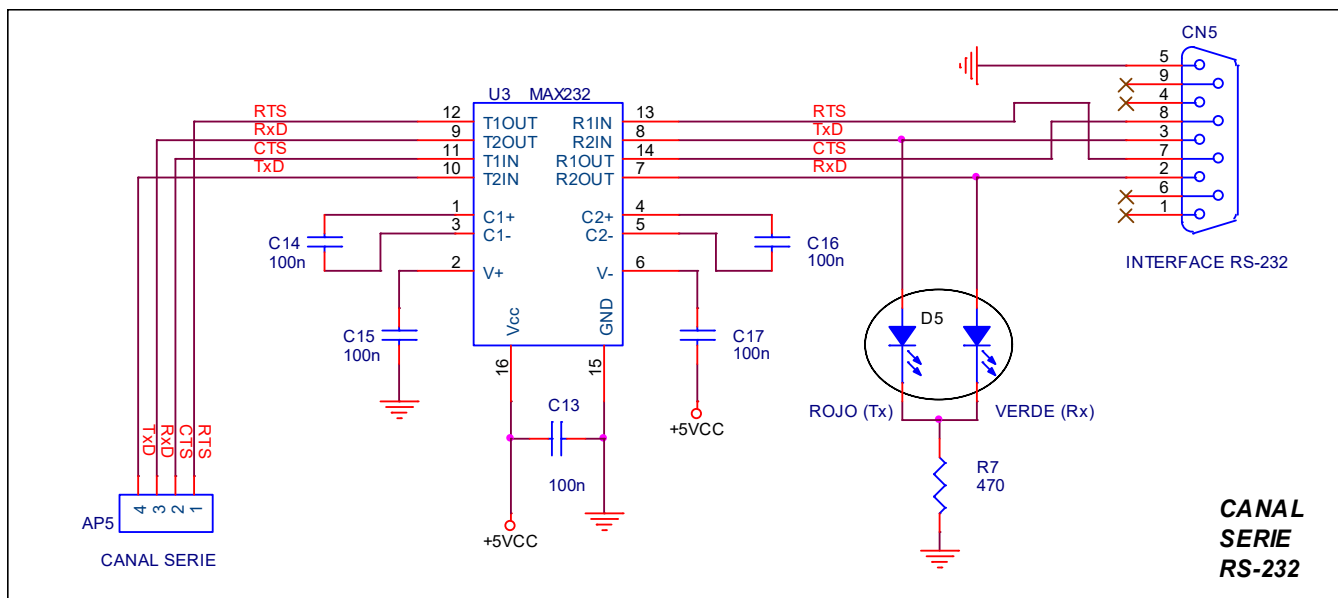
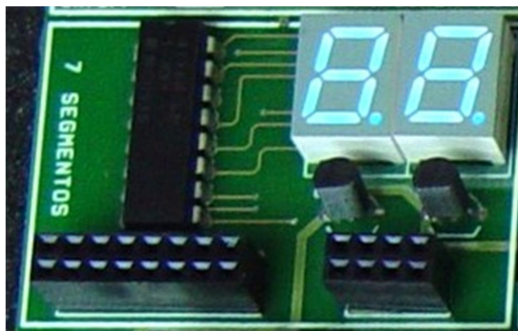


Figura 1-23. Esquema eléctrico del interfase RS-232

1.3.11 Displays de 7 segmentos



Se muestra en la figura 1-24 y consiste en dos displays de 7 segmentos de 0.5", color azul y alto contraste.

Figura 1-24. Los displays de 7 segmentos

El display de 7 segmentos es uno de los periféricos de salida más empleado para la visualización de datos numéricos. En el caso del laboratorio USB-PIC'School disponemos de dos displays de ánodo común conectados tal y como se muestra en el esquema de la figura 1-25

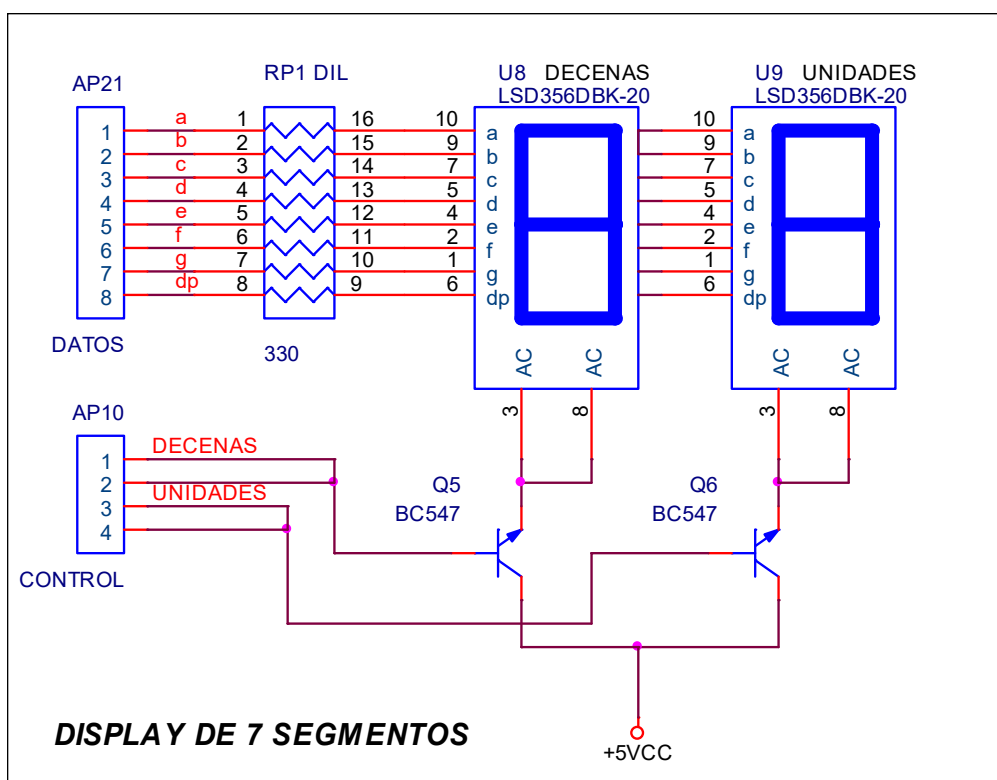


Figura 1-25. El display de 7 segmentos

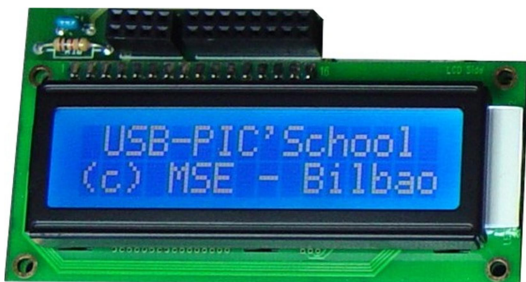
Ambos displays se encuentran multiplexados y con un pack de resistencias de absorción común a ambos. Mediante el conector AP21 se accede a cada uno de los segmentos "a ... dp". Se tratan de displays de ánodo común por lo que para la activación de un segmento en particular es necesario aplicarle un nivel lógico "0". Mediante el conector de control AP10 se accede a los ánodos de cada display, el de unidades y el de decenas. Estos se activan mediante niveles lógicos "1" a través de los amplificadores de corriente formados por los transistores Q5 y Q6.

La disposición de displays multiplexados es la configuración más habitual utilizada en las aplicaciones prácticas, por el ahorro de conexiones que supone. Todos los segmentos se conectan en paralelo y cada display se controla individualmente mediante su común.

Así pues, cuando se aplica un código binario a los segmentos, debemos también indicar a qué display va dirigido. Para ello, en nuestro caso, se activa el ánodo común deseado: el de las decenas o el de las unidades.

El multiplexado de displays consiste en ir activando secuencialmente cada uno de los displays al tiempo que se le aplica el código de 7 segmentos que se desea visualice. Esto se debe repetir a una determinada velocidad para provocar la sensación óptica de que todos los displays están activados.

1.3.12 La Pantalla LCD



Se trata de uno de los periféricos más versátiles e interesantes que dispone el laboratorio USB-PIC'School. Se muestra en la figura 1-26, y es capaz de visualizar sobre un fondo azul dos líneas de 16 caracteres alfanuméricos cada una.

Figura 1-26. La pantalla LCD

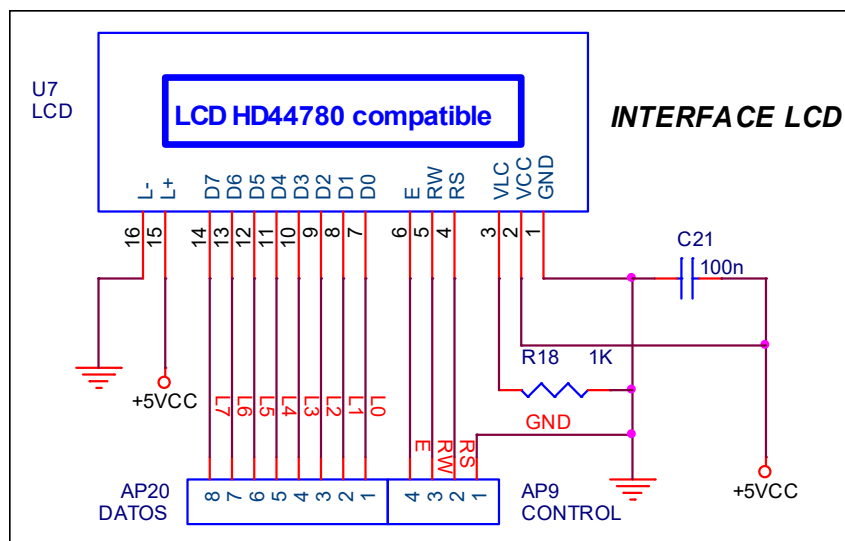
Este potente periférico de salida va a permitir representar cualquier tipo de mensaje compuesto de letras, números y símbolos produciendo además diferentes efectos de visualización como desplazamientos a izquierda y derecha, parpadeos, scrolls, etc.

La transferencia de información entre la pantalla LCD y el microcontrolador se realiza en paralelo en grupos de 4 u 8 bits de datos y se conecta con las líneas de E/S mediante el conector AP20. El esquema eléctrico se muestra en la figura 1-27.

Existe además otro conector, el AP9, donde se conectan las señales que controlan el flujo y tipo de información que se transfiere. Con la señal RS se determina si la pantalla recibe el código ASCII del carácter a visualizar (nivel "1") o bien un código de instrucción (nivel "0"). Mediante la señal RW el microcontrolador informa a la pantalla LCD si se va a leer o escribir sobre la misma. Finalmente, con la señal E, se habilita o no a esa pantalla. Cuando esta señal vale "0" la pantalla queda desconectada, las líneas de datos quedan en alta impedancia. Cuando vale "1", la pantalla queda habilitada para recibir o mandar datos.

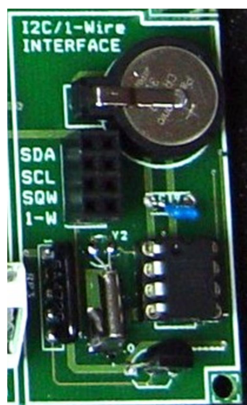
La resistencia R18 permite ajustar el contraste del LCD. Con el valor de 1K se consigue un contraste medio aceptable. Su valor se puede modificar.

Figura 1-27. Conexiones de la pantalla LCD



En el CDROM se incluye la información técnica que proporciona el fabricante de la pantalla LCD.

1.3.13 Dispositivos I2C y 1-wire



Dentro de las comunicaciones serie, merecen una especial mención los protocolos 1-wire e I2C. Se trata de una comunicación en la que se emplean tan sólo uno o dos hilos. Por ello se hace muy interesante en aquellas aplicaciones en las que el controlador necesite comunicarse con diferentes dispositivos existentes en el mercado, y que emplean dichos protocolos.

En el laboratorio USB-PIC'School nos hemos decidido por el empleo de dos de los dispositivos más representativos de esos protocolos de comunicación: El reloj en tiempo real (RTC) DS1307 y el sensor de temperatura DS18S20. Ver la figura 1-28.

Figura 1-28. Interface I2C y 1-wire

El esquema de conexiones se muestra en la figura 1-29. El DS1307 consiste en un reloj/calendario en tiempo real (RTC) que gestiona todas las funciones horarias típicas y dispone también de 56 bytes de memoria RAM para el almacenamiento de datos de propósito general. Dado su reducido consumo, se le puede alimentar mediante una pila de 3V. Esta permite mantener al reloj en hora y retener los datos de la memoria RAM aun a falta de la alimentación general, durante unos 3 años aprox. El protocolo de comunicación empleado entre este dispositivo y el controlador es el I2C, con el que se pueden realizar todas las tareas horarias necesarias así como la gestión de datos en RAM.

Por su parte el dispositivo DS18S20 consiste en un sensor de temperatura con una resolución de $\pm 0.5^{\circ}\text{C}$. La comunicación con el controlador se realiza mediante el protocolo 1-wire.

Disponemos pues de 2 dispositivos que abren un gran número de aplicaciones y posibilidades, a la vez que nos permiten experimentar con los protocolos I2C y 1-wire de comunicación.

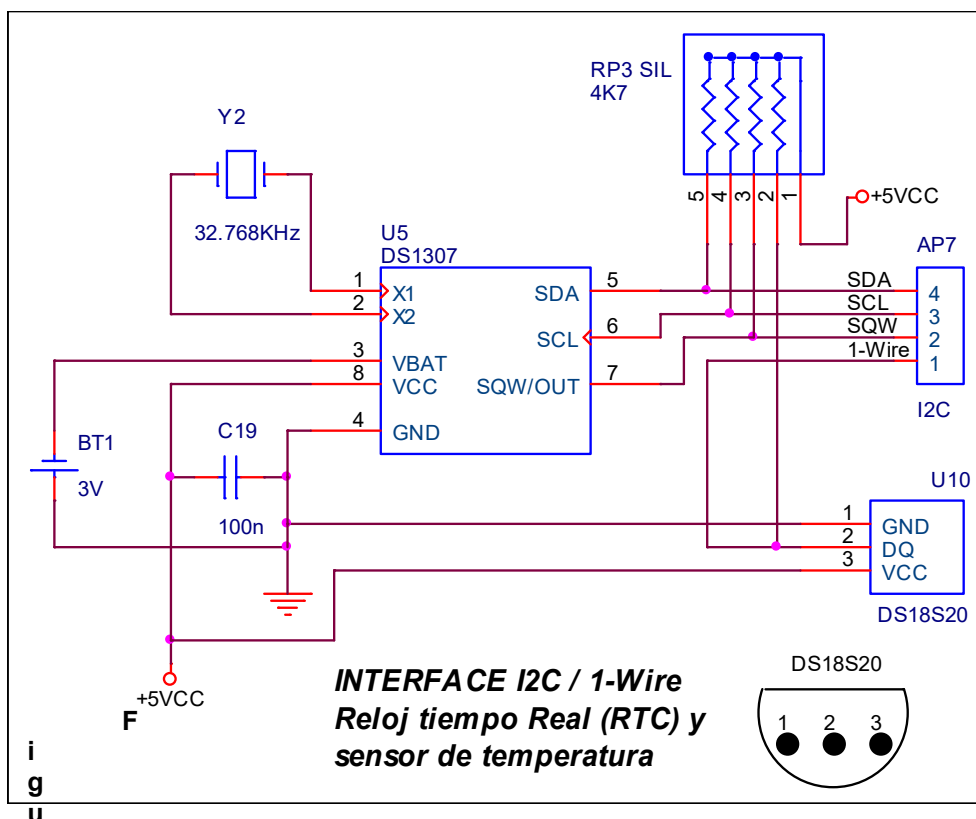
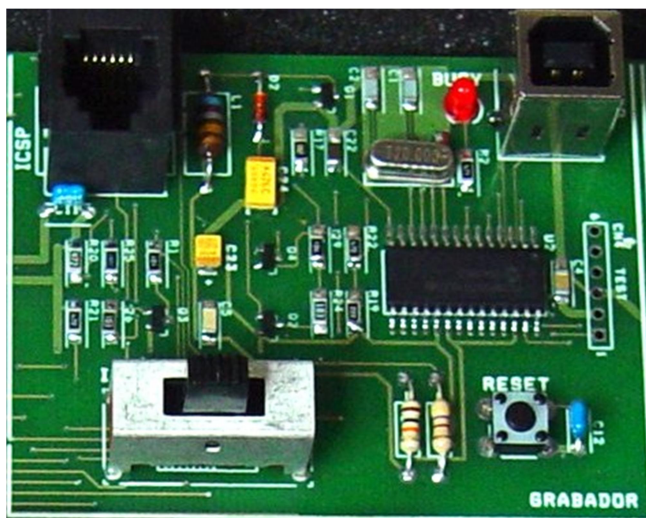


Figura 1-29. Esquema de conexiones de los dispositivos I2C y 1-wire

En el CDROM se adjunta la información técnica que suministra el fabricante de ambos dispositivos, así como librerías para su gestión y control.

1.3.14 Sección de Depuración/Grabación



Corresponde con el hardware mostrado en la figura 1-30 y se encarga, mediante el software de control apropiado, de la depuración del programa y/o de la grabación del mismo en la memoria del controlador.

Uno de los detalles más destacables del laboratorio USB-PIC'School es que este circuito para la depuración/grabación es compatible con el sistema PICKIT-2 de Microchip. Esto hace posible que todas las tareas de edición de programas, ensamblado/compilado de los mismos, depuración y grabación se puedan realizar desde el conocido y potente entorno de desarrollo MPLAB-IDE original de Microchip.

Figura 1-30. Sección de grabación/depuración

El esquema por bloques de la figura 1-31 trata de sintetizar el funcionamiento del circuito de Depuración/Grabación que dispone el laboratorio USB-PIC'School.

Todos los dispositivos PIC disponen de 3 señales para la lectura/grabación de la memoria de programa, la memoria EEPROM para datos y la memoria de configuración. La señal PICDATA permite transferir los datos de forma serie y síncrona entre el PIC y el hardware de grabación, la señal PICCLK transporta la señal de reloj para el sincronismo de los datos y la señal PICMCLR, que inicia los ciclos de lectura/grabación y aplica la tensión V_{pp} necesaria. Estas tres señales son además compartidas y empleadas como líneas de E/S de propósito general una vez que el PIC ha sido grabado.

Estas tres señales se conectan con tres fuentes diferentes según el modo de operación que seleccionemos seleccionado mediante el conmutador SW12.

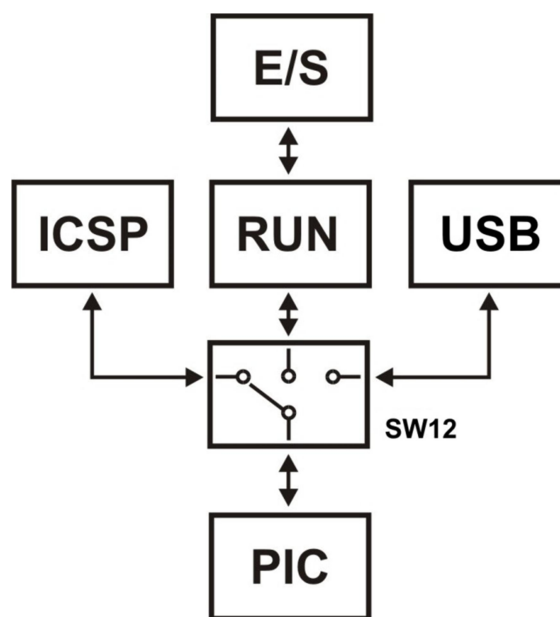


Figura 1-31. Esquema por bloques del circuito de grabación

El modo **RUN** se emplea cuando el dispositivo está grabado. Las tres señales del PIC anteriormente mencionadas se consideran líneas de E/S de propósito general y están disponibles a través de los bloques de conexión correspondientes, para ser utilizadas por los periféricos de la aplicación.

En el modo **ICSP** el laboratorio queda configurado para que el PIC actual en él insertado se pueda Depurar/Grabar a través de las señales que se aplican mediante el conector RJ11 (ICSP- In Circuit Serial Programming). Estas señales a su vez las debe proporcionar un depurador/grabador externo dotado de su correspondiente software. Por ejemplo entre otros, se pueden emplear los sistemas ICD2 o ICD3 de Microchip que también, al igual que nuestro USB-PIC'School, se controlan y gestionan mediante el entorno integrado de desarrollo MPLAB-IDE. Este se puede obtener y actualizar de forma gratuita desde www.microchip.com.



En el modo **USB** la Depuración/Grabación del PIC se realiza a través del puerto USB que se supone conectado al PC. Un diodo led, el BUSY, indica transferencia de datos entre el PC y el laboratorio. Como ya se comento anteriormente, el sistema USB-PIC'School es compatible con el sistema PICKIT-2 de Microchip, por lo que se puede emplear el entorno de desarrollo MPLAB-IDE.

Conviene indicar que, aunque el software de desarrollo MPLAB-IDE admite prácticamente todos los dispositivos PIC existentes, es necesario que también haya compatibilidad a nivel de hardware. **No todos los PIC's son compatibles con las mismas herramientas hardware.** Esta compatibilidad viene dada fundamentalmente por la distribución de patillas del dispositivo en cuestión, tal y como se explicó en el apartado 1.3.3. En el laboratorio USB-PIC'School se admiten todos los dispositivos PIC's de 8, 18, 28 y 40 patillas que estén distribuidas como se menciona en dicho apartado y que también sean soportados por la herramienta hardware PICKIT-2 de Microchip y su software MPLAB-IDE.

En el siguiente tema de este manual se hará una clara explicación sobre la depuración/grabación de aplicaciones sobre nuestro laboratorio USB-PIC'School.



TEMA 2: El Depurador/Grabador

2.1 INTRODUCCION

Una de las grandes aportaciones del laboratorio USB-PIC'School es la inclusión de un nuevo hardware y firmware que permite la depuración/grabación en circuito de la aplicación que se está desarrollando.

Disponemos pues de una potente herramienta gracias a la cual podemos, en fase de diseño, realizar la depuración en tiempo real y la puesta a punto de los programas en diseño. Una vez depurados y funcionando, podemos también proceder a grabar esos programas sobre la memoria interna del controlador. Se trata pues de una herramienta imprescindible en centros de enseñanza, laboratorios, centros de I+D, Ingenierías y, en general, diseñadores de aplicaciones basadas en controladores PIC.

Tanto el hardware como el firmware interno del depurador/grabador de nuestro laboratorio USB-PIC'SCHOOL **es compatible con el sistema PICKit-2 de Microchip**. Esto permite que, como software de control, se pueda utilizar el conocido entorno de desarrollo MPLAB-IDE desde el cual se realizan todas las operaciones típicas en el desarrollo de un programa: edición del programa fuente, ensamblado, depuración y grabación. También se puede emplear el software de grabación PICKit 2. Ambas son herramientas software de libre distribución y originales de Microchip, que se pueden descargar y/o actualizar desde www.microchip.com. Las pruebas realizadas por MK Electrónica así como todos los ejemplos que acompañan a USB-PIC'School, se hicieron con las siguientes versiones:

- **MPLAB-IDE V8.36**
- **PICKit 2 V2.61**

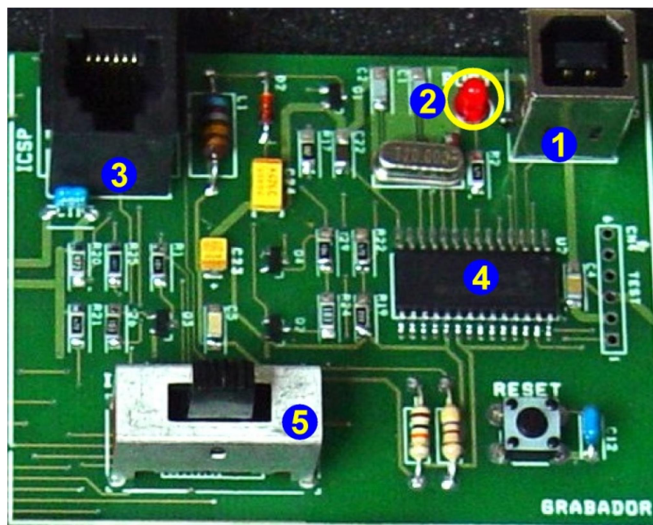
Las capacidades de depuración del USB-PIC'SCHOOL permiten la ejecución del programa de aplicación en tiempo real, con puntos de ruptura (BreakPoints) y paso a paso. Así, es posible visualizar los registros internos y hacer un seguimiento analizando el contenido de los mismos conforme se van ejecutando las instrucciones del programa. También es posible modificar ese contenido para estudiar y seguir la repercusión que esto puede provocar en el curso de la ejecución. Todo ello en tiempo real y sin extraer el PIC de nuestro laboratorio, empleando la técnica conocida como ICD=In Circuit Debugger (Depuración en Circuito).

Las limitaciones de depuración/grabación del USB-PIC'SCHOOL son las mismas que las del PICKit-2 de Microchip y están sujetas a las limitaciones inherentes al controlador que se vaya a emplear. Algunos controladores, los más antiguos, no admiten depuración en circuito (ICD). Algunos admiten un único punto de ruptura o breakpoint y otros admiten varios. Para este tipo de detalles es necesario consultar la información técnica específica del controlador y que suministra el fabricante en sus correspondientes data sheet y/o en el entorno de desarrollo MPLAB-IDE. En el CDROM se adjunta la información técnica del PIC16F866 que se incluye en el laboratorio.

2.2 DESCRIPCION

La figura 2-1 muestra la sección de Depuración/Grabación que integra el laboratorio USB-PIC'School y en donde se destacan los elementos más importantes, que se comentan a continuación:

Figura 2-1. Descripción de la sección de Depuración



1. Se trata de un conector USB hembra tipo B para la conexión con un puerto USB del PC. Para ello se emplea un cable adaptador macho tipo B a macho tipo A, que se incluye en el equipo.
2. Este led se activa siempre que el USB-PIC'SCHOOL esté realizando algún ciclo activo transfiriendo información hacia/desde el PC a través del puerto USB.
3. Conector RJ11 para la conexión, en modo ICSP, con otros dispositivos externos que se encarguen de la Depuración/Grabación. Contiene las señales propias de grabación: PGC, PGD, GND, Vcc y MCLR/Vpp, conocidas como “señales ICSP” In Circuit Serial Programming (Programación Serie en Circuito). Están distribuidas como se muestra en la figura 2-2 y en la siguiente tabla.

NºPIN	SEÑAL	DESCRIPCION
1	MCLR/Vpp	Señal Vpp de grabación
2	VDD	Alimentación +VDD
3	GND	Tierra de alimentación
4	PGD	Señal de datos
5	PGC	Señal de reloj
6	NC	No conectada

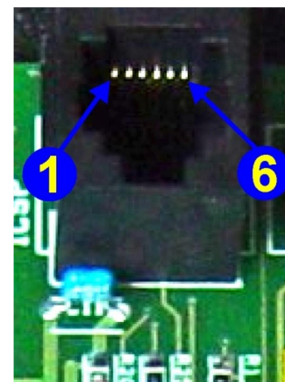


Figura 2-2. Conexiones ICSP

4. Controlador cuyo firmware realiza las tareas de Depuración/Grabación y comunicación con el PC. Emula y es compatible con el PICKit de Microchip.
5. Conmutador para la selección del modo de trabajo. Hay tres:
 - a. **ICSP:** Se trabaja con un sistema de depuración externo al USB-PIC'School. Este se debe conectar a través del conector RJ11.
 - b. **RUN:** Se emplea cuando el controlador ya está debidamente grabado con un programa totalmente funcional y no necesita de sistema de Depuración/Grabación alguno.
 - c. **USB:** Es el modo que seleccionamos cuando estamos en fase de Depuración y/o de Grabación desde el PC.

2.3 EL ENTORNO DE DESARROLLO MPLAB-IDE EMPLEADO PARA LA GRABACION

Se trata de un software de libre distribución original de Microchip que se puede descargar y/o actualizar desde www.microchip.com. Es conveniente hacerlo de forma periódica con objeto de obtener la última versión disponible donde se van corrigiendo los fallos y errores encontrados en versiones previas. Se descarga un fichero ZIP que contiene los archivos ejecutables, archivos de configuración, lenguajes, etc. El proceso de instalación es totalmente automático y basta con seguir las instrucciones. A la hora de escribir el presente documento hemos empleado la versión MPLAB-IDE V8.36.

Tanto en el mercado como en la red podemos encontrar gran cantidad de herramientas software, lenguajes y aplicaciones dedicadas a los microcontroladores PIC. Sin embargo, el entorno MPLAB-IDE es la herramienta software original, fundamental y de referencia que propone Microchip para el desarrollo de aplicaciones basadas en sus controladores. En MK Electrónica, aunque no podemos proporcionar su soporte técnico, la empleamos a diario y, a pesar de estar en Inglés, la recomendamos encarecidamente. Por ello lo incluimos en el disco de usuario así como su correspondiente manual.

MPLAB-IDE consiste en un entorno de trabajo que permite realizar, sin salir del mismo ni emplear otras herramientas software, todas las tareas propias del diseño de aplicaciones basadas en microcontroladores PIC. Sus características y posibilidades más importantes se podrían resumir a continuación:

- Completo editor de textos para la edición de los programas fuente.

- Admite cualquier dispositivo PIC de cualquiera de las familias disponibles.
- De forma natural integra el lenguaje ensamblador para todos los dispositivos PIC existentes. Opcionalmente también puede asociarse y trabajar con compiladores de lenguajes de alto nivel (p.e. el C) de las firmas más prestigiosas (en los ejemplos que proponemos hemos empleado el lenguaje C en la versión PCM de la firma CCS).
- Integra un simulador por software que permite simular la ejecución de un programa sobre cualquiera de los dispositivos PIC existentes: ejecución paso a paso, modificación/visualización de registros internos, puntos de ruptura y mucho más.
- Acepta cualquiera de las herramientas hardware para la depuración, emulación y grabación que desarrolla Microchip: los depuradores ICDx, los emuladores ICE2000 e IC4000 y Real ICE, así como grabadores como el PROMATE, PIC START PLUS y el PICKit-2 **compatible con nuestro USB-PIC'SCHOOL**.

En el presente documento no pensamos hacer un manual de usuario del MPLAB-IDE. En el CDROM que acompaña a nuestro laboratorio USB-PIC'SCHOOL se incluye una copia original en Ingles del manual que proporciona Microchip.

Lo que si haremos será un pequeño resumen de todas las operaciones **relacionadas con la grabación** de los dispositivos instalados en **USB-PIC'School** que, a todos los efectos, es compatible con la herramienta PICKit-2 de Microchip.

¡¡ IMPORTANTE !! para realizar cualquier operación de depuración y/o grabación es **imprescindible colocar el conmutador en la posición USB**. Una vez grabado el controlador con un programa funcional, dicho conmutador se coloca en la posición **RUN** para ejecutar la aplicación de forma autónoma.

2.3.1 Instalación

La instalación de MPLAB-IDE es totalmente automática y no supone ninguna complicación. Basta con seguir paso a paso las instrucciones que van apareciendo. En la figura 2-3 se muestra la pantalla principal de MPLAB-IDE para hacer una breve descripción de las opciones del menú principal.

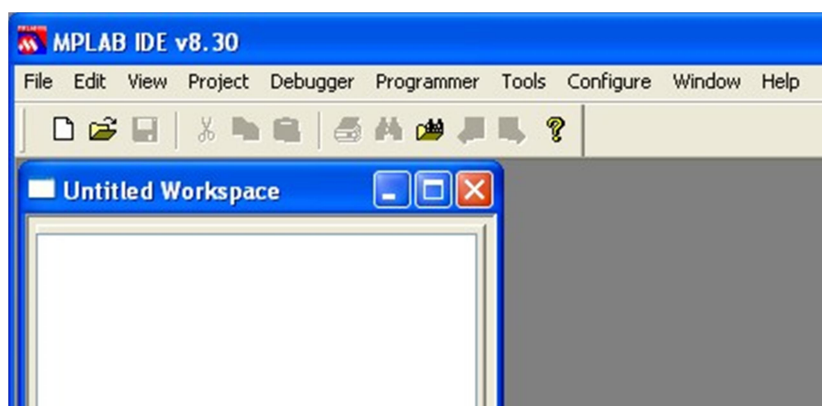


Figura 2-3. El MPLAB-IDE

Opciones del menú principal.

File	Contiene todas las opciones y comandos relacionados con el manejo de ficheros: abrir fichero, salvar fichero, imprimir, importar, exportar, etc..
Edit	Contiene todas las opciones y comandos típicos de cualquier editor de texto: buscar, buscar y sustituir, cortar, pegar, marcar, etc..
View	Con esta opción podemos abrir, visualizar y organizar sobre el área de trabajo diferentes tipos de ventanas relacionadas con el PIC seleccionado: memoria de programa, memoria de datos, registros especiales, EEPROM, etc..
Project	Presenta todos los comandos y opciones relacionados con los proyectos: abrir proyecto, cerrar, salvar proyecto, ensamblar y/o compilar, etc..

- Debugger** Presenta comandos y utilidades propios de la depuración de un programa de aplicación: seleccionar herramienta de depuración/emulación (simulador, PICKit-2, ICE2000, ICD2 y otros), ejecución, ejecución paso a paso, puntos de ruptura, reset y mucho más. Seleccionaremos el PICKit-2 que es compatible con nuestro USB-PIC'SCHOOL. En el apartado 2.5 se explica el funcionamiento del MPLAB-IDE empleado como depurador.
- Programmer** Presenta comandos y utilidades propios de la grabación de un programa sobre la memoria física del PIC: seleccionar grabador (PICKit-2, PIC START PLUS, PROMATE y otros). Seleccionaremos el PICKit-2 que es compatible con nuestro USB-PIC'SCHOOL. Desde este menú disponemos también de los comandos para leer un PIC, grabarlo, verificar, borrarlo, etc..
- Tools** Mediante este menú se accede a una serie de utilidades y herramientas software que, en sucesivas versiones de MPLAB-IDE, van apareciendo.
- Configure** Permite realizar diferentes tareas de configuración: seleccionar modelo de PIC, ajustar su palabra de configuración e identificación, realizar ajustes del propio entorno MPLAB-IDE, etc..
- Windows** Selecciona diferentes formas de visualización de las distintas ventanas que podemos visualizar sobre el área de trabajo.
- Help** Contiene un completo sistema de ayuda general (en inglés) para el MPLAB-IDE.

2.3.2 Selección de dispositivo

Una vez que el MPLAB-IDE está en ejecución, una de las primeras tareas a realizar será seleccionar el modelo de PIC con el que vamos a trabajar que en nuestro caso será, p.ej., el PIC16F886. En el menú principal seleccionamos *Configure* → *Select Device*. Aparece una ventana como la mostrada en la figura 2-4.

En el desplegable “**Device:**” seleccionamos el modelo de PIC, el PIC16F886 en el ejemplo. Automáticamente veremos un resumen de herramientas tanto hardware como software que el MPLAB es capaz de soportar con el modelo recién seleccionado. Las señales de color verde indican herramientas disponibles.

Las rojas representan a herramientas no disponibles en el MPLAB-IDE para el dispositivo seleccionado. También pueden aparecer señales amarillas que indican que la herramienta en cuestión se encuentra en fase de testeo y no es definitiva.

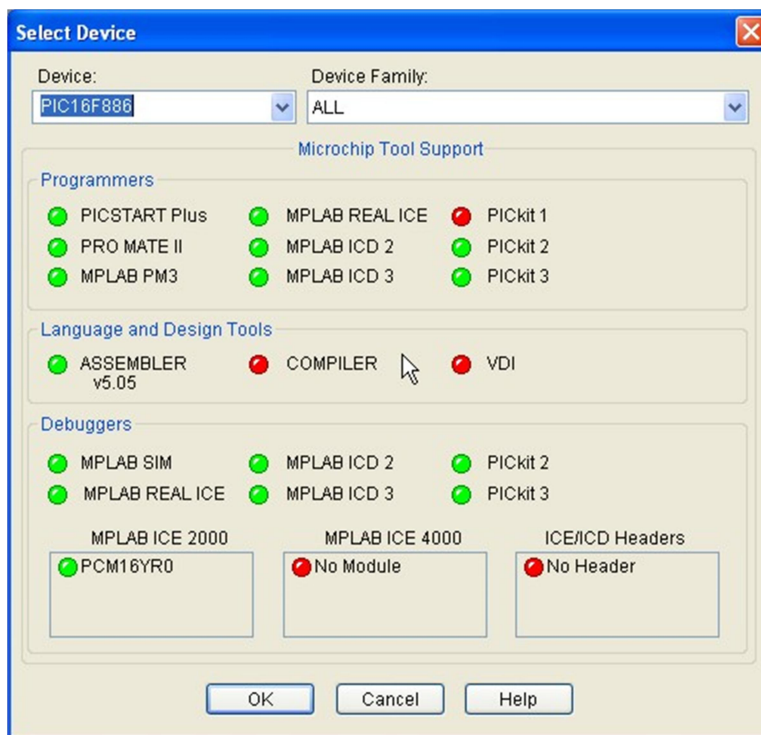


Figura 2-4. Seleccionando un dispositivo

En el ejemplo hemos seleccionado el PIC16F886. A la vista de la figura anterior podemos resumir qué herramientas puede emplear el MPLAB-IDE con dicho modelo. Cabe destacar que entre ellas se encuentra el Grabador/Depurador PICKit-2 que, como hemos venido repitiendo, **es compatible con nuestro USB-PIC'SCHOOL**.

2.3.3 Seleccionando el grabador

Seleccionado el dispositivo (PIC16F886) y suponiendo que el USB-PIC'SCHOOL está conectado a uno de los puertos USB del PC, elegimos ahora el tipo de grabador que vamos a emplear. En el menú principal seleccionamos *Programmer* → *Select Programmer*. Aparece un desplegable como el mostrado en la figura 2-5.

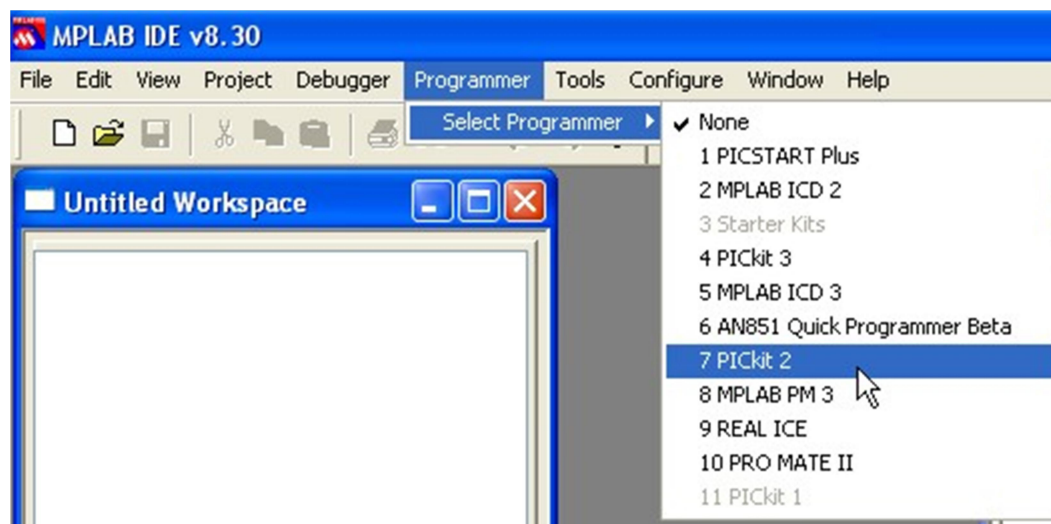


Figura 2-5. Tipos de grabadores admitidos por la versión actual del MPLAB-IDE

Vemos una lista de los sistemas de grabación que el MPLAB-IDE puede emplear para grabar el dispositivo PIC16F886 seleccionado. En nuestro caso seleccionamos el PICKit-2 que **es compatible con el USB-PIC'SCHOOL** que disponemos en estos momentos.

Inmediatamente el MPLAB-IDE trata de establecer conexión con nuestro USB-PIC'SCHOOL (compatible con PICKit-2). Si todo está va bien nos debe aparecer en la ventana "Output" una serie de mensajes como los mostrados en la figura 2-6, que nos indican que se ha localizado el PICKit-2, la alimentación es de 4.98V y que el PICKit-2 (USB-PIC'School) se encuentra listo y conectado.

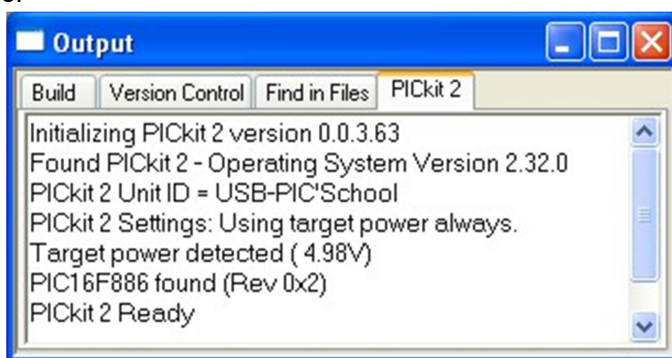


Figura 2-6. Conexión con PICKit-2 (USB-PIC'SCHOOL)

Si en el sócalo del laboratorio estuviera insertado el PIC seleccionado anteriormente, también aparecería el mensaje "PIC16F886 found (Rev 0x2)" indicando que dicho PIC ha sido identificado. En caso de que el PIC no estuviera insertado en el zócalo, aparecería el mensaje "PKWarn0003: Unexpected device ID: Please verify that a PIC16F886 is correctly installed in the application. (Expected ID = 0xE00, ID Read = 0x0)". Esto nos recuerda que debemos insertarlo para poder proceder con las operaciones propias del grabador.

2.3.4 Ficheros HEX y buffers de memoria

El fichero HEX es el que se obtiene tras el proceso de ensamblado y/o compilado de un programa fuente y cuyo contenido se grabará en las diferentes memorias del controlador. Efectivamente, el contenido de un fichero HEX puede estar compuesto de tres tipos de información que irán a parar a sus correspondientes regiones de memorias dentro del controlador:

- *Memoria de programa.* Contiene el código hexadecimal de las instrucciones del programa que el controlador debe ejecutar.
- *Memoria de datos.* Contiene información adicional o datos que se grabarán en la memoria EEPROM de datos del controlador.
- *Memoria de configuración.* Contiene información sobre la configuración del controlador y valores de identificación.

El MPLAB-IDE dispone de otros tantos buffers de memoria. Cada vez que se ensambla o compila un programa fuente se genera el correspondiente fichero HEX y esos buffers se llenan con el resultado hexadecimal de ese ensamblado de forma que, posteriormente, puedan copiarse o grabarse sobre la memoria física del PIC.

Mediante la comando *File → Import* también es posible importar ficheros HEX procedentes de otras fuentes distintas a las generadas por el MPLAB-IDE. Sea como fuere, cada vez que se abre un fichero HEX, los tres buffers de memoria del MPLAB-IDE se llenan con sus correspondientes contenidos: memoria, datos y configuración, para su posterior grabación sobre el PIC.

Los buffers del MPLAB también se pueden llenar manualmente con contenidos hexadecimales, mediante los correspondientes comandos:

- *View → Program Memory:* Permite visualizar y modificar el contenido del buffer correspondiente a la memoria de programa
- *View → EEPROM:* Permite visualizar y modificar el contenido del buffer correspondiente a la memoria EEPROM de datos.
- *Configure → Configuration Bits:* Permite visualizar y modificar los bits de configuración del buffer correspondiente a la memoria de configuración.
- *Configure → ID Memory:* Permite visualizar y modificar los datos ID de identificación del buffer correspondiente a la memoria de configuración.

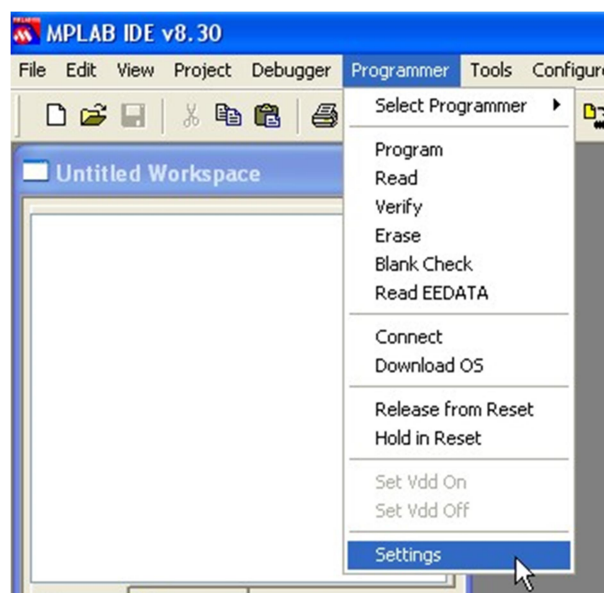
Una vez hemos abierto u obtenido un fichero HEX, los tres buffers de memoria del MPLAB-IDE contendrán la información necesaria que deberá grabarse en las memorias del controlador, tal y como veremos a continuación.

2.3.5 Funciones más comunes del grabador

Cuando se ha seleccionado el tipo de grabador (PICKit-2 = USB-PIC'SCHOOL) y se ha realizado la conexión con él, se activan todas las opciones que se encuentra en el menú "*Programmer*" y que son propias de un típico grabador. Ver la figura 2-7.

La primera opción, "*Select Programmer*", se empleó anteriormente para seleccionar el tipo de grabador que vamos a emplear. En nuestro caso seleccionamos el PICKit-2 que es compatible con nuestro USB-PIC'SCHOOL. A continuación se detallan las opciones restantes.

Figura 2-7. Funciones típicas del grabador



Program

Realiza el proceso de grabación propiamente dicho. Copia los contenidos de la memoria de programa, memoria de datos y memoria de configuración actual en los buffer del MPLAB-IDE, sobre las correspondientes memorias físicas del controlador.

Read

Lee los contenidos actuales de las memorias de programa, datos y configuración del controlador y los deposita sobre los correspondientes buffers del MPLAB-IDE.

Verify

Verifica que los contenidos actuales de las memorias físicas de programa, datos y configuración del controlador sean idénticos a los contenidos en los correspondientes buffers del MPLAB-IDE.

Erase

Borra todo el contenido actual de las memorias físicas del controlador.

Blank Check

Comprueba que las memorias del controlador estén borradas.

Read EEDATA

Lee el contenido de la memoria EEPROM de datos del controlador y lo deposita sobre el correspondiente buffer del MPLAB-IDE.

Connect

Restablece de nuevo la conexión con el grabador seleccionado, en nuestro caso el PICKit-2 (compatible con nuestro USB-PIC'SCHOOL). Presenta una serie de mensajes como los que se mostraron en la figura 2-6. Este comando puede ser necesario ejecutarlo en varias ocasiones: cuando se desconecta la alimentación, al cambiar de modelo de PIC y, en general, cuando se pierde la comunicación con el PC.

Download OS

Actualiza el firmware interno del PICKit-2 (USB-PIC'SCHOOL). Esta operación puede ser necesaria cuando se descargue una versión actualizada del propio MPLAB-IDE que nos avisa, al conectar con el PICKit-2 (USB-PIC'School), que el firmware interno de éste está desfasado.

Es una opción un tanto arriesgada que sólo hay que realizarla en condiciones óptimas de comunicación y alimentación. Un fallo de las mismas durante el proceso de actualización, puede conllevar el borrado del firmware interno y por consiguiente anular todas las posibilidades de Depuración/Grabación.

Release from Reset

Esta opción pone a nivel "1" la patilla MCLR del controlador.

Hold in Reset

Esta opción pone a nivel "0" la patilla MCLR del controlador.

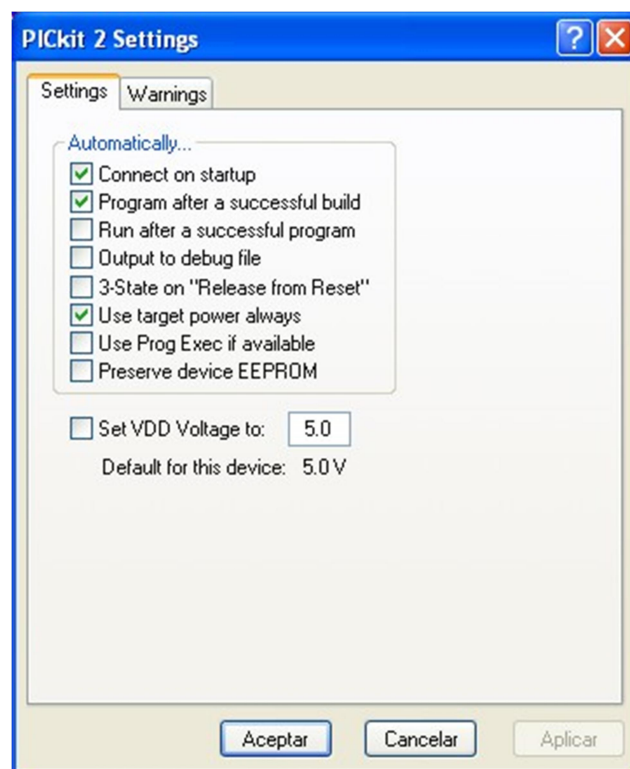
Settings

Con esta opción se abre una ventana similar a la mostrada en la figura 2-8 que permite activar o no una serie de procesos. En la pestaña "**Settings**" hemos activado los tres que consideramos más importantes:

- Conectar automáticamente al empezar. Cada vez que se selecciona el programador PICKit-2 (USB-PIC'SCHOOL) se procede automáticamente a la conexión con el mismo. En caso contrario hay que ejecutar manualmente el comando *Programmer* → *Connect*

- Programar automáticamente el controlador si no hay errores de ensamblado/compilado. Esta opción es muy interesante. Cada vez que ensamblamos nuestro programa fuente, si no hay errores, se procede automáticamente a grabar el dispositivo con el ejecutable recién obtenido.
- Usar siempre la alimentación del “target” o equipo bajo prueba que, en nuestro caso es el laboratorio USB-PIC'School.

Figura 2-8. Ajustes recomendados del PICKit-2 (USB-PIC'SCHOOL)



En la pestaña “Warnings” tenemos la posibilidad de activar o no los mensajes y avisos que deseamos que aparezcan si se producen determinados eventos.

2.3.6 Iconos de grabación

La mayor parte de las opciones del menú “Programmer” se pueden ejecutar mediante una serie de iconos de acceso directo como los mostrados en la figura 2-9.



Figura 2-9. Iconos de acceso directo para la grabación.

De izquierda a derecha su función se detalla seguidamente:

- *Program the target Device:* Programar el controlador
- *Read target Device memories:* Leer el contenido actual de las memorias del controlador
- *Read the target EEPROM memory:* Lectura de la memoria EEPROM de datos del controlador
- *Verify the contents of the target device:* Verificación del contenido del controlador
- *Erase the target device memories:* Borrar las memorias del controlador
- *Verify the target memories are erased:* Comprueba que el controlador esté borrado
- *Bring target MCLR to Vdd:* Pone MCLR del controlador a nivel “1”
- *Bring MCLR to VIL:* Pone MCLR del controlador a nivel “0”
- *Connection:* Restablece la conexión con el PICKit-2 (USB-PIC'SCHOOL)

2.4 EL SOFTWARE DE GRABACION PICKit-2

Se trata de otro programa de libre distribución, desarrollado por Microchip para su sistema PICKit-2 original que es compatible con nuestro grabador USB-PIC'SCHOOL. Se puede descargar y/o actualizar desde www.microchip.com.

A diferencia del MPLAB-IDE, este software está desarrollado para realizar, fundamentalmente, tareas de grabación y **NO** permite por tanto ni editar programas fuente, ni ensamblarlos, ni depurar, etc... Sin embargo admite la grabación de un mayor número de dispositivos PIC e incluye una serie de aplicaciones opcionales.

Insistimos en que nuestro grabador USB-PIC'SCHOOL es 100% compatible con el PICKit-2 en cuanto a las tareas de grabación se refiere, pero no incluye todas las funciones y posibilidades de éste. Por ello puede que algunas de esas aplicaciones opcionales que se incluyen en el software PICKit-2 no funcionen debidamente sobre el USB-PIC'SCHOOL.

IMPORTANTE Nos vamos a referir al software PICKit-2 exclusivamente en lo que a grabación respecta y que atañe a nuestro USB-PIC'SCHOOL. Dejaremos de lado el resto de funcionalidades diseñadas expresamente para el sistema PICKit-2 original y que se explican en su correspondiente manual incluido también en el CDROM que acompaña a nuestro USB-PIC'SCHOOL.

2.4.1 Instalación

El Grabador/Depurador USB-PIC'SCHOOL se conecta, por un lado, con uno de los puertos USB disponibles en el ordenador, mediante el correspondiente cable de interface incluido con el equipo.

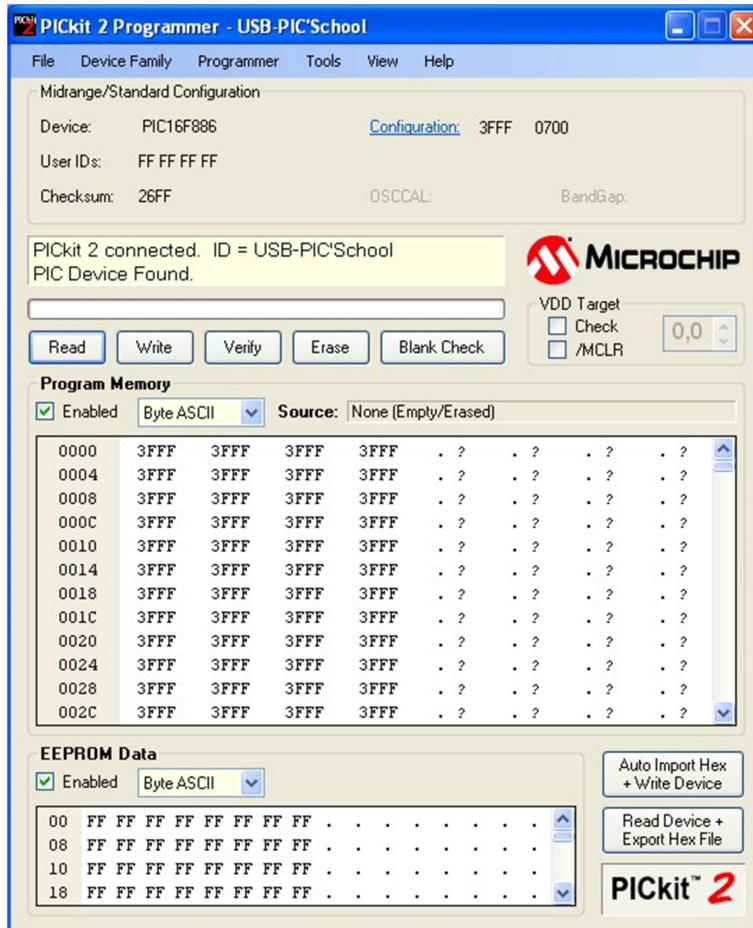
Inmediatamente es reconocido por Windows como un dispositivo de interface de usuario (HID) que se puede ver en *Panel de Control → Sistema → Hardware → Administrador de Dispositivos*.

El software de grabación PICKit-2 se incluye en el CDROM. Hay que descomprimirlo e iniciar la instalación. Esta no presenta ninguna dificultad. Basta con seguir las instrucciones que vayan apareciendo durante el proceso.

Su ejecución se puede realizar desde *Inicio → Programas → Microchip → PICKit 2* o bien desde el mismo icono de acceso directo que se crea en el escritorio tras la instalación. La figura 2-10 muestra el aspecto del programa.

Figura 2-10. El software PICKit-2

En ella se puede apreciar una ventana de estado en la que se nos informa que el PICKit-2 (USB-PIC'SCHOOL) ha sido localizado y conectado ("PICKit 2 connected. ID=US-PIC'School"). Si además hubiera algún PIC insertado en el laboratorio **USB-IC'School**, éste será detectado ("PIC Device Found"), identificado y seleccionado (p.e. PIC16F886). Ver la figura 2-12.



2.4.2 Selección de un dispositivo

Excepto si se selecciona la familia de gama baja ("*Baseline*"), que habrá que hacerlo siempre manualmente, en cualquiera de las otras familias se tratará de identificar el controlador PIC que en ese momento esté insertado en el laboratorio USB-PIC'School, de forma inmediata y automática.

También se puede hacer de forma manual. En cualquier momento podemos seleccionar entre las diferentes familias de PIC existentes.

Basta con activar la opción "*Manual Device Select*" del comando "*Programmer*" del menú principal.

A partir de este momento la opción "*Device Family*" del menú principal nos presenta un desplegable como en la figura 2-11, donde seleccionaremos la familia deseada.

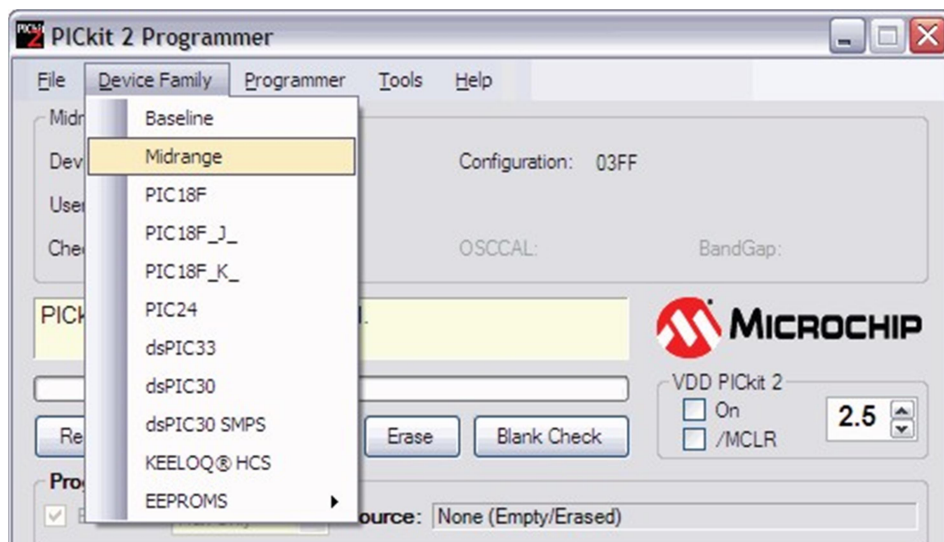
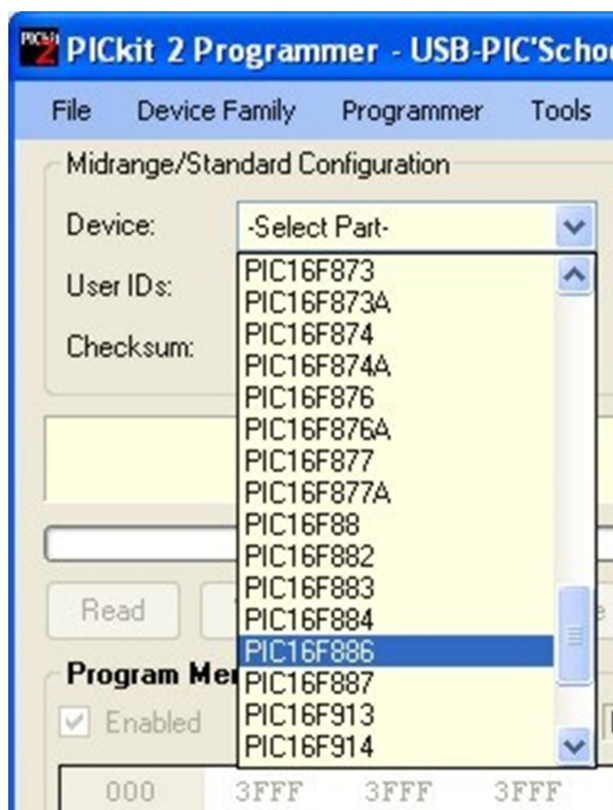


Figura 2-11. Selección de la familia de gama media PIC16



En el ejemplo se ha seleccionado la familia PIC16F ("*Midrange/Standard*") y en la lista "*Select Part*" se selecciona ahora un miembro o dispositivo en particular (p.e. el PIC16F886) como en la figura 2-12.

Se supone que estará insertado el controlador adecuado. Si no fuera así y, no hubiera ningún PIC insertado, o el insertado no es el mismo que el seleccionado, aparecerá el mensaje "*No Device Detected*".

Figura 2-12. Selección manual del PIC16F886 a grabar

Al igual que con los buffers de la memoria de programa y memoria de datos, el buffer de la memoria de configuración también se puede modificar manualmente.

Finalmente decir que también es posible crear y salvar un fichero HEX con el valor actual de los buffers de memoria del software PICKit-2, con todas las modificaciones realizadas sobre ellos. Basta con ejecutar *File → Export Hex* para que se nos abra la clásica caja de diálogo para seleccionar carpeta y nombre del fichero HEX que vamos a guardar.

2.4.4 Grabación del dispositivo

Una vez insertado el PIC, seleccionado la familia y modelo del mismo e importado el fichero HEX, podemos proceder a grabarlo. Las tareas básicas están disponibles en los botones de la figura 2-15.



Figura 2-15. Los botones con las tareas básicas

- Read:** Permite leer la memoria del PIC y el contenido lo copia sobre los buffers de memoria del software PICKit-2.
- Write:** Lee el contenido de los buffers de memoria del software PICKit-2 y lo graba sobre la memoria física del PIC.
- Verify:** Verifica que el contenido de los buffers sea el mismo que el contenido del PIC
- Erase:** Borra tanto los buffers de memoria del software PICKit-2 como la memoria física del PIC
- Blank Check** Comprueba que la memoria esté borrada.

Pulsamos el botón **Write** para proceder a grabar el controlador con el contenido de los buffers, previo borrado del mismo. En la ventana de estado tendremos constancia de si ha habido algún problema o bien el ciclo de grabación se realizó correctamente.

También es posible actuar únicamente sobre las regiones de memoria que deseemos dentro del PIC. Efectivamente, en la parte superior izquierda tanto del buffer de memoria de programa (*Program Memory*) como del buffer de la memoria de datos (*EEPROM Data*), disponemos de sendos "CheckBox" para habilitar (Enabled) o no el buffer correspondiente. Seleccionamos así sobre qué regiones deseamos o no actuar con las diferentes opciones y según la siguiente tabla:

CHECKBOX		OPCIONES	
Memoria de Programa	Memoria EEPROM de datos	Write/Read/Verify	Erase/Blank Check
Activado	Activado	Todas las regiones de memoria del PIC	Todas las regiones de memoria del PIC
Activado	Desactivado	Memoria de programa y memoria de configuración	Todas las regiones de memoria del PIC
Desactivado	Activado	Únicamente la memoria EEPROM de datos	Todas las regiones de memoria del PIC
Desactivado	Desactivado	No permitido	

2.4.5 Verificando el dispositivo

Esta función se ejecuta al pulsar el botón **Verify**. Consiste en comprobar que el contenido de las memorias o regiones del controlador coincide con el de los buffer de memoria del software PICKit-2: memoria de programa, memoria de datos en EEPROM y memoria de configuración e identificación.

Si se encuentra activada la opción *Programmer* → *Verify on Write*, todas las operaciones de escritura son verificadas automáticamente.

En la ventana de estado aparecerá el correspondiente mensaje que avisa si hay o no alguna diferencia en la comprobación.

2.4.6 Lectura del dispositivo

Esta función se ejecuta al pulsar el botón **Read**. Consiste en leer las diferentes regiones o áreas de memoria del controlador y copiarlas sobre los correspondientes buffers de memoria del software PICKit-2. Si la información leída es de todo ceros, es posible que el dispositivo estuviera protegido. En este caso los datos leídos no serán válidos.

2.4.7 Borrado del dispositivo

Esta función se realiza al pulsar el botón **Erase**. Provoca el borrado de todas las regiones o áreas de memoria del controlador empleando para ello el método "*Bulk Erase*" en aquellos controladores en que sea posible.

Normalmente no suele ser necesario el empleo de esta función ya que cada vez que se graba un dispositivo, implica su borrado automático.

2.4.8 Verificación de borrado

Esta función se ejecuta al pulsar el botón **Blank Check**. Realiza una lectura de todas las regiones de memoria del PIC para comprobar que se encuentran borradas.

2.4.9 Procedimientos automáticos de lectura/escritura

En la figura 2-16 se muestran los dos botones que posee el software PICKit-2 para realizar múltiples funciones cada vez que se actúa sobre cualquiera de ellos.



Figura 2-16. Botones para los procedimientos automáticos

Auto Import Hex + Write Device

Cada vez que se acciona aparece la clásica caja de diálogo de Windows gracias a la cual podemos localizar el fichero HEX deseado. Una vez seleccionado, automáticamente se procede a grabar el contenido del mismo sobre la memoria física del PIC.

Read Device + Export Hex File

Cada vez que se acciona se lee el contenido actual de las diferentes regiones de memoria del PIC y, automáticamente, se guarda en el fichero HEX indicado.

2.5 EL MODO DE DEPURACION

Como ya se ha venido explicando, el hardware del laboratorio USB-PIC'SCHOOL no es únicamente un grabador, es también un depurador en circuito, lo que lo convierte en una herramienta fundamental para el diseño y puesta a punto del programa de aplicación del usuario.

Básicamente la depuración en circuito consiste en poder controlar la ejecución de un programa grabado en el PIC y estando éste conectado a los periféricos de la aplicación. Para ello es necesario que el propio PIC disponga internamente de los recursos necesarios para la depuración.

NO todos los controladores PIC tienen capacidad de depuración, especialmente los más antiguos (p.e el conocido y popular PIC16F84). El fabricante proporciona una lista de los dispositivos que tienen o no capacidad de depuración.

Durante la depuración es necesario que el PIC reciba una serie de órdenes y comandos procedentes del software de desarrollo. De la misma manera, en el curso de la depuración, el PIC puede enviar al software de desarrollo información sobre el estado de sus registros internos, memoria RAM, etc.. Es por ello que algunas líneas de E/S del PIC **NO** pueden ser usadas por el hardware de la aplicación del usuario ya que se emplean para la comunicación con el PC. La depuración en circuito suele necesitar de 3 señales: MCLR, RB7/PGD, RB6/PGC. Por tanto estas señales no deben conectarse a ningún periférico o bien, mediante algún sistema de conmutación desconectarlas de esos periféricos durante el proceso de depuración. Es el caso que nos ocupa, empleamos el laboratorio **USB-PIC'School**. Cuando el conmutador se pone en la posición **USB**, esas señales se desconectan de los periféricos y quedan asociadas exclusivamente al hardware de depuración. Cuando se pone en la posición **RUN**, las señales se desconectan del depurador y se conectan con los periféricos en modo de ejecución.

Por los mismos motivos es posible que el sistema de depuración consuma algo de la memoria de programa, de datos, pila de stack, etc.. Así pues lo debemos tener en cuenta a la hora de desarrollar nuestros programas y aplicaciones respetar las zonas que estuvieran reservadas.

Incluso es posible que algunas prestaciones del depurador puedan verse modificadas en función del tipo de PIC con el que estemos trabajando. Así, hay modelos que admiten, por ejemplo, hasta 3 puntos de parada o breakpoints, y otros admiten únicamente uno.

Todos estos detalles están debidamente documentados en la información técnica que ofrece el fabricante e incluso en los diferentes tipos de ayuda que proporciona la plataforma de desarrollo MPLAB-IDE.

Sea como fuere, a pesar de las restricciones comentadas, en MK Electrónica consideramos que la depuración en circuito mediante nuestro laboratorio USB-PIC'SCHOOL (compatible con el PICKit-2) es una extraordinaria solución y una potente y económica herramienta que nos ayudará a depurar y poner a punto nuestros programas. Basta con decir que **TODOS** los ejemplos y ejercicios que proponemos en el CDROM que acompaña al equipo (más de 70), se han desarrollado y depurado en el propio laboratorio empleando su capacidad de depuración y puesta a punto. Únicamente es necesario tener en cuenta esas restricciones a la hora de desarrollar nuestros proyectos. Existen en el mercado otras soluciones como son el empleo de emuladores. Estos no tienen restricciones de ningún tipo pero su precio se eleva a varios cientos de euros.

Como software de desarrollo empleamos el MPLAB-IDE. Este permite el enlace entre el USB-PIC'SCHOOL (PICKit-2) y el PIC a depurar. Gracias a él tendremos la posibilidad de actuar internamente sobre el PIC, para ejecutar un programa en tiempo real, detener su ejecución, poner puntos de ruptura de forma que la ejecución se detenga al llegar a un punto en nuestro programa, ejecutar paso a paso una instrucción cada vez siguiendo así el flujo del programa, visualizar y modificar los registros internos, visualizar y modificar la memoria RAM interna, etc. Todo ello con un único fin: **localizar fallos en nuestro programa, corregirlos y ponerlo a punto en el mínimo tiempo posible y antes de proceder a grabarlo definitivamente sobre el PIC.**

En el presente documento consideramos que disponemos del laboratorio **USB-PIC'School** en el cual tenemos insertado el PIC16F886 que viene de serie. Asimismo recordamos que el conmutador de selección de modo **debe encontrarse en la posición USB.**

2.5.1 Seleccionando el dispositivo

Para comenzar a trabajar, lo primero que haremos será seleccionar el modelo de microcontrolador PIC con el que vamos a trabajar tal y como se hizo en el apartado 2.3.2 mediante *Configure → Select Device*. Este debe coincidir con el que en ese momento esté insertado en el laboratorio **USB-PIC'School** o en el hardware de la aplicación del usuario. En el ejemplo seleccionamos el PIC16F886.

2.5.2 Seleccionando el depurador

El software de desarrollo MPLAB-IDE admite diferentes tipos de herramientas hardware para realizar la emulación o depuración. En nuestro caso, tal y como muestra la figura 2-17 seleccionaremos el PICKit-2 (compatible con el USB-PIC'SCHOOL) mediante *Debugger → Select Tool → 7 PICKit 2*.

¡¡ ATENCION !!: MPLAB-IDE sólo admite al PICKit-2 (y por tanto al USB-PIC'SCHOOL) trabajando como emulador o como grabador, pero nunca ambos al mismo tiempo. Si estuviera activado como grabador, se debe desactivar mediante *Programmer → Select Programmer → None*

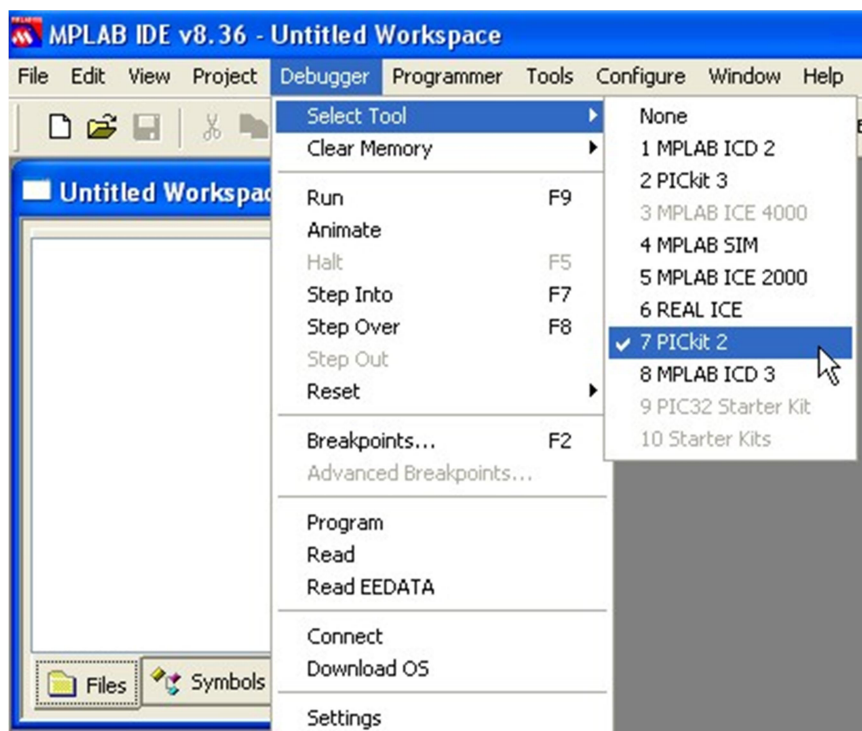
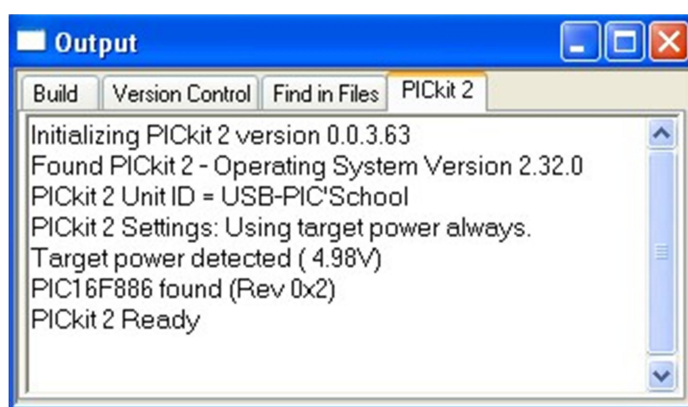


Figura 2-17. Selección del PICKit-2 (USB-PIC'SCHOOL) como herramienta de depuración



En la ventana Output de salida, como la mostrada en la figura 2-18, nos aparece una serie de mensajes que nos indican que se ha localizado e identificado tanto el PICKit-2 (USB-PIC'SCHOOL), como el microcontrolador sobre el que se hará la depuración.

Figura 2-18. Conexión del PICKit-2 (USB-PIC'SCHOOL)

¡ IMPORTANTE ! A partir de este instante no es conveniente cambiar el dispositivo seleccionado para la depuración (p.e. el PIC16F886) mediante *Configure → Select Device*, ya que se pueden provocar errores de funcionamiento. En caso de ser necesario, primero hay que desactivar el depurador mediante *Debugger → Select Tool → None*. A continuación se selecciona el nuevo dispositivo PIC. Este deberá insertarse en el laboratorio **USB-PIC'School**. Por último se vuelve a activar el PICKit-2 (USB-PIC'SCHOOL) mediante *Debugger → Select Tool → PICKit 2*.

2.5.3 Funciones más comunes en la depuración

Una vez se ha seleccionado el PICKit-2 (compatible con USB-PIC'SCHOOL) como herramienta de depuración, se activan todas las funciones disponibles relacionadas con la depuración y que se encuentran en el menú “*Debugger*” del menú principal, tal y como se muestra en la figura 2-19.

La primera de las opciones, “**Select Tool**”, es la que ya hemos empleado para seleccionar, de entre todas las disponibles, el PICKit-2 (USB-PIC'SCHOOL) como herramienta de depuración. A continuación resumiremos las opciones restantes.

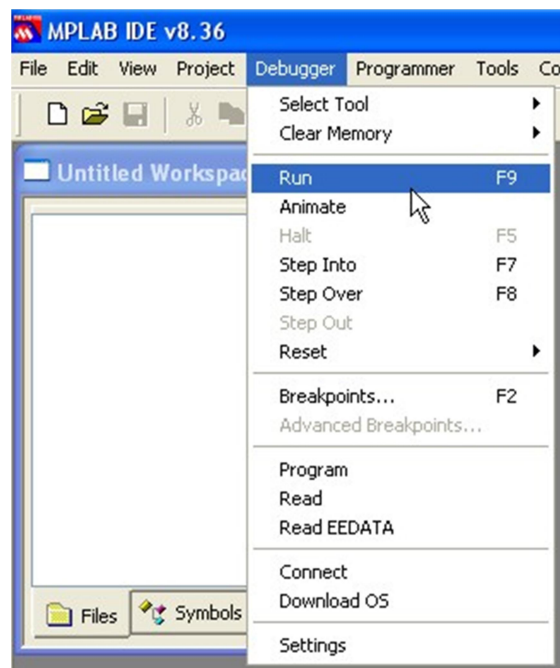
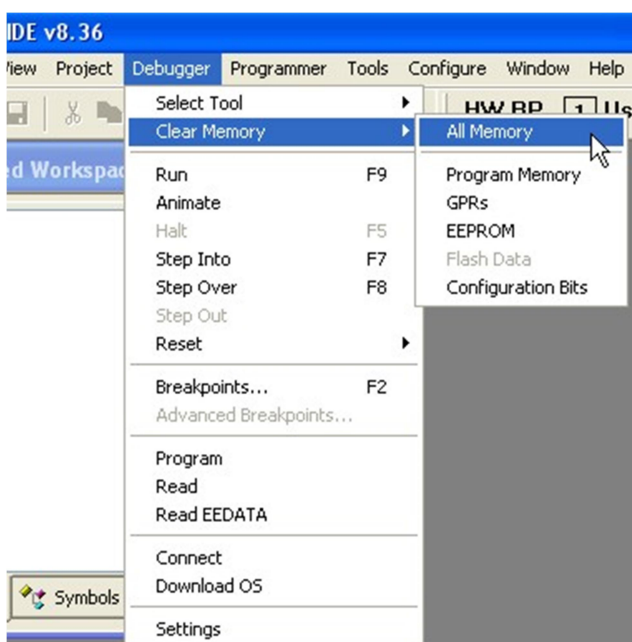


Figura 2-19. Funciones típicas en la depuración

Clear Memory



Run

Ejecuta en tiempo real las instrucciones que en ese momento se encuentren en la región o área de memoria de programa del PIC. Dichas instrucciones forman el programa ejecutable que se habrá obtenido en el último ensamblado realizado, o bien, como consecuencia de haber abierto un fichero HEX cuyo contenido se ha de grabar previamente en el dispositivo.

La ejecución en tiempo real significa que el programa se ejecuta de forma continua hasta que se detenga manualmente o se encuentre con un punto de parada o BreakPoint.

Borra las diferentes regiones o áreas de memoria que se emplean durante la depuración y que se pueden seleccionar mediante el menú desplegable como el mostrado en la figura 2-20. **All memory** borra todas las regiones, **Program Memory** borra sólo la memoria de programa, **GPRs** borra la memoria RAM de datos, **EEPROM** borra la memoria EEPROM de datos y **Configuration Bits** borra la palabra de configuración.

Estas regiones de memoria del MPLAB-IDE se pueden visualizar y/o modificar mediante el comando **View** del menú principal y se corresponden con las regiones o áreas de memoria física del PIC.

Figura 2-20. Borrado de las regiones de memoria

Animate

Ejecuta las instrucciones que en ese momento se encuentre en la región o área de memoria de programa del PIC. Dichas instrucciones forman el programa ejecutable que se habrá obtenido en el último ensamblado realizado, o bien, como consecuencia de haber abierto un fichero HEX cuyo contenido se ha de grabar previamente en el dispositivo.

En este tipo de ejecución el programa se ejecuta de forma continua hasta que se detenga manualmente o se encuentre con un punto de parada o BreakPoint. Sin embargo, con cada instrucción que se ejecuta, nuestro depurador USB-PIC'SCHOOL (PICKit-2) envía al MPLAB-IDE el valor actual de los registros internos del PIC, de forma que las ventanas de la memoria de programa, de datos, etc. del MPLAB-IDE se actualizan y refrescan constantemente con cada instrucción del programa ejecutada (Animación).

Realmente este modo de ejecución no tiene un gran interés ya que la misma NO es en tiempo real. Efectivamente, con cada instrucción ejecutada, el depurador envía al MPLAB-IDE una gran cantidad de información que éste ha de visualizar en las correspondientes ventanas, relentizando así el tiempo de ejecución.

Halt

Esta función únicamente queda habilitada si el sistema se encuentra en el modo **Run** o en el modo **Animate** anteriormente descritos. Su propósito es detener manualmente la ejecución del programa en el punto en que en ese momento se encuentre.

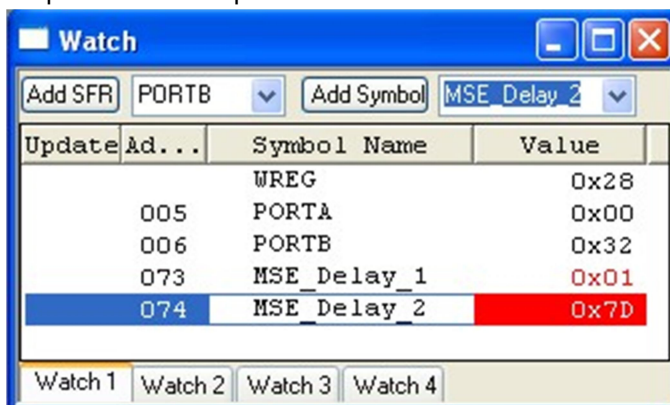
Step Into

Cada vez que se emplea este comando se procede a la ejecución de la siguiente instrucción del programa y el sistema se detiene. Es decir, se ejecuta una instrucción cada vez, paso a paso. Por cada instrucción ejecutada, el USB-PIC'SCHOOL envía información al MPLAB-IDE con la que se actualizan todos los registros de todas las áreas o regiones de memoria, con los valores internos actuales del PIC.

Esta es una de las funciones especialmente útiles en la fase de depuración de cualquier programa. Podemos ir ejecutando las instrucciones una a una, al tiempo que vamos viendo los resultados obtenidos en los registros de las diferentes áreas o regiones que visualiza el MPLAB-IDE y que se vean afectados por la instrucción recién ejecutada.

El MPLAB-IDE también permite modificar manualmente cualquiera de los registros de estas regiones. De esta forma podemos cargar cualquier registro o variable con cualquier valor y éste será tenido en cuenta y se evaluará al ejecutar la siguiente instrucción del programa.

Como cada vez que se ejecuta una instrucción se produce una importante transferencia de información entre el MPLAB-IDE y el depurador de USB-PIC'SCHOOL, la ejecución paso a paso puede verse relentizada por tener que refrescar las ventanas del MPLAB-IDE cada vez. Esto es inevitable y, además, no debe ejecutarse la siguiente instrucción hasta que no se finalice toda la transferencia de datos ya que puede originar un bloqueo del sistema. En alguna medida esta relentización se puede paliar abriendo el mínimo de ventanas y registros posibles en el MPLAB-IDE. De esta forma se consigue una mayor velocidad en el refresco y actualización de las mismas.



gura 2-21. La ventana Watch

En MK Electrónica sugerimos tener abierta únicamente la ventana **Watch** mediante el comando **View** → **Watch**. La figura 2-21 muestra un ejemplo de la misma.

Sobre esta ventana podemos indicar únicamente qué registros del PIC y qué variables RAM queremos refrescar para visualizar y/o modificar sus contenidos. En el ejemplo se refrescan los registros internos *WREG*, *PORTA* y *PORTB*, así como las variables del programa *MSE_Delay_1* y *MSE_Delay_2*.

Step Over

La tarea de depuración de un programa puede ser a veces muy tediosa ya que hay que analizar al 100% el comportamiento de ese programa. La ejecución paso a paso es la herramienta más versátil a la hora de localizar un fallo, pero también la que más tiempo nos puede llevar.

La mayor parte de las veces los programas contienen subrutinas que se deben ejecutar en múltiples ocasiones. Es posible que el funcionamiento de una subrutina lo tengamos que verificar paso a paso la primera vez, tal y como se explicó anteriormente, pero sería poco práctico que haya que verificarla tantas veces como el programa principal haga uso de la misma.

La función **Step Over** nos evita esta aburrida tarea. Efectivamente, esta función se comporta de forma similar a la función **Step Into** que vimos antes, ejecuta un programa paso a paso. La diferencia consiste en que si se trata de ejecutar la instrucción de llamada a subrutina (CALL), todas las instrucciones que ésta contiene se ejecutan en tiempo real. La parada se produce justo después del retorno de la misma.

Ahorraremos mucho tiempo de depuración si, una vez comprobado el correcto funcionamiento de una subrutina, la ejecutamos en tiempo real cada vez que sea llamada por el programa principal.

Step Out

Esta función sólo estará habilitada para aquellos dispositivos PIC y/o herramientas hardware que sean capaces de soportarla.

Si la ejecución paso a paso de un programa nos lleva a introducirnos en una subrutina, esta función permite realizar la ejecución de las instrucciones que la componen en tiempo real, desde el punto en el que en ese momento se encuentre hasta la instrucción de fin de subrutina (RETURN)

Reset

Esta función genera un RESET del sistema haciendo que el Contador de Programa (PC) del PIC apunte a la primera instrucción del programa a ejecutar, justo tras el vector de RESET.

El efecto que produce su empleo se puede apreciar claramente si tenemos abierta la ventana de nuestro programa fuente (ASM o C) debidamente ensamblado/compilado y que se carga mediante **File** → **Open**. También se puede apreciar sobre la ventana de memoria de programa que se abre mediante **View** → **Program Memory**. La figura 2-22 muestra ambas ventanas.

En la parte superior vemos la ventana que nos muestra un fragmento del programa fuente en ASM. En la parte inferior vemos la ventana de la memoria de programa con el mismo fragmento pero debidamente ensamblado y convertido a código ejecutable.

En ambos casos, a la izquierda, se aprecia una flecha verde que apunta a la instrucción que toca ejecutar justo tras haber producido el RESET y que se corresponde con la siguiente instrucción a la del vector RESET.

Esa flecha simboliza al Contador de Programa (PC) y apunta a la instrucción que toca ejecutar. Si se ejecuta en tiempo Real (**Run**), la ejecución se inicia a partir de esa posición. Si se detiene la ejecución (**Halt**) la flecha aparecerá justo en la siguiente instrucción donde se produjo la detención y que sería la siguiente en caso de reanudación.

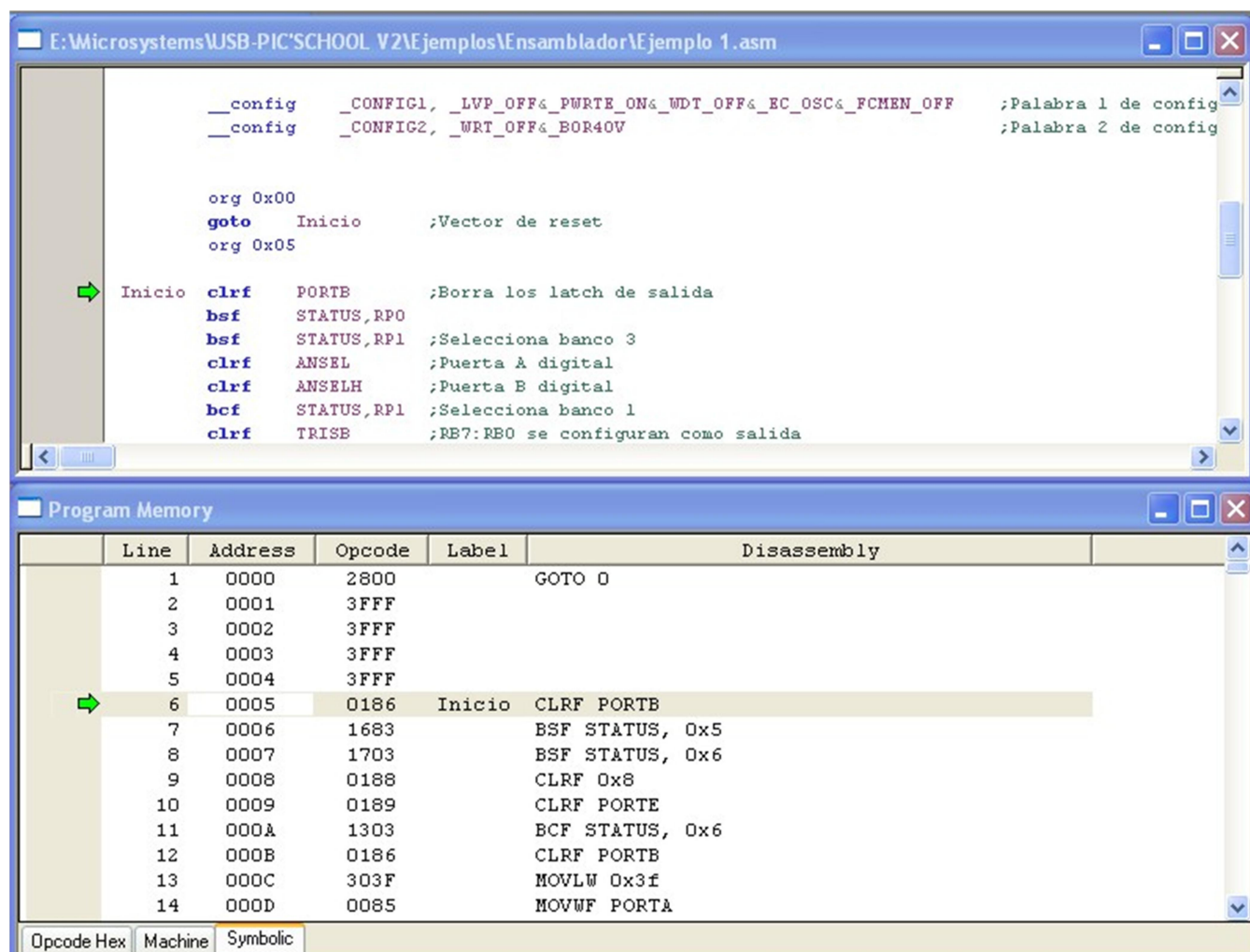


Figura 2-22. El PC del PIC tras el RESET

Si se ejecuta en modo animado (**Animate**) veremos claramente cómo la flecha va evolucionando instrucción tras instrucción a lo largo del programa hasta que éste sea detenido.

En la ejecución paso a paso veremos que la flecha avanza a la siguiente instrucción a ejecutar, cada vez que se emplea **Step Into**.

Breakpoints

Los puntos de parada o Breakpoints constituyen otra importante herramienta para la depuración y puesta en marcha de los programas de aplicación. Consiste en establecer uno o varios puntos en los que debe detenerse la ejecución en tiempo real del programa al llegar a ellos, momento en el que se actualizan todos los registros y regiones de memoria sobre las correspondientes ventanas del MPLAB-IDE. Esto permite comprobar, de forma rápida y eficaz, cómo funciona y reacciona un determinado bloque de instrucciones del programa de aplicación.

Para establecer un punto de parada, basta con hacer doble click sobre la instrucción deseada. Podemos emplear para ello tanto la ventana con el programa fuente (ASM o C) como la ventana del área de programa.

En la figura 2-23 vemos ambas ventanas. Se hizo un doble click sobre la instrucción "**movwf TRISA**". Un círculo rojo a la izquierda de la instrucción nos recuerda que hay un punto de parada (se emplea el mismo método para eliminarlo).

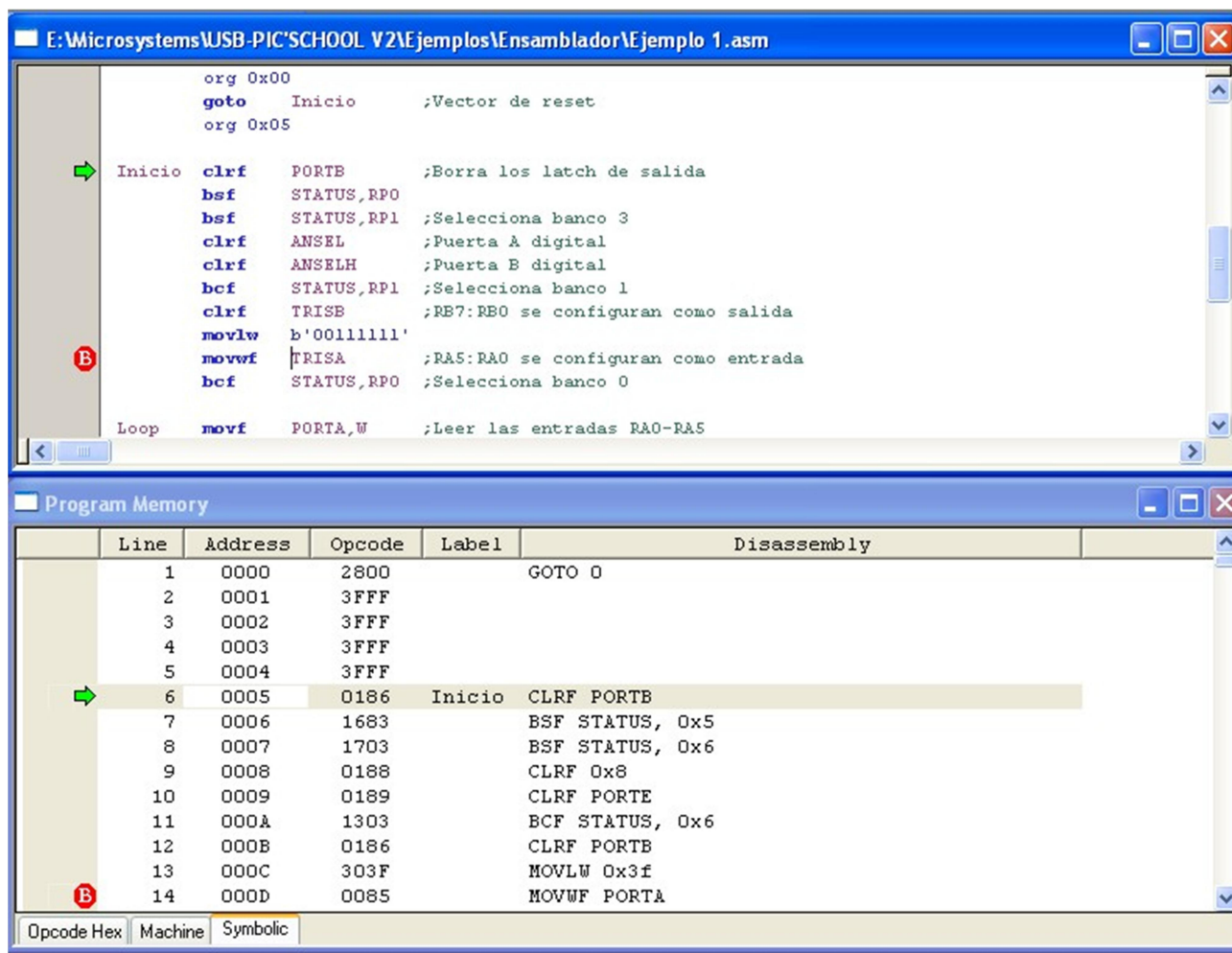
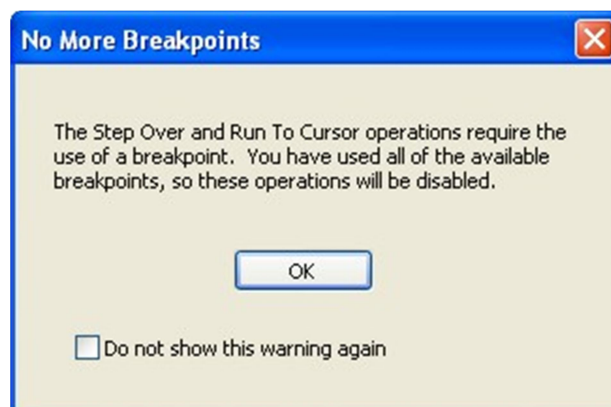


Figura 2-23. Estableciendo un punto de parada o breakpoint

A la vista de esa figura se puede determinar que el inicio de la ejecución se realizará a partir de la instrucción “**clrf PORTB**” (flecha verde) y se detendrá después de la instrucción “**movwf TRISA**” (círculo rojo). Ambas ventanas son equivalentes y reflejo la una de la otra.

Según el modelo de PIC que estemos empleando es posible que aparezca un mensaje como el mostrado en la figura 2-24. Con él se nos recuerda que el USB-PIC'SCHOOL (PICKit-2) depurando sobre un PIC16F886 sólo dispone de un BreakPoint, por lo que algunas otras operaciones pueden quedar deshabilitadas. Estas son algunas de las limitaciones del PICKit-2 (y por tanto del USB-PIC'SCHOOL) de las que se habló al principio de este apartado.

Figura 2-24. Aviso sobre el empleo de los breakpoints



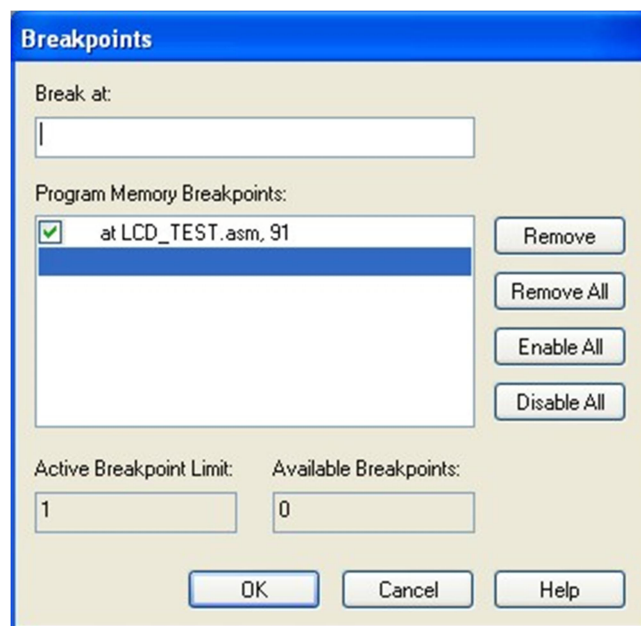
Ahora basta con ejecutar el programa en tiempo real (**Run**). Todas las instrucciones comprendidas entre la posición actual del PC (flecha verde) y el círculo rojo (breakpoint) se ejecutarán en tiempo real, a la velocidad propia del controlador.

La ejecución se detiene justo después del punto de parada. En ese momento las ventanas del MPLAB-IDE se actualizan con el valor de los registros internos y variables del programa del PIC tras haberse ejecutado ése trozo de programa.

Ya hemos dicho que establecer un punto de parada es tan sencillo como hacer doble click sobre la instrucción del programa en la que deseamos detener la ejecución. El mismo proceso se emplea para eliminarlo. No obstante mediante el uso del comando *Debugger* → *Breakpoints* del menú principal obtenemos una ventana de edición como la mostrada en la figura 2-25. En ella podemos visualizar el lugar en que se encuentra cada punto de parada.

También es posible activarlos, desactivarlos y borrarlos tanto de forma individual como colectiva. Finalmente también nos informa cuántos Breakpoint se pueden colocar y cuántos quedan disponibles.

Figura 2-25. Control de los Breakpoints



Advanced Breakpoints

Esta opción está destinada a las herramientas hardware de alto nivel para la emulación en tiempo real como son el ICE2000 y el ICE4000. No funciona por tanto con el depurador de USB-PIC'SCHOOL; está deshabilitada.

Los Breakpoints avanzados permiten establecer muchas y complejas condiciones de parada durante la emulación/depuración del programa de aplicación del usuario. Puntos de parada no sólo basados en la dirección de una instrucción como se vio anteriormente.

Permite establecer condiciones tales como *“Cuando el PC se encuentre en la dirección X, se esté realizando un ciclo de escritura sobre la posición Y de memoria RAM, el valor del registro Z sea igual a 0x033 y esta condición se repita 18 veces”*, entonces se produce la parada y la ejecución se detiene.

Programm

Esta opción permite grabar directamente el contenido de las distintas áreas de memoria del MPLAB-IDE sobre las correspondientes regiones de la memoria física del PIC. Es especialmente útil para grabar la información directamente importada desde un fichero HEX.

Efectivamente, con esta opción no es necesario abrir un programa fuente (ASM o C) ni ensamblarlo o compilarlo. Basta con abrir un fichero HEX mediante **File** → **Import**, que contenga la información a grabar.

Read

Es la opuesta a la anterior. Lee el contenido de las diferentes regiones de memoria del PIC y lo deposita sobre las correspondientes áreas de memoria del MPLA IDE.

Mediante **File** → **Export** podemos guardar la información recién leída sobre un fichero HEX.

Read EEDATA

Procede a la lectura de la región de memoria EEPROM del PIC y la deposita sobre la correspondiente área o buffer del MPLAB. El contenido recién leído se puede visualizar activando la correspondiente ventana mediante la opción **View → EEPROM** del menú principal.

Connect

Restablece de nuevo la conexión con el depurador seleccionado, en nuestro caso el PICKit-2 (compatible con el USB-PIC'SCHOOL). Esta operación puede ser necesario realizarla en varias ocasiones ya que a veces el sistema se puede colgar durante la transferencia de información hacia/desde el PC y el PIC en depuración.

Download OS

Actualiza el firmware interno del PICKit-2 (USB-PIC'SCHOOL). Esta operación puede ser necesaria cuando se descargue una versión actualizada del propio MPLAB-IDE.

Es una opción un tanto arriesgada que sólo hay que realizarla en condiciones óptimas de comunicación y alimentación. Un fallo de las mismas durante el proceso de actualización, puede conllevar el borrado del firmware interno y por consiguiente anular todas las posibilidades de Depuración/Grabación.

Settings

Con esta opción se abre una ventana similar a la mostrada en la figura 2-26 y que permite activar o no una serie de procesos internos. En la pestaña "**Settings**" hemos elegido los tres que consideramos más importantes:

- Conectar automáticamente al empezar. Cada vez que se selecciona el depurador PICKit-2 (USB-PIC'SCHOOL) se procede automáticamente a la conexión con el mismo. En caso contrario hay que ejecutar manualmente el comando **Debugger → Connect**
- Programar automáticamente el controlador si no hay errores de ensamblado/compilado. Esta opción es muy interesante. Cada vez que ensamblamos nuestro programa fuente, si no hay errores, se procede a grabar el dispositivo con el programa ejecutable recién obtenido así como ciertas tareas internas, quedando así listo para la depuración. Si no se activa esta opción, la grabación habrá que hacerla manualmente mediante el comando **Debugger → Programm**.
- Usar siempre la alimentación del "target" o equipo bajo prueba que, en nuestro caso es el laboratorio USB-PIC'School.

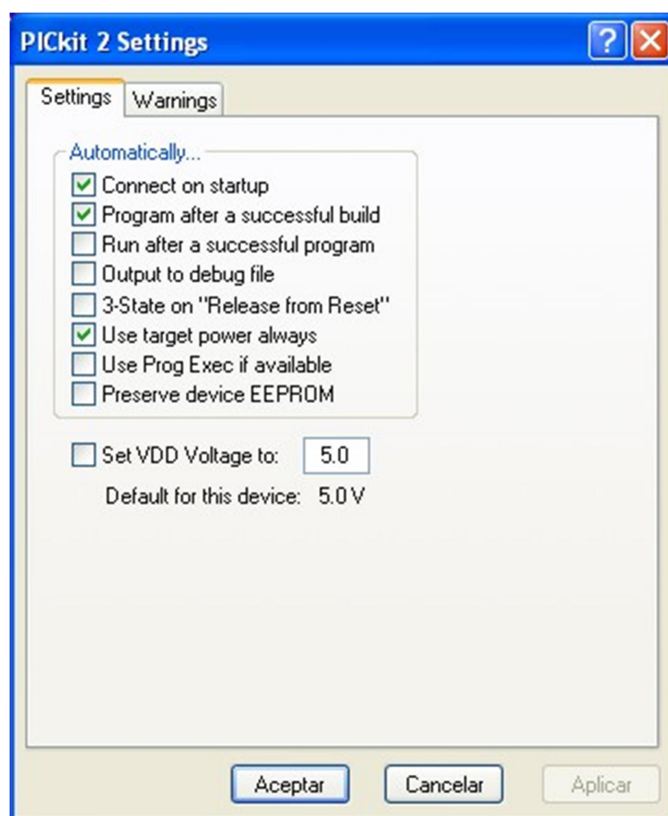


Figura 2-26. Ajustes del entorno de depuración

En la pestaña “**Warnings**” tenemos la posibilidad de activar o no los mensajes y avisos que deseamos que aparezcan si se producen determinados eventos.

2.5.4 Iconos empleados en la depuración

La mayor parte de las opciones del menú “Debugger” se pueden ejecutar mediante una serie de iconos de acceso directo como los mostrados en la figura 2-27 y que ya se han explicado anteriormente.



Figura 2-27. Iconos de acceso directo para la depuración

De izquierda a derecha su función se detalla a continuación:

- *Run*: Ejecución en tiempo real del programa
- *Halt*: Detener la ejecución
- *Animate*: Ejecución con animación
- *Step Into*: Ejecución paso a paso
- *Step Over*: Ejecución paso a paso y las subrutinas en tiempo real
- *Step Out*: Salir de una subrutina ejecutando en tiempo real
- *Reset*: Reinicio del controlador
- *Breakpoints*: Visualiza estado de los puntos de parada
- *Program the target Device*: Programar el controlador
- *Read target Device memories*: Leer el contenido del controlador
- *Read the target EEDATA memory*: Lee el contenido de la memoria EEPROM del controlador
- *Connection*: Re establecer la conexión entre el MPLAB-IDE y el PICKit-2 (USB-PIC'SCHOOL)
- *Advanced Breakpoint*: Acceso a los breakpoints avanzados (no disponible en PICKit-2/USB-PIC'SCHOOL)



TEMA 3: Tutorial y Ejemplos

3.1 INTRODUCCION

En este tema vamos a tratar de describir las tareas y herramientas necesarias para desarrollar el programa de control de una determinada aplicación. También vamos a proponer una serie de ejemplos básicos con programas de carácter didáctico que se grabarán y ejecutarán sobre el dispositivo PIC 16F886 que incorpora de serie el laboratorio USB-PIC'School. El organigrama mostrado en la figura 3-1 trata de resumir los pasos necesarios.

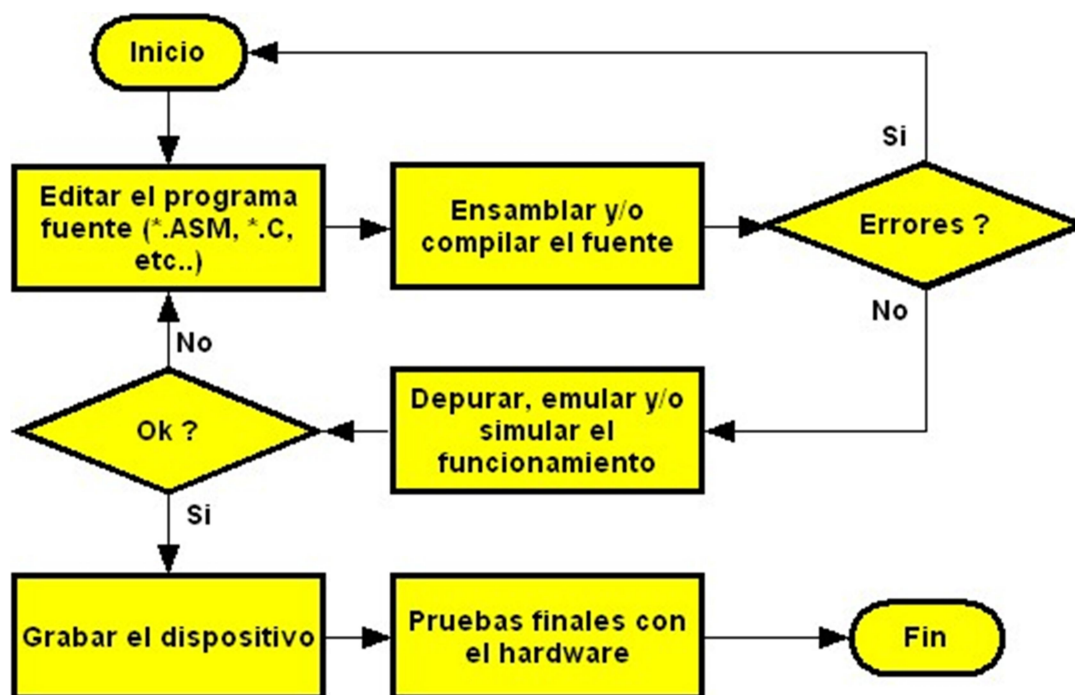


Figura 3-1. Esquema básico de trabajo

- 1º El primer paso para el diseño del software de una aplicación es editar el programa fuente. Para ello se puede emplear cualquier editor de textos típico y que sea capaz de guardar ficheros de texto tipo ASCII como pueden ser el WordPad o el Bloc de notas incluidos en Windows, el Microsoft Word, etc.. El entorno de desarrollo MPLAB-IDE que proponemos, integra un completo editor de textos sin necesidad de tener que recurrir a ningún otro.

Se escribe el programa fuente siguiendo la sintaxis propia del lenguaje de programación que estemos empleando: Ensamblador, C, BASIC, etc.. La extensión del archivo que contiene el programa fuente suele indicar el tipo de lenguaje para el que está escrito (*.ASM, *.C, *.BAS, etc..).

- 2º Ensamblar y/o compilar consiste en traducir el programa fuente escrito mediante el editor de textos, en su equivalente en código máquina o hexadecimal. Para ello existen diferentes herramientas según el tipo de lenguaje empleado. Las herramientas para traducir código ensamblador suelen ser gratuitas y están disponibles en la web del fabricante. En el caso de Microchip el propio MPLAB-IDE integra el ensamblador.

Las herramientas compiladoras para C, BASIC o cualquier otro tipo de lenguaje de alto nivel, suelen tener un determinado coste. De todas formas es posible encontrar en la red versiones "shareware" de las mismas, versiones "demo" e incluso versiones para su evaluación.

- 3º En el proceso de traducción se pueden producir errores. Estos errores son siempre de tipo sintáctico. Es decir, la herramienta empleada para la traducción ha encontrado alguna incongruencia en el programa fuente, instrucciones desconocidas, parámetros incorrectos, etc. Es algo así como si se detectaran faltas de ortografía. En este caso el proceso se detiene y se debe volver al editor del programa fuente (punto 1º) para corregir los errores encontrados.

Cuando un programa fuente está correctamente escrito, los ensambladores y/o compiladores generan una serie de archivos. Uno de ellos, el más importante, contiene el resultado de la traducción, es decir el código máquina o hexadecimal. La extensión de este archivo es *.HEX. El resto de archivos suelen ser archivos auxiliares y contienen información de la traducción, listados, información para simulación, depuración, emulación, etc.. Suelen ser necesarios y empleados por las herramientas hardware/software para la depuración, emulación y/o simulación.

- 4º Obtenido el fichero ejecutable *.HEX debemos ahora verificar su funcionamiento. Para ello todos los fabricantes disponen de una serie de herramientas hardware/software que permiten comprobar el correcto funcionamiento del programa de aplicación recién obtenido y, en su defecto, facilitar la localización del/los fallo/s.

La herramienta más profesional pero de mayor coste es la conocida como “Emulador”. Está formado por un hardware y un software especial que permiten un análisis en tiempo real del resultado de la ejecución. El emulador se conecta físicamente en el hardware de la aplicación del diseño sustituyendo a todos los efectos y, sin ningún tipo de restricción, al microcontrolador que en dicho hardware debiera estar insertado. Mediante el software de emulación adecuado el emulador ejecuta el programa de aplicación como si del propio microcontrolador se tratara, controlando en tiempo real a todos los periféricos existentes. Es posible detener la ejecución en un momento determinado (breakpoint), ejecutar paso a paso todas las instrucciones, visualizar/modificar los registros internos, analizar tiempos de ejecución, registrar sucesos o eventos externos y un largo etcétera.

En el caso de los PIC's, Microchip dispone de sendos emuladores como el IC2000 y el IC4000 cuyo control se realiza mediante el conocido entorno integrado MPLAB-IDE. También existen varios fabricantes o terceras partes que comercializan este tipo de herramientas para estos microcontroladores.

En un término medio y a un coste bastante asequible nos encontramos con los “Depuradores en Circuito” (ICD) como el que integra el laboratorio USB-PIC'School (compatible con el PICKit-2). Algunos microcontroladores, como es el caso de algunos dispositivos PIC, incorporan en su interior mecanismos de depuración integrados. Basta un sencillo hardware externo llamado ICD para comunicar dichos mecanismos con el software de depuración. Gracias al sistema ICD el programa de aplicación del usuario se graba físicamente sobre el propio microcontrolador al que se le supone insertado en el hardware final (p.e. el USB-PIC'School). Este programa se ejecuta por tanto en tiempo real gestionando así todos los periféricos que hubiera conectados. El ICD también permite el empleo de puntos de parada (breakpoints), ejecución paso a paso, visualización/modificación de registros, etc..

Puesto que el ICD es una herramienta de bajo coste, también tiene una serie de limitaciones o restricciones. Dado que se emplean algunos recursos del propio microcontrolador, estos no están disponibles para el usuario. Consume parte del área de programa, algunas líneas de E/S y, cada vez que se modifique el programa de aplicación, es necesario volver a re grabarlo sobre el microcontrolador con el desgaste que esto supone. En ningún caso se puede por tanto comparar un “Depurador” con un “Emulador”. De estas restricciones ya hablamos en el apartado de Depuración del tema anterior. Decir que el entorno MPLAB-IDE también da soporte a la depuración mediante los sistemas ICDx y el PICKit-2 (compatible con el depurador del laboratorio USB-PIC'School).

Por último nos encontramos con las herramientas de simulación. Es la solución menos profesional pero la mayor parte de fabricantes la distribuyen de forma gratuita a sus usuarios. Se trata de un software que se ejecuta en el PC y trata de “simular” el comportamiento del

microcontrolador cuando éste ejecuta el programa de aplicación. Naturalmente esta ejecución NO se realiza en tiempo real ni tampoco podemos “conectar” periféricos reales al PC.

El ya mencionado entorno de desarrollo MPLAB-IDE de microchip también tiene la posibilidad de hacer simulación de los diferentes dispositivos PIC existentes. Es una solución bastante válida a nivel didáctico en ciclos formativos y, sobre todo, **es gratuita**.

- 5º Sea cual sea la herramienta de “emulación/depuración/simulación” seleccionada, el objetivo final es siempre localizar posibles fallos en el programa de aplicación y corregirlos. La corrección de esos fallos implica volver al punto 1º para editar y corregir el programa fuente y ensamblarlo nuevamente. El proceso finaliza cuando dicho programa cumpla todas las expectativas.
- 6º Grabar el dispositivo. Efectivamente, una vez que se ha obtenido un programa funcional y ejecutable (*.HEX) se procede a grabarlo en la memoria interna del microcontrolador. Para ello se recurre a un grabador dotado del software de grabación oportuno.

Por supuesto que el entorno de desarrollo MPLAB-IDE también tiene la posibilidad de grabar dispositivos. Concretamente permite la conexión con varios equipos de grabación diseñados y comercializados por Microchip: PICSTART Plus, MPLAB ICDx, MPLAB PM3, PROMATE II y PICKit-2, que es compatible con el que integra el laboratorio USB-PIC'School y del que ya se habló en el apartado dedicado a la grabación del tema anterior.

- 7º El último paso consiste en hacer las pruebas finales y definitivas. El microcontrolador recién grabado se inserta en el hardware final de la aplicación donde se verifica el correcto funcionamiento de todo el conjunto hardware/software.

El laboratorio USB-PIC'School es una potente y autónoma herramienta hardware para el desarrollo de aplicaciones basadas en microcontroladores PIC. Admite los dispositivos más representativos, posee un buen número de periféricos con un flexible sistema para su interconexión, dispone de un área para el montaje rápido sin soldaduras, es capaz de Depurar/Grabar el dispositivo PIC insertado en su correspondiente zócalo y todas las tareas se pueden realizar desde el entorno MPLAB-IDE.

3.2 TUTORIAL

En MK Electrónica hemos optado por el empleo del software MPLAB-IDE para el desarrollo del presente tutorial y sus ejemplos, por ser un Entorno Integrado de Desarrollo (IDE) muy completo:

- Es totalmente gratuito y desde www.microchip.com se pueden obtener las diferentes actualizaciones.
- Permite realizar todas las tareas necesarias en un proyecto: editar el programa fuente, ensamblar y/o compilar (si se dispone del compilador) el programa fuente, simular el funcionamiento del programa ejecutable, etc..
- Si se dispone de las herramientas hardware correspondientes (p.e. USB-PIC'School) también es posible depurar y grabar el dispositivo.

Se supone que el programa está debidamente instalado siguiendo las instrucciones de instalación. También se sugiere crear una carpeta llamada “Ejemplos para USB-PIC'School” y copiar en ella todos los programas fuente *.ASM de ejemplo contenidos en el CD-ROM que viene con el equipo.

3.2.1. Creación de un proyecto

Cuando se está en fase de desarrollo de un programa de aplicación, se genera una abundante información relacionada con el mismo que se aglutina en torno a lo que se llama un “Proyecto”. Un proyecto no es más que un fichero en el que queda reflejado una importante serie de datos relacionados con esa aplicación:

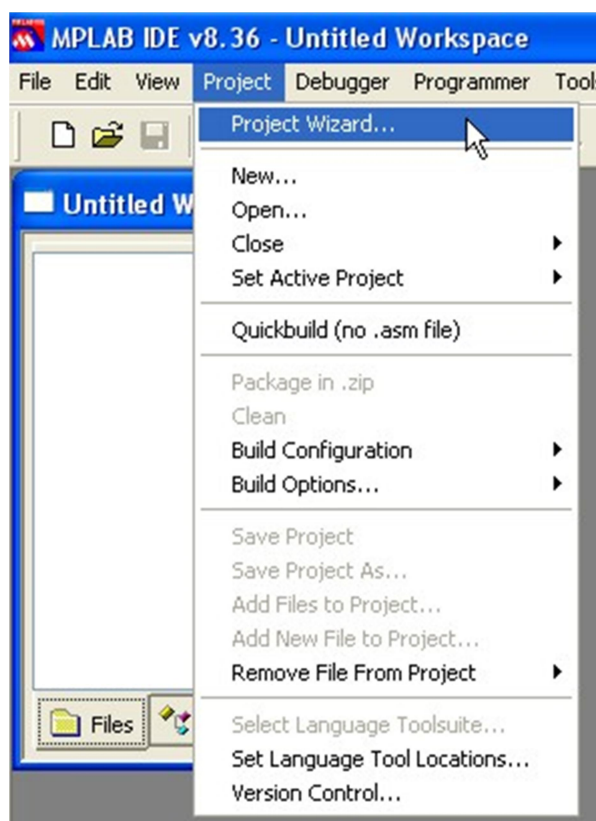
- Tipo de procesador empleado.
- Lenguaje de programación.
- Nombre y ruta del fichero que contiene el programa fuente.

- Ficheros que se obtienen tras el ensamblado y/o compilación: ejecutables (*.HEX), listados (*.LST), errores (*.ERR), información de simulación, depuración, emulación, etc..
- Entorno de trabajo
- Parámetros de configuración, etc...

De esta forma, cuando se retoma el trabajo de un día para otro, basta con abrir simplemente el proyecto en cuestión. Automáticamente y de forma rápida y cómoda aparece el entorno de trabajo tal y como se dejó.

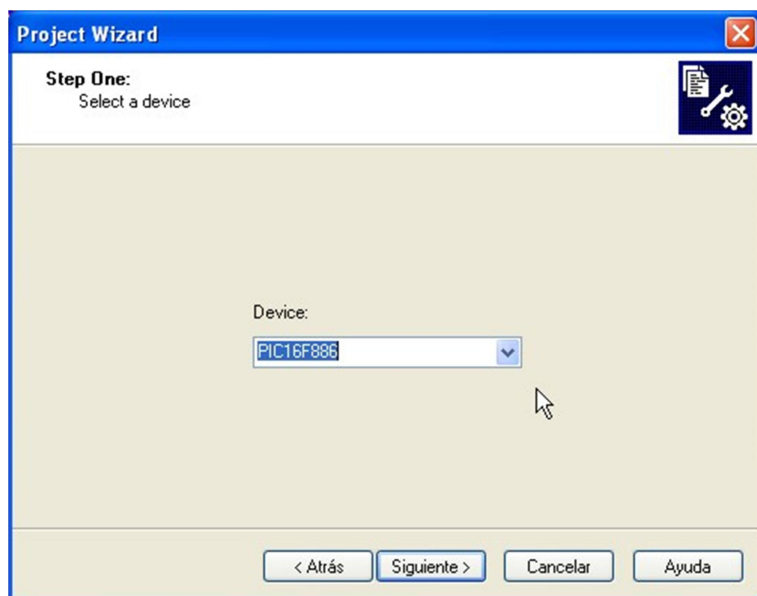
El entorno de desarrollo MPLAB-IDE dispone de un cómodo gestor de proyectos que, tal y como se muestra en la figura 3-2, se inicia mediante la opción “Project/Project Wizard” del menú principal.

Figura 3-2. Iniciando el gestor de proyectos



Este gestor nos va guiando a definir el proyecto y se puede resumir en los siguientes pasos:

- 1º Definir el tipo de microcontrolador que se va a emplear en la aplicación. Ver la figura 3-3.



Aparece una lista desplegable con todos los dispositivos admitidos en la versión actual de MPLAB-IDE. Estamos empleando la versión 8.36 y el dispositivo seleccionado es el PIC16F886 que es el que incorpora de serie el laboratorio USB-PIC'School.

Se pulsa el botón “Siguiente” para seguir con el proceso.

Figura 3-3. Selección del dispositivo

- 2º Se selecciona la herramienta software que se va a emplear en el proceso de ensamblado y/o compilación para obtener el programa ejecutable (*.HEX) a partir del programa fuente (*.ASM, *.C, etc.). Ver la figura 3-4.

Mediante la persiana denominada "Active Toolsuite" se selecciona una de esas herramientas.

En la actualidad, MPLAB-IDE se puede integrar con herramientas propias de Microchip o de terceras fuentes. Para ello se deben seguir las instrucciones de instalación propias de cada una de esas fuentes. En el caso que nos ocupa seleccionamos "Microchip MPASM Toolsuite" que es el ensamblador que se suministra gratuitamente y se instala junto con el propio MPLAB-IDE. Esta herramienta consta del ensamblador (mpasmwin.exe) propiamente dicho, el enlazador o lincador (mplink.exe) y las librerías (mplib.exe). Con el botón "Browse..." se puede seleccionar el lugar o ruta donde se encuentran cada uno de estos ficheros, que por defecto es la mostrada en la figura.

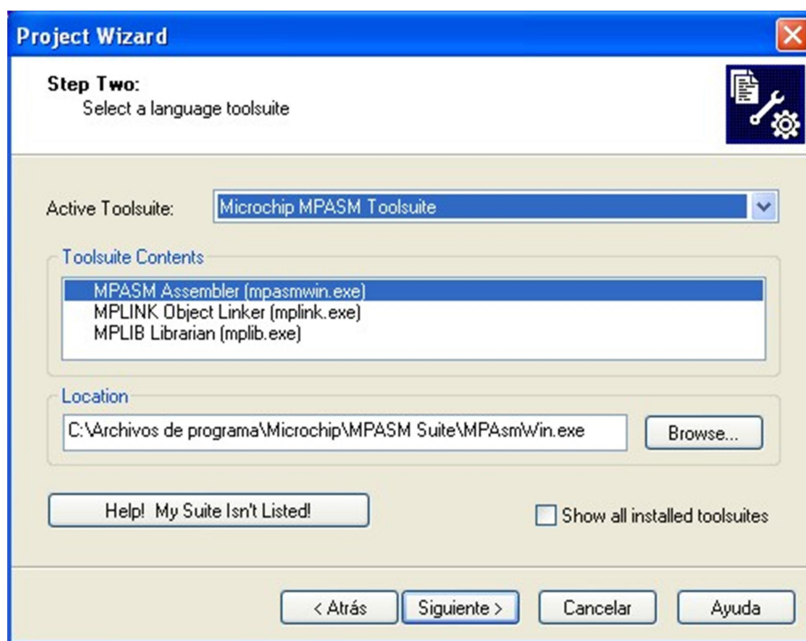
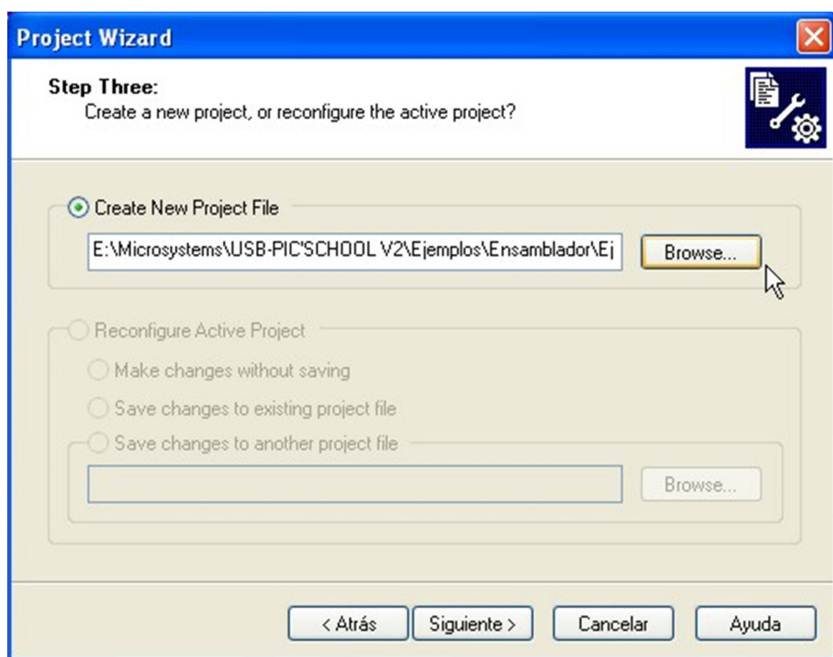


Figura 3-4. Seleccionando la herramienta de ensablado/compilación

3º Dar nombre al proyecto e indicar la carpeta donde ha de guardarse, tal y como se muestra en la figura 3-5.



En este caso el proyecto se denomina "Ejemplo 1" y se guardará en la carpeta que seleccionemos al pulsar *Browse*

Esta carpeta es la que se sugirió crear anteriormente para guardar en ella todos los programas fuentes de ejemplo (*.ASM) que se suministran en el CD-ROM.

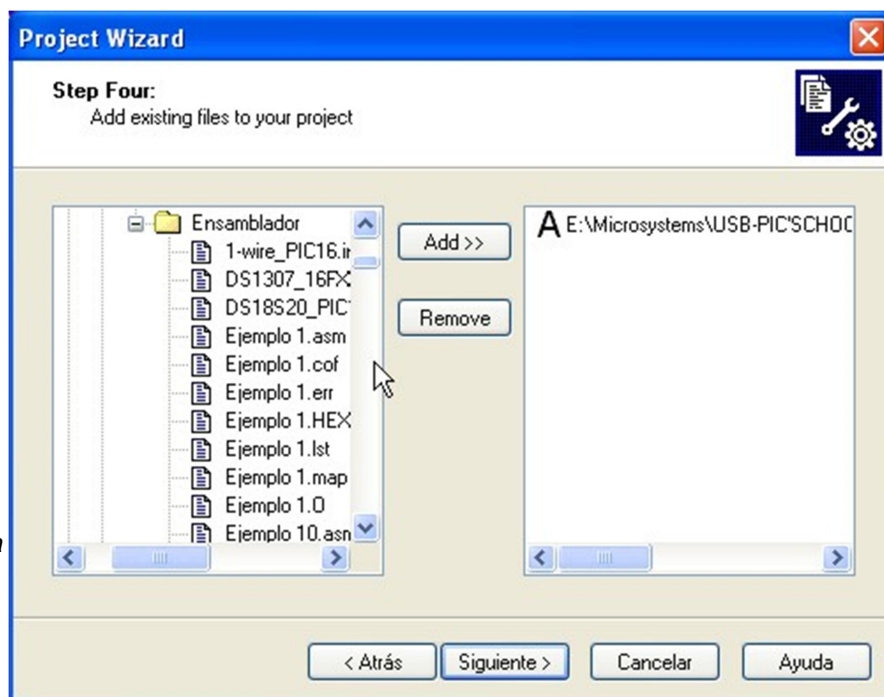
Figura 3-5. Asignando nombre al proyecto

4º Asignar el programa fuente asociado al proyecto. Ver la figura 3-6. En el caso que nos ocupa los programas de ejemplo se supone están editados y almacenados en una carpeta creada por el usuario y procedentes del CD-ROM. De esta manera evitamos tener que teclearlos. En otros casos habría que editarlos mediante el editor de textos que dispone el propio MPLAB-IDE.

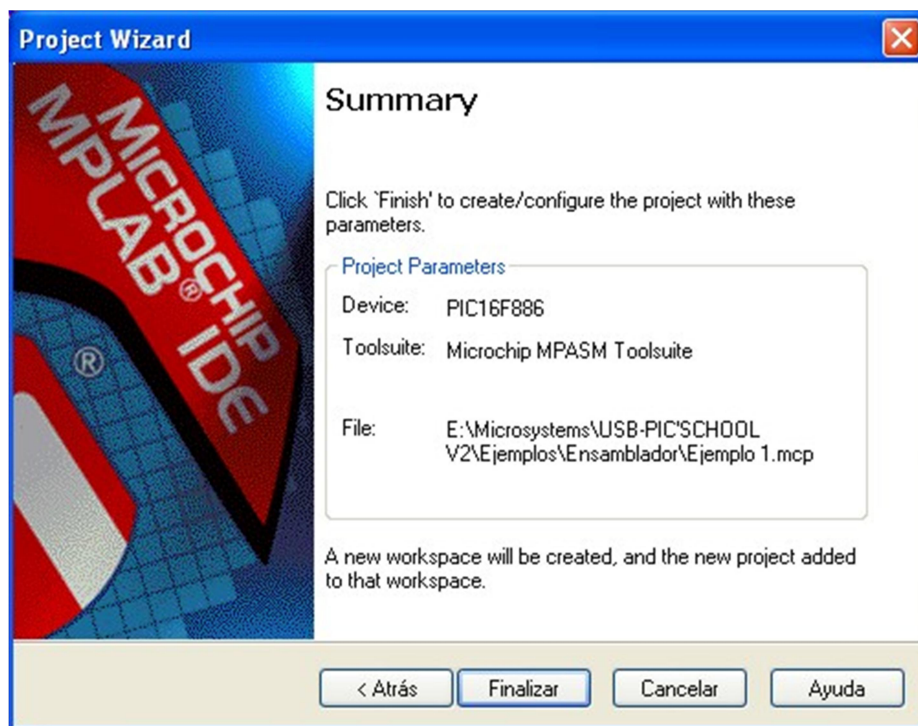
Seleccionamos el fichero "Ejemplo 1.asm" y se pulsa el botón "Add". Al proyecto Ejemplo 1 le hemos asignado el programa fuente Ejemplo 1.asm

Conviene indicar que un proyecto puede tener asignado más de un programa fuente.

Figura 3-6. Asignación del programa fuente



Por último el gestor de proyectos presenta una ventana resumen como la mostrada en la figura 3-7 donde se resume el proyecto.



Se indica tipo de dispositivo empleado, herramienta de ensamblado/compilación y carpeta y nombre del fichero que contiene el proyecto.

Cuando se pulsa *Finalizar* se visualiza la ventana principal de MPLAB-IDE que muestra el área de trabajo y una pequeña ventana que resumen al proyecto Ejemplo 1.mcp y los ficheros asociados.

Figura 3-7. Sumario del proyecto Ejemplo 1.mcp

También podemos configurar el área de trabajo de MPLAB a nuestro gusto, visualizando las ventanas que nos interesen, su tamaño, posición, etc.. Por ejemplo y, como referencia, podemos abrir el programa fuente para tenerlo siempre a la vista. Se emplea la opción "File/Open" del menú principal y se busca y selecciona el fichero Ejemplo 1.asm. Mediante la opción "View" se pueden abrir ventanas que representan la memoria RAM del dispositivo (File Registers), sus registros internos (Special Function Registers), la memoria EEPROM de datos (si la hubiera), etc. La figura 3-8 puede servir como muestra del área de trabajo configurada a gusto del usuario.

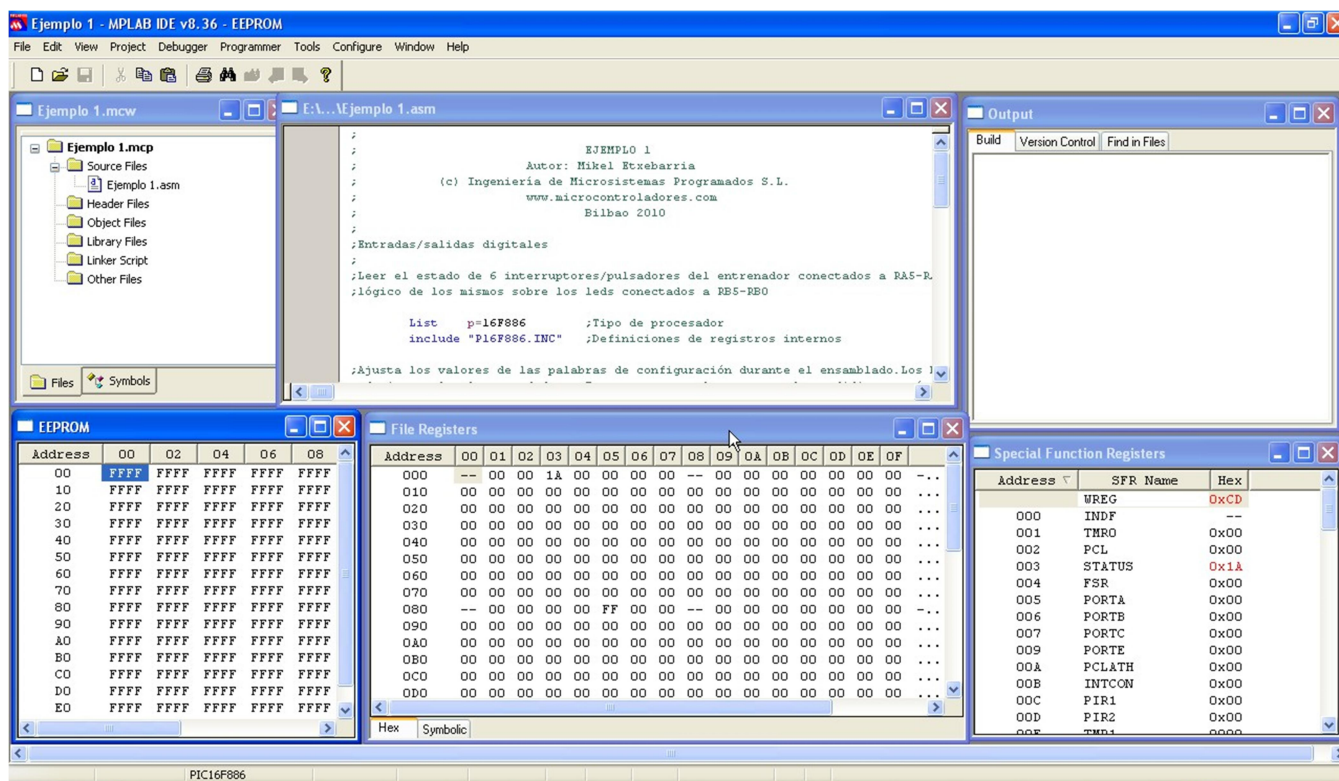
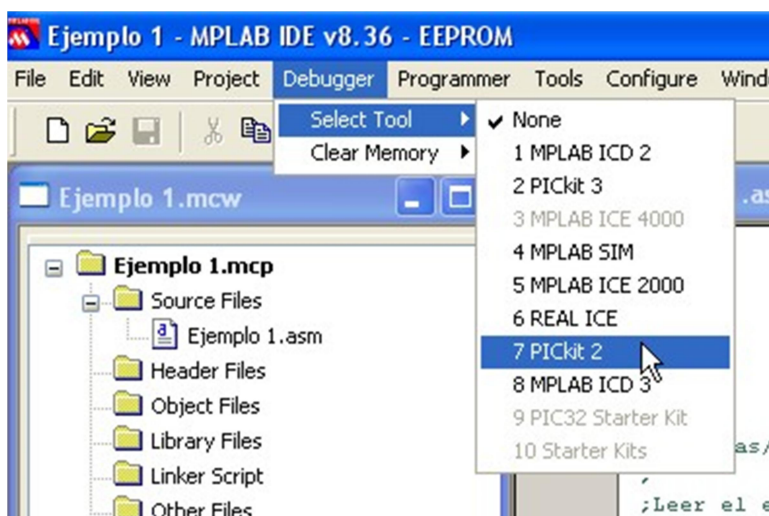


Figura 3-8. Configurando el área de trabajo



Se puede seleccionar el tipo de herramienta para realizar la puesta a punto del programa. En la figura 3-9 se ha seleccionado el depurador PICKit-2 que es compatible con USB-PIC'School, pero se hubiera podido seleccionar el emulador MPLAB ICE 2000 o el depurador en circuito MPLAB ICD 2, si estuvieran disponibles.

Figura 3-9. Selección del simulador

Todos estos ajustes y configuraciones también forman parte del proyecto. Mediante las diferentes opciones del menú "Project" se puede salvar un proyecto, recuperarlo, iniciar uno nuevo, etc.

En el CDROM que se incluye con el equipo, se adjunta el manual completo, en inglés, del MPALB-IDE. En la red es posible encontrar herramientas equivalentes a MPLAB-IDE, aunque nosotros recomendamos su empleo por ser un producto original fabricado por el propio Microchip. Su aprendizaje puede resultar un poco costoso, pero es una herramienta potente, actualizada y que permite cubrir diferentes etapas desde la simulación hasta la emulación en tiempo real.

estos ficheros puede ser empleado por múltiples programas sin necesidad de tener que teclearlos cada vez. En el ejemplo se incluye el fichero P16F886.INC que contiene la declaración de etiquetas de bits y registros de ese dispositivo. Estas declaraciones son siempre iguales por lo que no es necesario escribirlas cada vez que se necesiten. Se hace un fichero con ellas y luego se “incrusta” en todos los programas fuente que las necesite.

La directiva “__config” permite establecer cómo han de quedar los bits de configuración del dispositivo a la hora de grabarlo. Esta información queda registrada en el fichero *.HEX y es empleada por el software de grabación. En el ejemplo que nos ocupa se ha seleccionado, entre otras, una configuración de Protección de código y datos=OFF, LVP=OFF, WDT=OFF y OSC=Externo, etc. Se recomienda la lectura del Data sheet correspondiente para analizar y familiarizarse con la configuración de un determinado modelo de PIC.

El programa comienza en la dirección 0x0000 también llamada “*vector de Reset*”. Todos los programas deben comenzar configurando las líneas de E/S del PIC, determinado quiénes y de qué tipo son las entradas y quiénes las salidas. En el ejemplo la puerta A se configura como entrada digital y la B se configura como salida.

El cuerpo principal del ejemplo comienza en la dirección indicada por la etiqueta **Loop**. De una forma constante se procede a leer el estado de las entradas para reflejarlo sobre las salidas. Las entradas se supone conectadas a unos interruptores y las salidas a unos indicadores tipo led, periféricos contenidos en el laboratorio USB-PIC'School.

Por último la directiva **end** se pone al final del programa fuente y su empleo es obligatorio. Le informa al ensamblador del final del mismo.

3.2.3 Preparando el hardware de la aplicación

El programa de este ejemplo de aplicación consiste en leer el estado de 6 interruptores de entrada conectados a las líneas RA0-RA5 de la puerta A y reflejarlo en otras tantas salidas conectadas a las líneas RB0-RB5 de la puerta B.

La figura 3-10 muestra el esquema de conexiones de los interruptores de entrada y los leds de salida del laboratorio USB-PIC'School conectados a las líneas de las Puertas A y B respectivamente.

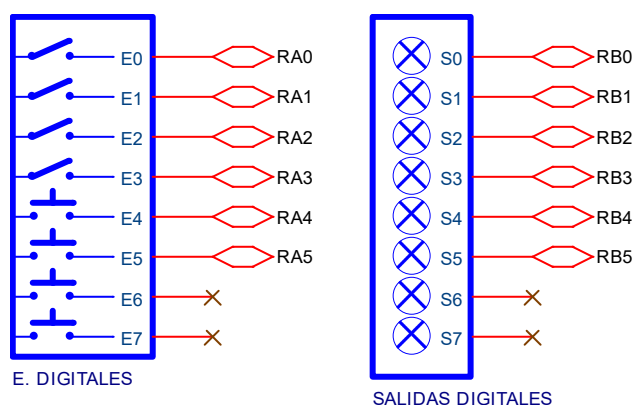


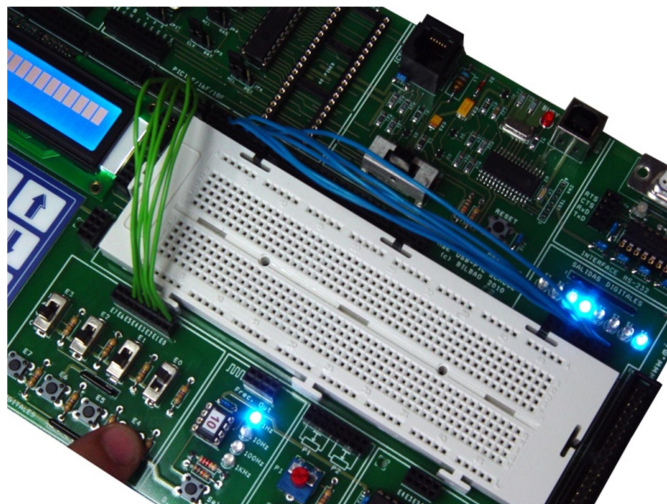
Figura 3-10. Conexiones eléctricas de los interruptores y leds en USB-PIC'School

Las entradas están disponibles a través del bloque de conexiones correspondiente y están representadas como E0-E7. Las cuatro primeras se corresponden con los interruptores E0-E3 y las cuatro últimas con los pulsadores E4-E7. En este ejemplo sólo se conectan las entradas E0-E5 con las líneas RA0-RA5 del microcontrolador.

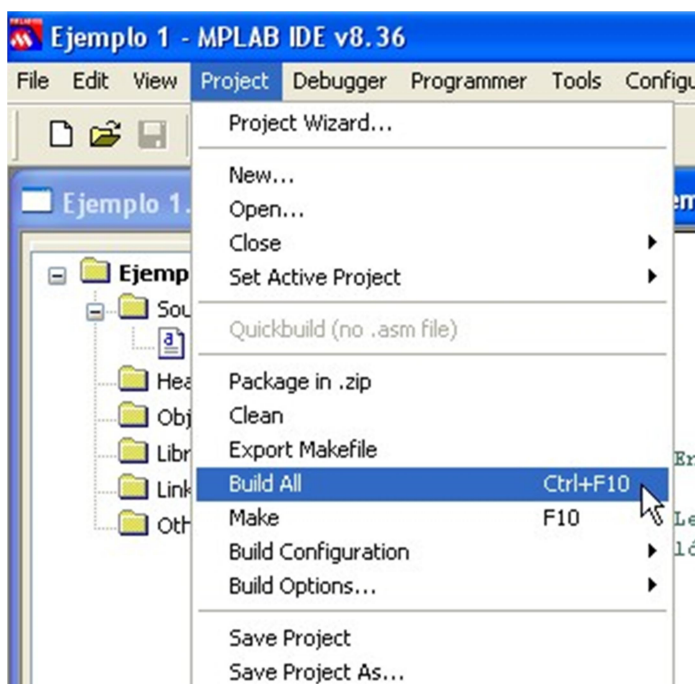
Las conexiones con los leds de salida se realizan a través de su bloque de conexiones y están representados como S0-S7. En este ejemplo de aplicación únicamente se emplean los leds S0-S5 que se conectan con las líneas de salida RB0-RB5 del PIC.

Todas las conexiones se pueden realizar mediante cable rígido de conexiones de 0.6 a 0.8 mm de grosor tal y como se muestra en la fotografía de la figura 3-11.

Figura 3-11. Conexiones realizadas sobre USB-PIC'School



3.2.4 Construyendo el proyecto



Básicamente consiste en ensamblar/compilar el programa fuente (*.ASM) para obtener el programa en código máquina (*.HEX). Recordemos que nuestro proyecto "Ejemplo 1.mcp" está compuesto del programa fuente "Ejemplo 1.asm". Como ya se ha explicado anteriormente, tras el ensamblado se obtiene una serie de ficheros como es el ejecutable "Ejemplo 1.hex", de listado "Ejemplo 1.lst", de errores "Ejemplo 1.err", de simulación "Ejemplo 1.cod", etc..

La figura 3-12 muestra alguna de las opciones del menú "Project", entre las que se encuentra la opción "Build All" (ctrl.+F10) que es la que emplearemos para proceder a ensamblar.

Figura 3-12. Ensamblando el proyecto

3.2.5 Grabación del dispositivo

Tras el ensamblado se obtiene el fichero ejecutable Ejemplo 1.hex. Se supone preparado el hardware sobre la USB-PIC'School con las conexiones necesarias. Lo único que queda por hacer es grabar el dispositivo. Esto se puede realizar desde el mismo entorno de desarrollo MPLAB-IDE realizando los siguientes pasos:

- 1º Conectar la alimentación del laboratorio USB-PIC'School.
- 2º El conmutador del laboratorio debe estar en la posición "USB"
- 3º Conectar el cable de comunicaciones entre el laboratorio y el puerto USB del PC.
- 4º Mediante "Programmer → Select Programmer" seleccionar el PICkit-2 como herramienta de grabación (compatible con USB-PIC'School).

5º Proceder a ensamblar el programa fuente mediante "Project→Build All". Como ya se explicó, esto inicia el proceso de ensamblado del programa fuente. Si no se detecta ningún error, el ejecutable se graba automáticamente sobre el PIC16F866 alojado en el laboratorio. **¡¡ ASI DE FÁCIL !!**

6º Si el sistema de Depuración/Grabación del USB-PIC'School está debidamente conectado y configurado, como ya se explicó en el tema anterior, deberá aparecer una ventana como la mostrada en la figura 3-13. En ella se informa de que el PIC16F866 ha sido borrado, grabado con el nuevo programa recién ensamblado y verificado.

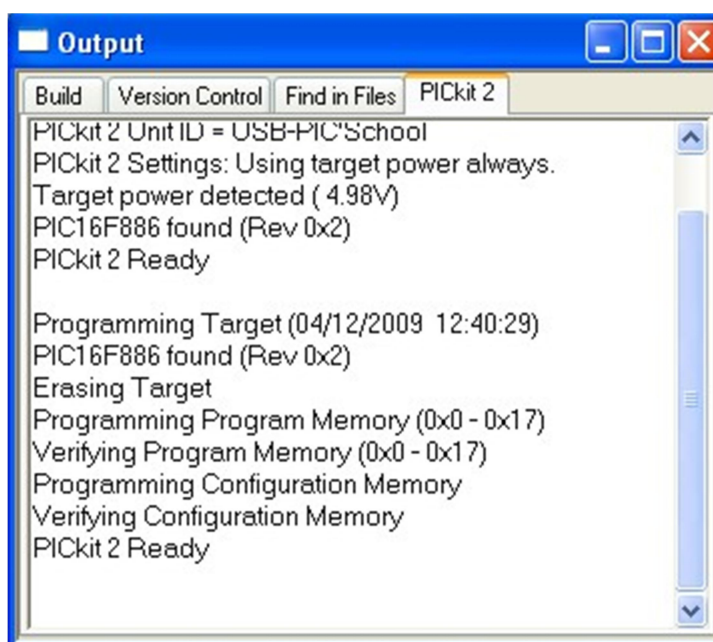


Figura 3-13. Proceso de grabación

¡¡ Todo ello sin salir del entorno MPLAB-IDE desde donde se editó el programa fuente, se ensambló y se grabó sobre el PIC16F886 alojado en USB-PIC'School !!

3.2.6 Comprobando el funcionamiento de la aplicación

El dispositivo PIC16F886 ya está grabado, vamos a verificar el correcto funcionamiento de la aplicación:

- 1º Colocar el conmutador del grabador del laboratorio USB-PIC'School en la posición **RUN**
- 2º Comprobar que cualquier actuación sobre los interruptores/pulsadores de entrada queda inmediatamente reflejado sobre los leds de salida. Recordar que los pulsadores están normalmente, en situación de reposo, a nivel "1". Cuando se accionan generan nivel lógico "0".

Es el momento de experimentar con el programa Ejemplo 1.asm para modificarlo como queramos y repetir todos los pasos explicados en el presente tutorial. También es el momento de ir revisando los diferentes ejemplos propuestos a continuación y cuyos programas fuentes se suministran en el CD-ROM.

3.3.7 Depuración del ejemplo anterior

Es otra, la mejor, de las prestaciones del laboratorio, **¡¡ La Depuración !!**. Como ya se explicó en el TEMA 2, USB-PIC'School tiene capacidad para hacer depuración en circuito en tiempo real del programa recién ensamblado/compilado. La posibilidad de ejecutar ese programa paso a paso, con puntos de parada (breakpoints), etc., así como el poder visualizar/modificar registros internos, RAM de datos, etc., nos permite seguir el desarrollo del mismo y localizar de forma efectiva los posibles fallos.

Básicamente se siguen los mismos pasos que para la grabación:

- 1º Conectar la alimentación del laboratorio USB-PIC'School.
- 2º El conmutador del laboratorio debe estar en la posición "**USB**".
- 3º Conectar el cable de comunicaciones entre el laboratorio y el puerto USB del PC.
- 4º Mediante "**Debugger→Select Tool**" seleccionar el PICkit-2 como herramienta de depuración (compatible con USB-PIC'School). Recordar que si estuviera seleccionado el PICkit-2 como programador, habría que deshabilitarlo. **Sólo puede seleccionarse uno de los dos al mismo tiempo: El Depurador o el Grabador.**

5º Proceder a ensamblar el programa fuente mediante “*Project→Build All*”. Como ya se explicó, esto inicia el proceso de ensamblado del programa fuente. Si no se detecta ningún error, el ejecutable se graba automáticamente sobre el PIC16F866 alojado en el laboratorio. **¡¡ ASI DE FÁCIL !!**

6º Si el sistema de Depuración/Grabación del USB-PIC'School está debidamente conectado y configurado, como ya se explicó en el tema anterior, deberá aparecer una ventana como la mostrada en la figura 3-14. En ella se informa de que el PIC16F866 ha sido borrado, grabado con el nuevo programa recién ensamblado y verificado. Además se le ha grabado distintas ejecutivas y vectores para la depuración. Finalmente se visualiza el mensaje “**PICkit 2 Ready**”.

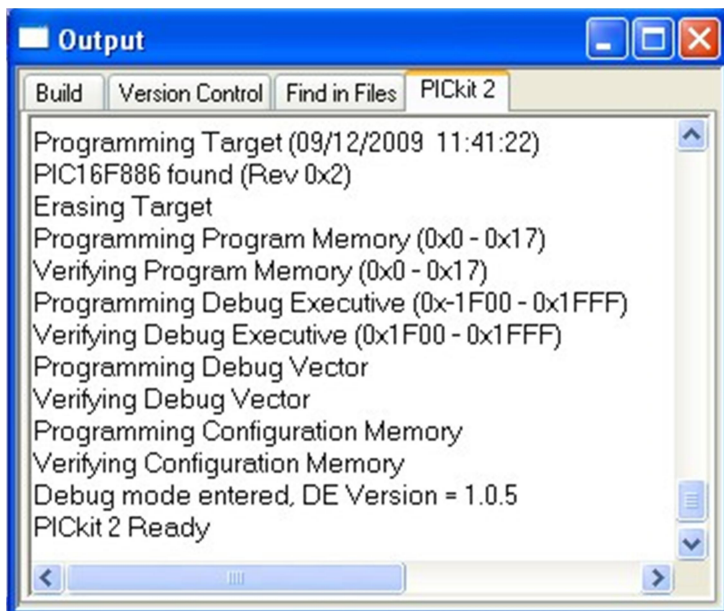


Figura 3-14. Proceso de depuración

7º Ocasionalmente, se hay algún problema de comunicación con el PC, puede ser necesario establecer la misma mediante la opción “*Connect*” y proceder a grabar manualmente mediante la opción “*Program*” del menú “*Debugger*”

8º **El conmutador se debe mantener permanentemente en la posición USB** ya que en el modo de depuración hay un flujo constante de información entre el PC y el PIC del laboratorio.

9º Mediante las opciones del menú “*Debugger*” o bien de los iconos previstos al efecto, el programa se va ejecutando paso a paso, con puntos de parada, etc..

Repasar, en el TEMA 2, el apartado dedicado a la depuración. En él se explica sus ventajas y limitaciones. También se detallan las funciones propias y típicas empleadas en la fase de depuración y puesta a punto de un programa.

3.3 EJEMPLOS

Por fin ha llegado el momento de ir probando con el resto de ejemplos que se suministran en el CD-ROM. Además de aprender ciertas técnicas y “trucos” de programación, con los ejercicios propuestos también nos vamos a familiarizar con el manejo del laboratorio USB-PIC'School, sus periféricos y las conexiones de los mismos con las líneas de E/S del PIC.

Igualmente estos ejemplos tratan de hacernos comprender el funcionamiento básico y las posibilidades de los diferentes recursos internos que incorpora el PIC16F886, uno de los más completos de su categoría: E/S, Timers, conversores AD, comparadores, y mucho más.

Los ejemplos propuestos tienen un evidente carácter didáctico. A veces hay que comprometer sencillez con precisión y claridad con funcionalidad. Por ello no se les debe considerar como aplicaciones finales de tipo real o comercial. Todos los ejemplos son susceptibles de modificarlos, mejorarlos y adaptarlos a necesidades concretas. Animamos desde aquí a que así sea.

Con objeto de ofrecer otras formas de programación, todos los ejemplos se presentan escritos en lenguaje ensamblador y en lenguaje C (Ejemplo xx.ASM o Ejemplo xx.C). El lenguaje ensamblador es el lenguaje original de los controladores y su traductor a código máquina o ensamblador es suministrado gratuitamente por Microchip (normalmente integrado con el MPLAB-IDE). Como lenguaje C hemos empleado la versión PCM de la firma CCS. Este compilador es comercial y por tanto NO es gratuito.

En algunos casos el ejemplo escrito en C presentará algunas ventajas respecto al mismo ejemplo escrito en ensamblador. Esto es porque en aquellos casos que hace falta realizar algún tipo de cálculo matemático, el lenguaje C ofrece unas claras e indiscutibles ventajas.

En cualquier caso en el CD-ROM, además de los programas fuente escritos tanto en ensamblador como en lenguaje C, también se proporcionan los correspondientes programas ejecutables (*.HEX) que se obtuvieron al ensamblar y al compilar. Estos ejecutables se pueden grabar directamente sobre el PIC y listo para funcionar.

También se incluyen ficheros auxiliares y librerías con extensión .INC o .H según estén escritas en ensamblador o en C respectivamente. Estos ficheros contienen información o rutinas que ejecutan una serie de tareas comunes a los diferentes ejemplos. En lugar de escribirlas cada vez, se escriben en diferentes ficheros que luego pueden incluirse en todos los programas fuente que las requieran. Se adjunta una relación de los mismos:

FICHERO	DESCRIPCION
<i>P16F886</i>	Contiene las etiquetas que definen los registros y los bits del PIC16F886
<i>MSE_Delay</i>	Contiene una macro para realizar temporizaciones variables a partir de una determinada velocidad de trabajo
<i>Lcd4bitsPIC16</i>	Contiene rutinas para el manejo de una pantalla LCD alfanumérica compatible con el controlador HD44780. Se adaptan a la familia PIC16 mediante un interface de 4 bits
<i>Teclado</i>	Contiene rutinas para el control de una teclado matricial de 4 x 4 como el del laboratorio USB-PIC'School
<i>I2C_16FXXX</i>	Contiene rutinas básicas para la comunicación I2C mediante controladores de la familia PIC16
<i>1-Wire_PIC16</i>	Contiene rutinas para implementar, vía software, comunicación serie según el protocolo 1-wire
<i>DS18S20_PIC16</i>	Contiene rutinas y funciones para el control del sensor de temperatura DS18S20 de Dallas. Adaptadas para la familia PIC16.
<i>DS1307_16FXXX</i>	Contiene rutinas y funciones para el control del reloj/calendario en tiempo real (RTC) DS1307. Adaptadas para la familia PIC16

3.3.1 Ejemplo 1: Simple manejo de las entradas y salidas

Es el ejemplo que se empleo en el tutorial del apartado 3.2 anterior.

3.3.2 Ejemplo 2: Función lógica NOT

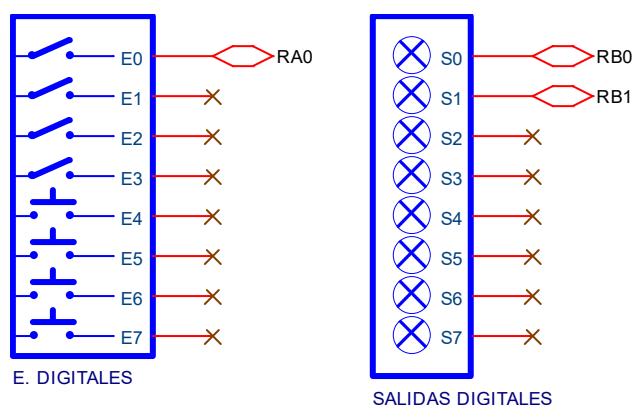
Objetivos

Leer el estado de un interruptor conectado a RA0 y visualizarlo sobre un led conectado a RB0. En RB1 se conecta otro led que visualiza el complemento de RA0.

Esquema Eléctrico de conexiones

Realizar las conexiones mostradas en el esquema de la figura 3-15 en las que RA0 se conecta con el interruptor E0, RB0 se conecta con el led S0 y RB1 se conecta con el led S1.

Figura 3-15. Conexiones a realizar con el ejemplo 2



Comentarios

Este ejemplo realiza un sencillo tratamiento a partir de una señal de entrada.

3.3.3 Ejemplo 3: Decodificador combinacional

Objetivos

El presente ejemplo propone generar unas señales de salida sobre RB0-RB7 en función de dos señales de entrada que se aplican por RA0 y RA1, según la siguiente tabla de la verdad. Se trata de una clásica aplicación combinacional.

ENTRADAS		SALIDAS							
RA1	RA0	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
0	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1
1	0	0	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0	0

Esquema

Las líneas de entrada RA0 y RA1 se conectan con los interruptores E0 y E1 respectivamente. Las líneas RB0-RB7 se conectan con los leds S0-S7. Ver el esquema de la figura 3-16.

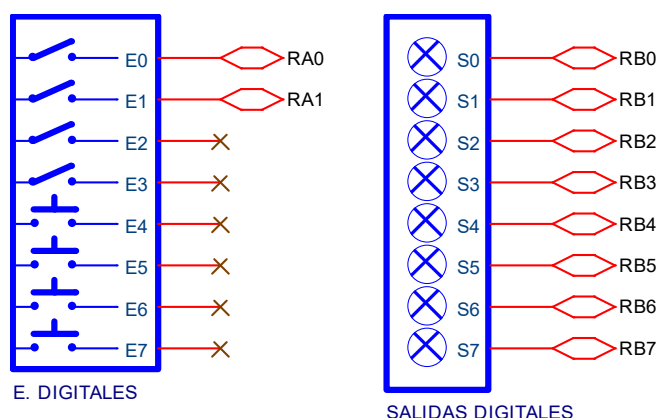


Figura 3-16. Conexiones del Ejemplo 3

Comentarios

El programa muestra el empleo de las máscaras que permiten filtrar una información de entrada de varios bits y procesar sólo los que interese.

Recordemos que cuando vayamos a grabar el dispositivo con este programa, el conmutador del laboratorio debe estar en la posición **USB**. Una vez grabado y realizadas las conexiones necesarias pasamos el conmutador a la posición **RUN** para observar la ejecución y el funcionamiento de las salidas conectadas e RB7:RB0.

Sin embargo, si elegimos trabajar en modo de depuración, el conmutador se debe quedar en la posición **USB** tanto cuando se graba el programa como cuando se ejecuta el mismo bien sea paso a paso, con punto de parada o en tiempo real. En este caso veremos que las salidas RB7 y RB6 **NO FUNCIONAN**. Esto es porque en el modo de depuración estas líneas las emplea el propio sistema ICD para la comunicación con el PC y no están por tanto disponibles como líneas de E/S del usuario. Es una de las limitaciones propias de un depurador ICD tal y como ya se explicaron el TEMA 2 de este manual.

Este detalle debe tenerse muy presente en aquellos ejemplos o aplicaciones que vayan a emplear a RB7 y RB6 como líneas de E/S.

3.3.4 Ejemplo 4: Automatismo secuencial, simulando una maniobra de torneado

Objetivos

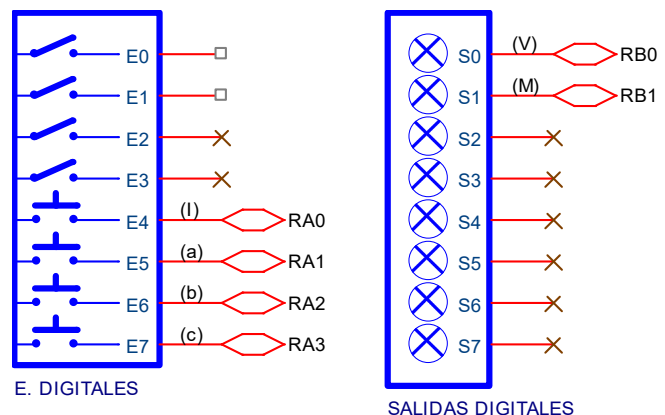
Se trata de simular una sencilla maniobra para el torneado de una pieza. La activación/desactivación de un cilindro simula el arrastre de la pieza a torner. Esta pasa por distintos sensores a lo largo del recorrido provocando que el motor del torno se active o no. El funcionamiento se resume a continuación:

- 1º La secuencia se inicia al pulsar "I" (RA0) que activa el cilindro "V" (RB0).
- 2º Este en su avance empuja la pieza a torner y acciona el sensor "b" (RA2) activando el relé del motor "M" (RB1). Comienza el torneado.
- 3º Durante el avance se alcanza el sensor "c" (RA3) que desactiva el cilindro "V" (RB0) iniciándose el retroceso del mismo.
- 4º Al volver a pasar por "b" (RA2), se desconecta "M" (RB1).
- 5º Cuando se alcanza el final de carrera "a" (RA1) se activa una señal acústica "A" (RB2) para que el operario retire la pieza, coloque otra nueva y, se inicie un nuevo ciclo, al pulsar "I" (RA0).

Esquema

El pulsador "I" E4 se conecta con RA0 e inicia la secuencia. Los pulsadores E6 y E7 simulan los sensores "b" y "c" respectivamente y se conectan con las entradas RA2 y RA3. El pulsador E5 representa al final de carrera "a" y se conecta con RA1. Por su parte el led S0 se conecta con RB0 y representa al cilindro "V" mientras que el led S1 se conecta con RB1 y representa al motor "M". El esquema de conexiones se resume en la figura 3-17.

Figura 3-17. Conexiones del ejemplo 4



Comentarios

Programa que consiste en un conjunto de instrucciones que siguen fielmente la secuencia de funcionamiento de la máquina simulada. Se recuerda por otra parte que los pulsadores empleados en la simulación son activos por nivel "0" cada vez que se accionan.

3.3.5 Ejemplo 5: Automatismo secuencial, simulando una maniobra de taladrado

Objetivos

Se trata de controlar el cabezal de un taladro para la perforación de una pieza. Dicho cabezal es sometido a distintos movimientos de aproximación según la siguiente secuencia:

- 1º Al accionar el pulsador "I" (RA0) el cabezal realiza un descenso rápido de aproximación, activando el motor de bajada rápida "BR"(RB3).
- 2º Al llegar al sensor "b" (RA2), se activa el relé "M" (RB1) que hace girar el motor de la broca y se realiza un descenso lento para el taladrado de la pieza "BL" (RB4).
- 3º Cuando se activa el sensor "c" (RA3) se considera que la pieza está taladrada. Se inicia una subida rápida del cabezal "SR" (RB0) al tiempo que el relé "M" (RB1) de giro sigue activado.
- 4º Cuando se alcanza el final de carrera "a" (RA1), se detiene la subida rápida "SR"(RB0), el relé de giro "M" (RB1) y se activa una señal acústica "A" (RB2) de aviso. El ciclo comienza con una nueva pulsación de "I" (RA0).

Esquema

El pulsador de entrada E4 "I" se conecta con RA0 e inicia la secuencia. Los pulsadores E6 y E7 simulan los sensores "b" y "c" y se conectan con las señales RA2 y RA3 respectivamente. El final de carrera "a" se simula con el pulsador E5 y se conecta con la entrada RA1. Los leds de salida S0-S4 se conectan con las líneas RB0-RB4. Respectivamente simulan las señales que controlan la subida rápida del cabezal (SR), el motor (M), la señal de aviso (A), la bajada rápida del cabezal (BR) y la bajada lenta (BL). Seguir el esquema de conexiones de la figura 3-18.

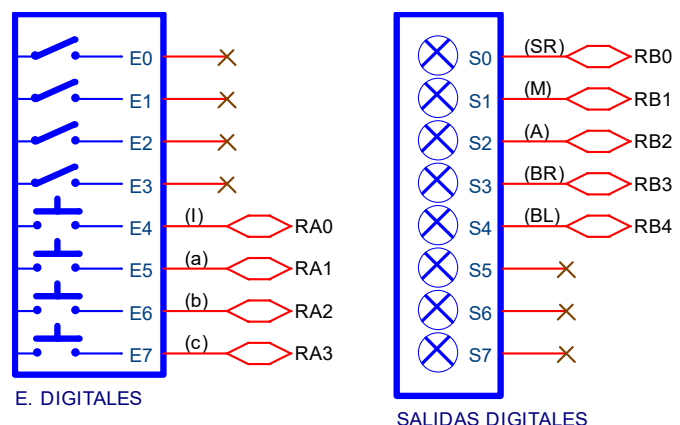


Figura 3-18. Conexiones del ejemplo 5

Comentarios

Este ejemplo al igual que el anterior, consiste en un conjunto de instrucciones que siguen fielmente la secuencia de funcionamiento de la máquina simulada. Se recuerda por otra parte que los pulsadores empleados en la simulación son activos por nivel "0" cada vez que se accionan.

3.3.6 Ejemplo 6: Automatismo, máquina de envasar

Objetivos

Dos cintas transportadoras transportan piezas y envases respectivamente. Cuando se detecta un envase debidamente posicionado comienzan a entrar piezas. Cuando un embase se llena con 10 piezas, se retira y se reemplaza por otro en su lugar. La secuencia de funcionamiento es la siguiente:

- 1º Dos relés "M1" (RB0) y "M2" (RB1) gobiernan dos motores que arrastran dos cintas transportadoras. "M1" (RB0) transporta piezas y "M2" (RB1) embalajes.
- 2º Un sensor "DP" (RA1) detecta el paso de piezas y, otro "DE" (RA2), detecta el correcto posicionamiento de un envase.
- 3º Al detectarse el paso de 10 piezas, el envase se considera lleno, se activa una señal acústica "A" (RB2) y, la cinta que transporta embalajes, se desplaza hasta situar un nuevo envase vacío.
- 4º En este momento se desactiva la señal acústica "A" (RB2) y nuevamente avanza la cinta de piezas repitiéndose así el ciclo.
- 5º Un interruptor "I" (RA0) activa o no a todo el sistema.

Esquema

Mediante el interruptor E0 (I) conectado a RA0 se inicia el funcionamiento de la máquina. El interruptor E1 (DE) se conecta a la línea RA1 y simula el detector de embalaje posicionado. El pulsador E4 (DP) simula el detector de paso de piezas. Se conecta a RA2. Por su parte los leds S0-S2 se conectan con las líneas RB0-RB2 que respectivamente simulan las salidas de avance de piezas (M1), avance de embalaje (M2) y señal acústica (A). Todo ello queda reflejado en el esquema de conexiones mostrado en la figura 3-19.

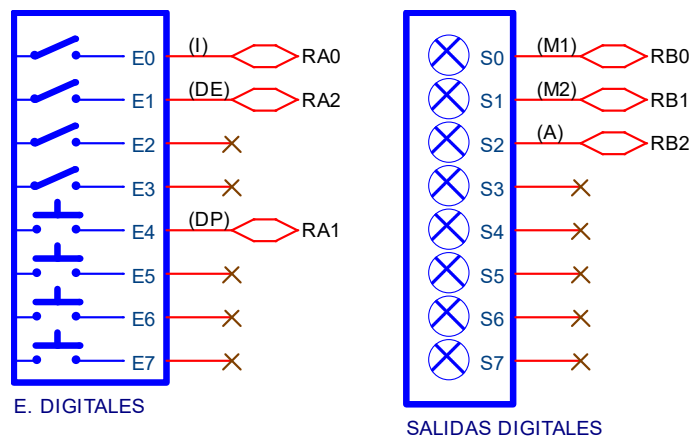


Figura 3-19. Esquema del ejemplo 6

Comentarios

Este ejemplo trata de mostrar el empleo de variables sobre la memoria RAM del microcontrolador y también la forma de contar pulsos en ciertas señales de entrada. El pulsador E4 es el encargado de simular el paso de piezas y genera un pulso con transición 1-0-1 en cada una.

3.3.7 Ejemplo 7: Máquina de envasar, versión mejorada

Objetivos

Dos cintas transportadoras transportan piezas y envases respectivamente. Cuando se detecta un envase debidamente posicionado comienzan a entrar piezas. Cuando un embase se llena con 10 piezas, se retira y se reemplaza por otro en su lugar. La secuencia de funcionamiento es la siguiente:

- 1º Dos relés "M1" (RB0) y "M2" (RB1) gobiernan dos motores que arrastran dos cintas transportadoras. "M1" (RB0) transporta piezas y "M2" (RB1) embalajes.
- 2º Un sensor "DP" (RA1) detecta el paso de piezas y, otro "DE" (RA2), detecta el correcto posicionamiento de un envase.
- 3º Al detectarse el paso de 10 piezas, el envase se considera lleno, se activa una señal acústica "A" (RB2) y, la cinta que transporta embalajes, se desplaza hasta situar un nuevo envase vacío.
- 4º En este momento se desactiva la señal acústica "A" (RB2) y nuevamente avanza la cinta de piezas repitiéndose así el ciclo.
- 5º Un interruptor "I" (RA0) activa o no a todo el sistema.

Esquema

Mediante el interruptor E0 (I) conectado a RA0 se inicia el funcionamiento de la máquina. El interruptor E1 (DE) se conecta a la línea RA1 y simula el detector de embalaje posicionado. El pulsador E4 (DP) simula el detector de paso de piezas. Se conecta a RA2. Por su parte los leds S0-S2 se conectan con las líneas RB0-RB2 que respectivamente simulan las salidas de avance de piezas (M1), avance de embalaje (M2) y señal acústica (A). Todo ello queda reflejado en el esquema de conexiones mostrado en la figura 3-20.

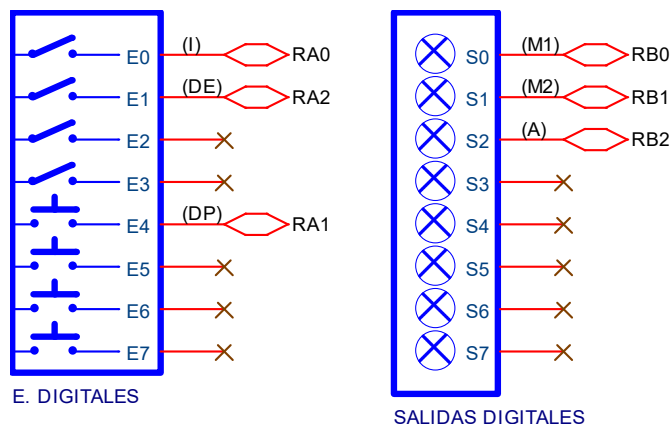


Figura 3-20. Esquema del ejemplo 7

Comentarios

En el ejemplo anterior se pudo comprobar claramente que el nº de piezas a envasar era incorrecto. No es que el programa estuviera mal diseñado, sino que el pulsador que simula el paso de piezas provoca un desagradable efecto rebote. Un único accionamiento del pulsador puede suponer generar varias señales que, por supuesto son detectadas por el PIC. Estos rebotes se producen en todos los dispositivos electromecánicos. Es posible que este efecto se haya notado también en el interruptor que detecta el posicionamiento de embalajes.

Para evitarlo se intercala una temporización al detectar un cambio de estado o accionamiento. De esta forma esperamos que las láminas metálicas de los contactos se estabilicen.

En este ejemplo se hace uso por vez primera de la Macro **Delay** de temporización contenida en el fichero "**MSE_Delay.INC**", que se incluye en nuestro programa fuente de ejemplo mediante la directiva "**include "MSE_Delay.inc"**". Este fichero y las rutinas que contiene se utilizará en numerosos ejemplos posteriores y permite realizar temporizaciones de forma rápida y cómoda. Se obtiene una precisión razonable con valores de temporización desde 1 hasta 2000 mS y con frecuencias del oscilador (Fosc) que van desde 1MHz hasta 48MHz.

Para hacer uso de esta Macro de temporización, nuestros programas fuente deben incorporar obligatoriamente las siguientes directivas (analizar el programa fuente del ejemplo que nos ocupa):

1. "**#define Fosc 4000000**": Para definir la frecuencia a la que trabaja el controlador, y a partir de la cual se calculan los diferentes valores de tiempo. En el caso del laboratorio USB-PIC'School de serie incorpora un oscilador de 4MHz.
2. "**MSE_Delay_V equ 0x73**": Para definir el inicio de un área libre de memoria RAM donde las rutinas de temporización almacenan y manipulan sus propias variables. Esas rutinas emplean 3 posiciones contiguas o variables que, en el ejemplo actual, empiezan en la posición 0x73.
3. "**include "MSE_Delay.inc"**": Esta directiva incorpora o "pega" en nuestro programa fuente el fichero que contiene las rutinas propias que conforman la Macro **Delay** de temporización.
4. "**Delay xxxx Milis**": Es el nombre de la Macro que nos permite hacer una temporización de xxxx Milisegundos. Esta macro instrucción se puede insertar en nuestro programa fuente tantas veces sea necesaria y con el valor deseado.

3.3.8 Ejemplo 8: Juego de luces

Objetivos

Se va a realizar un juego de luces con unos leds conectados a las líneas de la Puerta B. Se iluminarán de forma secuencial de izda. a dcha. y viceversa en función del valor de la entrada RA0. El intervalo en que cada led permanece encendido es de 0.25s (250 mS)

Esquema

Conectar los leds S0-S7 a las líneas RB0-RB7 de salida. El interruptor E0 se conecta con la entrada RA0. Todo ello se muestra en la figura 3-21.

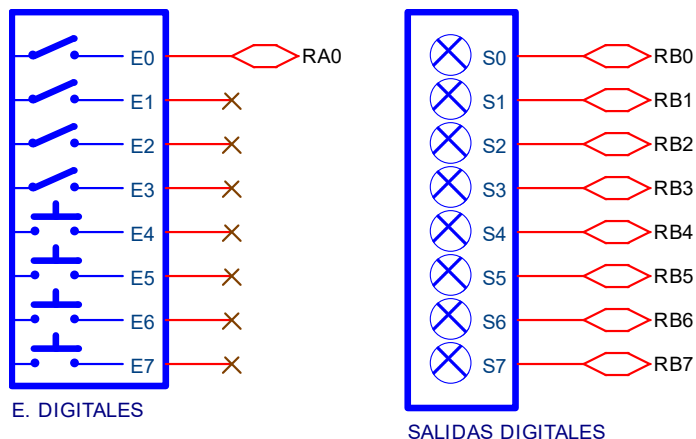


Figura 3-21. Conexiones del ejemplo 8

Comentarios

Nuevamente se hace uso del fichero **MSE_Delay.inc** y de su rutina o Macro **Delay** de temporización.

Recordemos que cuando vayamos a grabar el dispositivo con este programa, el conmutador del laboratorio debe estar en la posición **USB**. Una vez grabado y realizadas las conexiones necesarias pasamos el conmutador a la posición **RUN** para observar la ejecución y el funcionamiento de las salidas conectadas e RB7:RB0.

Sin embargo, si elegimos la depuración como modo de trabajo, el conmutador se debe quedar en la posición **USB** tanto cuando se graba el programa como cuando se ejecuta el mismo bien sea paso a paso, con punto de parada o en tiempo real. En este caso veremos que las salidas RB7 y RB6 **NO FUNCIONAN**. Esto es porque en el modo de depuración estas líneas las emplea el propio sistema ICD para la comunicación con el PC y no están por tanto disponibles como líneas de E/S del usuario. Es una de las limitaciones propias de un depurador ICD tal y como ya se explicaron el TEMA 2 de este manual.

Este detalle debe tenerse muy presente en aquellos ejemplos o aplicaciones que vayan a emplear a RB7 y RB6 como líneas de E/S.

3.3.9 Ejemplo 9: La interrupción del TMR0

Objetivos

Se pretende demostrar el funcionamiento de la interrupción provocada por el rebasamiento del TMR0. El programa lee constantemente el estado de dos entradas y lo visualiza sobre sendas salidas. Al mismo tiempo un led de salida genera una intermitencia constante de 0.5 seg.

Esquema

A la vista del esquema de la figura 3-22 conectar los interruptores E0 y E1 con las líneas de entrada RA0 y RA1. Los leds S0-S1 se conectan con las señales RB0 y RB1 y visualizarán de forma constante el estado de esas entradas. El led S7 se conecta a la salida RB7 y estará constantemente parpadeando cada 0.5 s.

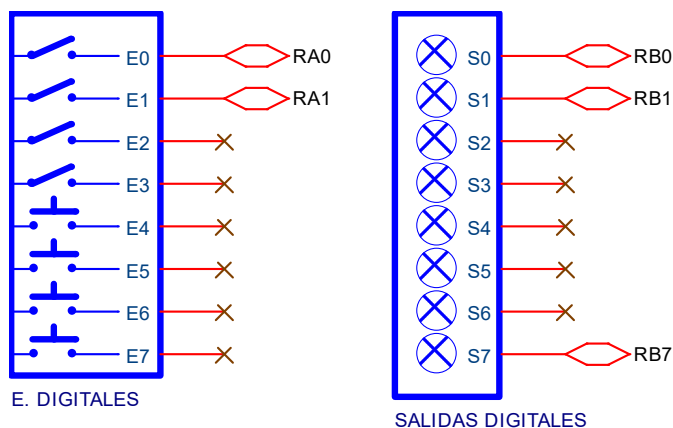


Figura 3-22. Esquema para el ejemplo 9

Comentarios

El ejemplo nos introduce en el mundo de las interrupciones, en este caso la que produce el TMR0 cada 10mS. En este momento el PIC salta a la dirección del vector de interrupción (0x0004) y, desde aquí, se ejecuta el programa de tratamiento correspondiente. En este caso dicho programa consiste en determinar si han transcurrido 50 interrupciones (tiempo total 0.5s.) para hacer que RB7 cambie de estado. No obstante las entradas RA0 y RA1 no dejan en ningún momento de ser reflejadas sobre las salidas RB0 y RB1.

3.3.10 Ejemplo 10: La interrupción externa RB0/INT y el modo sleep

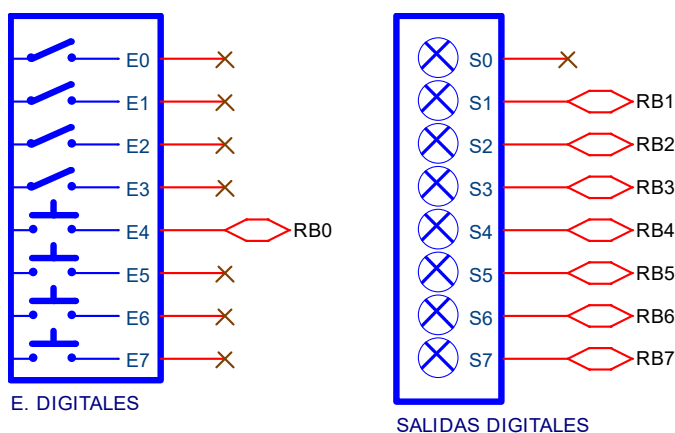
Objetivo

El programa activa las salidas RB1-RB7 durante 1 segundo. Cada vez que se produzca la interrupción externa RB0/INT.

Esquema

Los leds de salida S1-S7 se conectan con las líneas RB1-RB7. La línea RB0 se conecta con el pulsador entrada E4. Ver el esquema de la figura 3-23.

Figura 3-23. Esquema del ejemplo 10



Comentarios

Cada vez que se acciona el pulsador E4 se provoca una interrupción externa por flanco descendente en la patilla RB0/INT configurada como entrada. El programa de tratamiento ilumina los leds de salida durante 1 seg. y los vuelve a apagar, retornando al programa principal. Este simplemente consiste en mantener al PIC en el modo SLEEP de bajo consumo de forma constante que sólo “despierta” cuando se produzca la mencionada interrupción.

Cabe mencionar que en este ejemplo, por variar, la temporización de 1 segundo se realiza mediante el empleo del TMR0. Efectivamente este se ajusta para que desborde cada 10mS. Cuando se hayan contado 100 desbordamientos, habrá transcurrido el segundo deseado.

Recordemos que cuando vayamos a grabar el dispositivo con este programa, el conmutador del laboratorio debe estar en la posición **USB**. Una vez grabado y realizadas las conexiones necesarias pasamos el conmutador a la posición **RUN** para observar la ejecución y el funcionamiento de las salidas conectadas a RB7:RB1.

Sin embargo, si estamos en modo de depuración, el conmutador se debe quedar en la posición **USB** tanto cuando se graba el programa como cuando se ejecuta el mismo bien sea paso a paso, con punto de parada o en tiempo real. En este caso veremos que las salidas RB7 y RB6 **NO FUNCIONAN**. Esto es porque en el modo de depuración estas líneas las emplea el propio sistema ICD para la comunicación con el PC y no están por tanto disponibles como líneas de E/S del usuario. Es una de las limitaciones propias de un depurador ICD tal y como ya se explicaron en el TEMA 2 de este manual.

Este detalle debe tenerse muy presente en aquellos ejemplos o aplicaciones que vayan a emplear a RB7 y RB6 como líneas de E/S.

3.3.11 Ejemplo 11: Interrupciones por cambio de estado en RB2:RB0

Objetivos

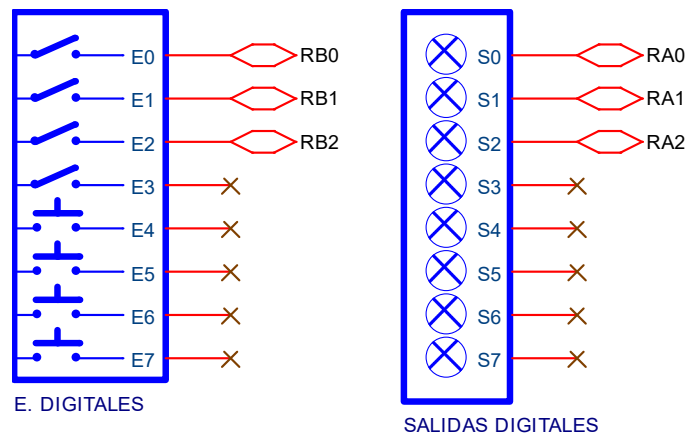
A diferencia de sus predecesores de la familia PIC16F87X, la familia PIC16F88X tiene capacidad de provocar interrupción por cambio de estado en cualquiera de las líneas RB7:RB0 que se configuren como entradas digitales y se les habilite individualmente esa capacidad de interrupción.

En este caso se emplean las líneas RB0-RB2. Un cambio de estado en cualquiera de ellas provocará que las salidas RA0-RA2 reflejen, durante 1 segundo, qué línea cambió de estado.

Esquema

Se muestra en la figura 3-24. Las líneas RB2:RB0 actúan como entradas y se conectan a los interruptores E2:E0. Por su parte las líneas RA2:RA0 actúan como salidas y se conectan a los leds S2:S0.

Figura 3-24. Esquema del ejemplo 11



Comentarios

Dos son los puntos a destacar en este ejemplo. En primer lugar hacemos notar que aunque en todos los ejemplos anteriores la puerta A se configuraba como entrada y la B como salida, esto no tiene porqué ser así. Las puertas A y B se pueden configurar indistintamente como entrada y/o como salida.

También destacamos el hecho de que la interrupción por cambio de estado se puede aplicar a TODAS las líneas de la puerta B que estén configuradas como entradas. Además esa capacidad de interrupción se puede habilitar o no de forma individual a cada una de ellas.

En la familia PIC16F87X dicha capacidad solo era aplicable a las líneas RB7:RB4 que estuvieran configuradas como entradas y se habilitaban en bloque, es decir, o todas esas entradas tenían capacidad de interrupción o ninguna de ellas.

Finalmente apreciamos que el programa principal coloca al controlador en el modo SLEEP de bajo consumo. Solo “despierta” y ejecuta programa cuando se produce una de esas interrupciones.

Resumiremos diciendo que la familia PIC16F88X potencia notablemente su capacidad de interrupciones externas.

3.3.12 Ejemplo 12: El Watch Dog Timer (WDT)

Objetivos

El ejemplo trata de mostrar el funcionamiento del WDT que provoca un RESET periódicamente si no se le refresca. También muestra cómo distinguir entre un RESET provocado por el WDT o por cualquier otra fuente. Efectivamente, si se provoca un RESET por activación de MCLR o el Power On Reset al conectar, se produce un flash de 0.2 seg. sobre la salida RB3. Si el RESET es provocado por desbordamiento del WDT, el flash de 0.1 seg. se realiza sobre la salida RB0. Mediante las entradas RA3:RA0 se ajusta el preescaler de 16 bits asociado al WDT. De esta forma se puede conseguir que el WDT desborde en 12 intervalos diferentes que van desde los 0.13 seg. hasta los 270 seg. El tiempo se duplica de un intervalo al siguiente.

Esquema

Se muestra en la figura 3-25. Los interruptores E0-E3 se conectan con las líneas RA0-RA3. Con ellos se puede aplicar un código binario de 4 bits gracias al cual se ajusta el valor del preescaler del WDT. Los leds de salida S0 y S3 se conectan con las líneas RB0 y RB3 para reflejar qué tipo de reset se ha producido.

Figura 3-25. Esquema del ejemplo 12

Comentarios

El WDT se alimenta a partir de un oscilador interno e independiente de 31KHz (periodo de 32uS). Dispone de un prescaler de 8 bits controlado mediante el OPTION_REG y, otro de 16 bits, que se controla mediante WDTCON. El tiempo total para desbordar es de $32\mu S * \text{OPTION_REG} * \text{WDTCON}$. En el presente ejemplo el prescaler de OPTION_REG es de 128. De esta forma el intervalo de desbordamiento es de $32\mu S * 128 * \text{WDTCON}$. Por su parte WDTCON se carga con el valor binario procedente de las entradas RA3:RA0. De esta forma se obtienen desbordamientos a intervalos variables según la siguiente tabla:

RA3:RA0	WDTCON	TIEMPO	RA3:RA0	WDTCON	TIEMPO
0000	32	0.13 s	0110	2048	8.45 s
0001	64	0.26 s	0111	4096	16.91 s
0010	128	0.52 s	1000	8192	32.82 s
0011	256	1.05 s	1001	16384	67.65 s
0100	512	2.11 s	1010	32768	135.30 s
0101	1024	4.22 s	1011	65536	270.60 s

Mediante este ejemplo también se destaca otra de las diferencias entre la antigua familia PIC1687X y la actual PIC16F88X. En aquella sólo se disponía de un único preescaler de hasta 128 que se gestionaba desde el OPTION_REG. En la familia actual se dispone de lo mismo y se le añade un nuevo registro, el WDTCON, con un nuevo preescaler de hasta 65536. De esta forma el tiempo de desbordamiento del WDT, en caso de no ser refrescado, aumenta hasta los 270.60 segundos.

3.3.13 Ejemplo 13: El display de 7 segmentos

Objetivos

El programa presenta sobre uno de los displays de 7 segmentos del laboratorio USB-PIC'School, el estado lógico presente en la entrada RA0, visualizando "1" o "0" según corresponda. Con la entrada RA1 se activa o no el punto decimal del display.

Esquema

Como se muestra en el esquema de la figura 3-26 basta con conectar las líneas RB0:RB7 con los segmentos a:dp del display. Se trata de un display de color azul y alto contraste del tipo de ánodo común. El ánodo correspondiente al display de las unidades se conecta a +5Vcc. El de las decenas no se emplea y por tanto no se conecta. Ya que son de ánodo común, cada segmento se activará cuando se le aplique nivel lógico "0".

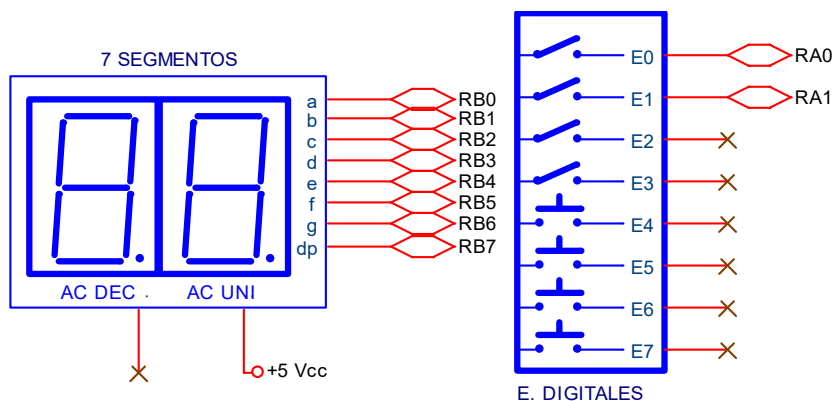
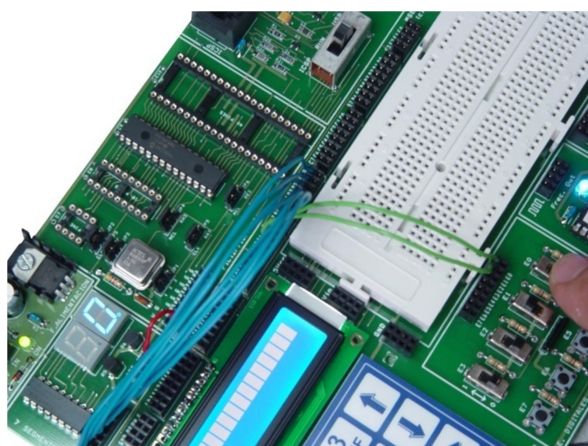


Figura 3-26. Esquema de montaje del ejemplo 13



Los interruptores E0 y E1 se conectan con las señales RA0-RA1 respectivamente, que serán configuradas como entradas. En la figura 3-27 se muestra el montaje práctico.

Figura 3-27. Montaje práctico del ejemplo 13

Comentarios

El ejemplo es una simple muestra de cómo visualiza dos números (0 y 1) sobre uno de los dos displays de 7 segmentos del laboratorio. Sirve como una primera toma de contacto con la decodificación a 7 segmentos.

3.3.14 Ejemplo 14: Decodificador BCD-7 segmentos

Objetivo

Mediante 4 señales de entrada se introduce el código BCD (desde 0 hasta F) del dígito a visualizar sobre el display 7 segmentos correspondiente a las unidades.

Esquema

Mediante los interruptores de entrada E0-E3 conectados a las líneas RA0-RA3 se aplica el código BCD del dígito a visualizar. Las líneas RB0-RB7 se conectan con los segmentos a-dp del display. Todo ello se muestra en el esquema de la figura 3-28 que es muy similar al del ejemplo anterior.

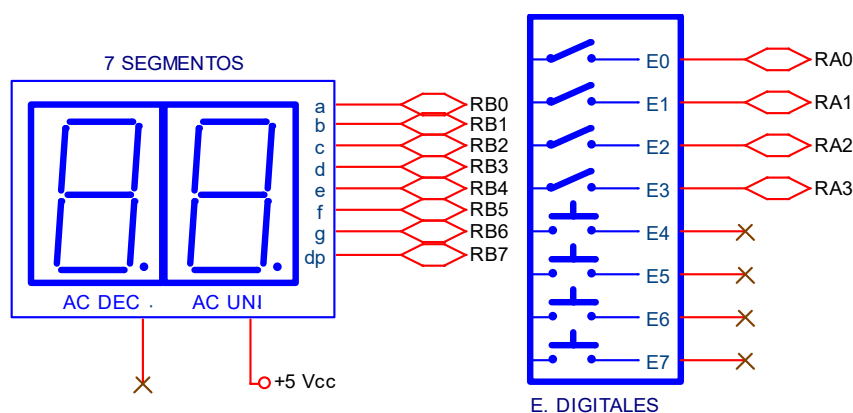


Figura 3-28. Esquema del ejemplo 14

Comentarios

Este ejemplo pretende introducirnos al manejo de tablas con datos constantes. Efectivamente, la tabla contiene los 16 códigos de 7 segmentos de otros tantos dígitos a visualizar. Mediante la suma del PC con un desplazamiento, se selecciona cualquiera de los elementos de esa tabla, en este ejemplo el del código 7 segmentos a visualizar.

3.3.15 Ejemplo 15: Barrido de displays

Objetivos

Mostrar la técnica que permite controlar dos o más displays de 7 segmentos conectados en paralelo pero de forma que cada uno visualice el código o dígito que le corresponda. Dicha técnica se le conoce como “barrido de displays”. En el ejemplo se trata de visualizar el valor 35 (se puede mostrar cualquier otro).

Esquema

Las 8 líneas RB0:RB7 de la puerta B se conectan con los segmentos a:dp de ambos displays que se encuentran en paralelo (todos los segmentos son comunes). El ánodo del display de las unidades se conecta y controla desde la salida RA2 y el de las decenas desde RA3. Ver la figura 3-29.

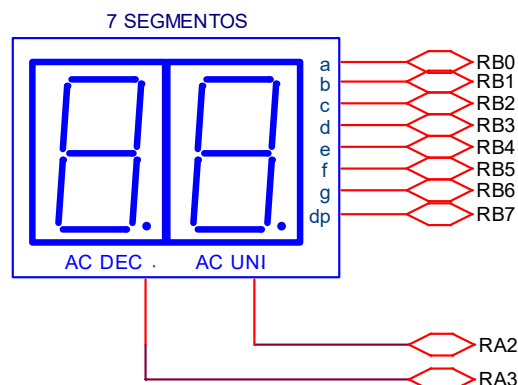


Figura 3-29. Conexiones del ejemplo 15

Comentarios

Dado que todos los segmentos se conectan en paralelo debemos conseguir que cada display visualice únicamente el dígito que le corresponde. Esto se consigue controlando individualmente a cada ánodo común. Efectivamente, si únicamente se activa el ánodo del display de las unidades mediante RA2, el valor que en ese momento se aplique a los segmentos mediante RB7:RB0 afecta y se visualiza exclusivamente en el display de las unidades. A continuación se repite el proceso activando únicamente el ánodo común de las decenas mediante RA3. En este momento el valor presente en las mismas líneas RB7:RB0 afecta y se visualiza exclusivamente en el display de las decenas.

Es decir la idea consiste en ir activando secuencialmente el ánodo de cada display al tiempo que se le envía el código 7 segmentos del dígito que le toca visualizar. Evidentemente sólo hay un display activado cada vez, pero si esa secuencia se repite a una determinada velocidad y cada cierto tiempo, producirá la sensación óptica de que todos los displays están iluminados y cada uno con su información correspondiente. A esto se le llama "barrer" los displays.

Esta técnica permite ahorrar gran cantidad de líneas de E/S ya que, independientemente del número de displays que dispongamos, siempre se consumen las mismas 8 líneas para los segmentos de todos ellos. Cada display necesita una única línea, la común, para su selección y control en particular.

Se trata de un ejemplo experimental en el que se visualiza el valor 35 definido en la constante "Valor" del programa. Este se puede modificar. Se ha creado una rutina llamada "Visu_Disp" que se encarga de barrer secuencialmente cada display manteniéndolos activados durante 1 mS a cada uno. Este valor se puede modificar para apreciar el efecto óptico que se produce.

El programa principal simplemente llama a la rutina "Visu_Disp" cada intervalo de 1 mS. Este intervalo también se puede modificar para apreciar la sensación óptica que se produce. En la figura 3-30 se aprecia el montaje del ejemplo y su ejecución.

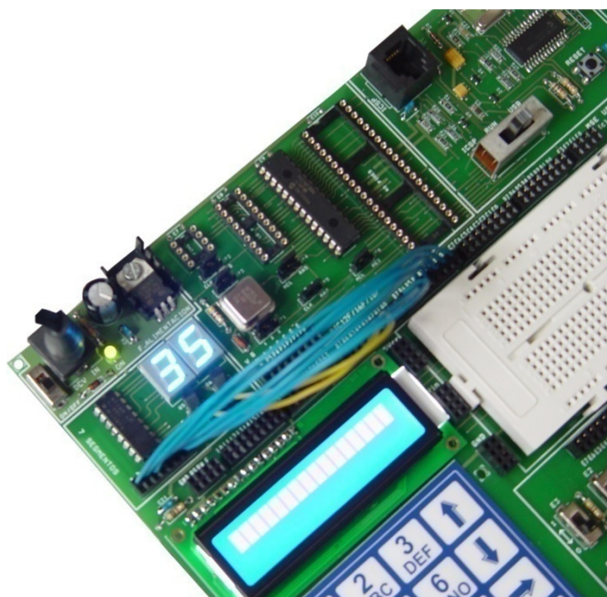


Figura 3-30. Montaje y ejecución del ejemplo 15

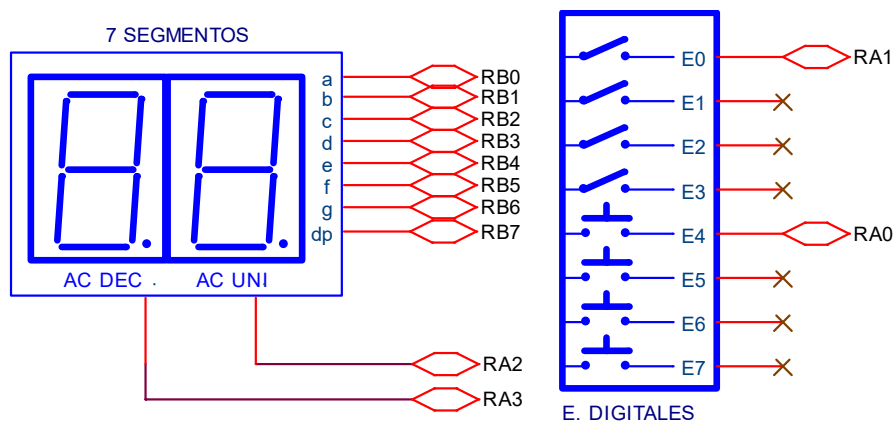
3.3.16 Ejemplo 16: Contador decimal UP/DOWN

Objetivos

Se trata de realizar un contador ascendente/descendente (UP/DOWN) de dos dígitos (00-99). Los pulsos a contar se aplican por RA0 y deben estar libres del efecto rebote. Mediante la entrada RA1 se selecciona si la cuenta es ascendente o descendente.

Esquema

El pulsador E4 se conecta con la línea RA0 y el interruptor E0 con la línea RA1. Las líneas RB0-RB7 se conectan con los segmentos del display. RA2 y RA3 controlan los ánodos de las unidades y decenas respectivamente. Todo ello se muestra en el esquema de la figura 3-31 que es bastante similar al del ejemplo anterior.



Comentarios

Este ejemplo es una aplicación práctica que consiste en realizar un contador decimal ascendente/descendente entre 00 y 99. Se emplea la técnica de barrido explicada en el ejemplo anterior para el control de ambos displays.

Figura 3-31. Esquema del ejemplo 16

También se emplea una temporización de unos 5 mS cada vez que se detecta un flanco ascendente y otro ascendente sobre la entrada de pulsos a contar por RA0. Esto evita el "efecto rebote" que produce el pulsador E4 en esa entrada cada vez que se aplica un pulso (transición 1-0-1). Si eliminamos estas temporizaciones observaremos que la cuenta se "salta" varios pulsos.

La línea RA0 la podemos conectar a la salida del generador lógico del laboratorio USB-PIC'School en lugar de al pulsador E4. Este generador produce pulsos a diferentes frecuencias y libre de rebotes.

3.3.17 Ejemplo 17: El TMR0 como contador de pulsos externos

Objetivos

Procedente del generador lógico del entrenador se aplican pulsos por RA4/T0CKI para que el TMR0 los vaya contando y visualizando sobre el display. Cuando lleguen 6 pulsos el display activa todos sus segmentos durante 1 seg. luego se apagan y el sistema queda detenido hasta provocar un RESET manualmente mediante el pulsador. Comienza un nuevo ciclo.

Esquema

La salida del generador se conecta con la línea RA4/T0CKI. Las líneas RB0-RB7 se conectan con las resistencias de absorción y estas a su vez se conectan con los segmentos del display. Todo ello se muestra en el esquema de la figura 3-32 que es bastante similar al de ejemplos anteriores.

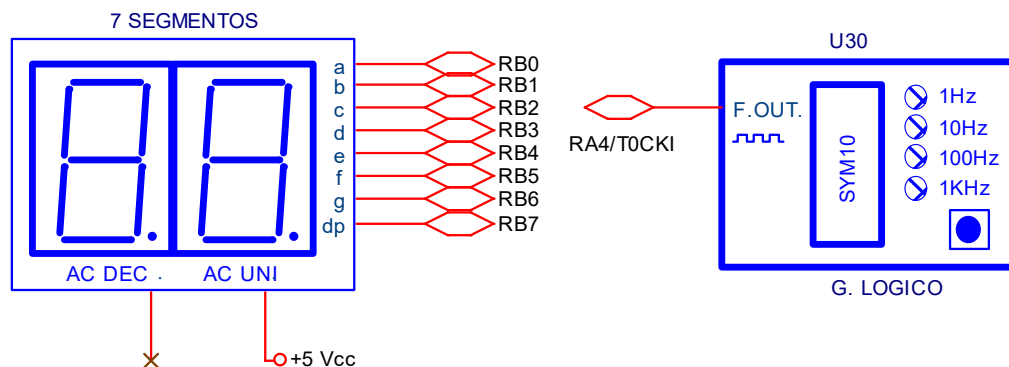


Figura 3-32. Conexiones del ejemplo 17

Comentarios

En este ejemplo se demuestra que el TMR0 puede actuar como contador de eventos o pulsos externos que se introducen por la línea RA4/T0CKI y, si procede, provocar interrupción al finalizar la cuenta. También se demuestra la posibilidad de leer en cualquier momento el valor actual del TMR0 para conocer el estado de la cuenta y visualizarlo sobre el display de las unidades como en el ejemplo.

Por último resaltar que cada vez que finaliza la secuencia el microcontrolador queda en standby de bajo consumo gracias a la instrucción SLEEP. En este ejemplo “despierta” al provocar un RESET manual.

La velocidad del conteo se puede variar modificando la frecuencia que suministra el generador lógico y que se selecciona mediante el pulsador “Frec.Sel.” entre 1Hz y 1KHz.

3.3.18 Ejemplo 18: La memoria EEPROM de datos, la máquina “Su Turno”

Objetivos

Se trata de emular el funcionamiento de la típica máquina “Su Turno” habitual en muchos comercios de venta al público. Sobre el display se visualiza el turno actual. Este se incrementa con cada pulso aplicado por RA0. En la memoria EEPROM de datos del PIC se almacena el turno en curso de forma que, ante un fallo de alimentación, se reanude la cuenta desde donde se quedó.

Esquema

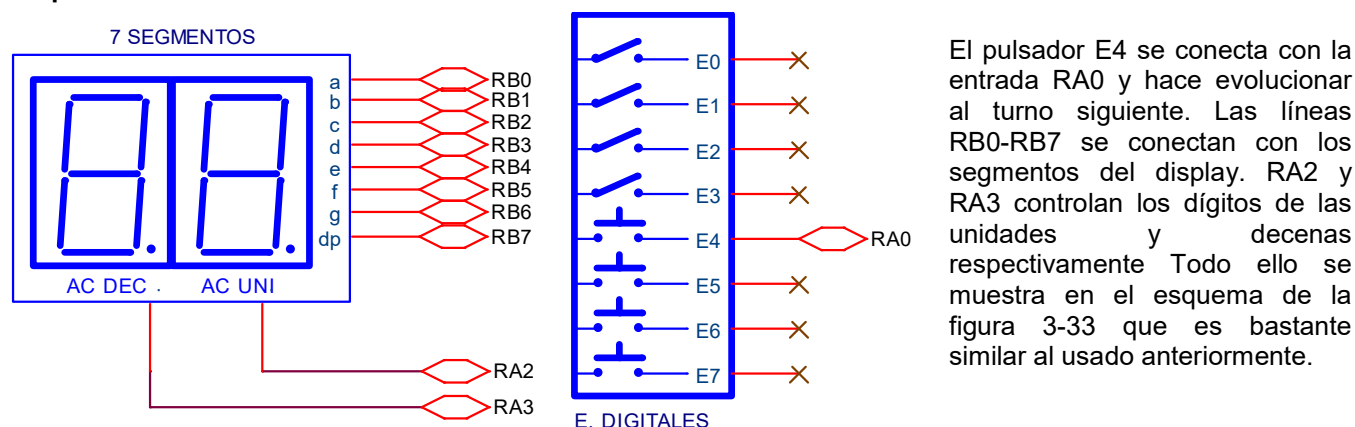


Figura 3-33. Esquema del ejemplo 18

Comentarios

El ejemplo pretende ilustrar la forma de grabar y recuperar datos no volátiles sobre la memoria EEPROM de datos del PIC. Las aplicaciones de este tipo de memoria son numerosas (códigos de acceso, agenda no volátil, etc..)

Incluye rutinas básicas que permiten grabar o leer un byte en cualquier posición de la memoria EEPROM de datos del PIC. También se hace uso de un par de interesantes directivas:

```
org 0x2100
de 0xff
```

Con ellas, cuando se ensambla y/o compila y se genera el fichero .HEX, se incluye en éste información para que el sistema de Depuración/Grabación del laboratorio USB-PIC'School grabe físicamente cualquier valor sobre cualquier posición de la memoria EEPROM del PIC. Así la directiva “org 0x2100” establece la dirección inicial de la EEPROM y la directiva “de 0xff” el dato a grabar (0xff). Obtenemos pues un fichero .HEX con la información a grabar tanto en la memoria de programa, como en la EEPROM de datos, como en la memoria de configuración del PIC.

Ese valor que grabamos en la EEPROM, lo emplea el programa del ejemplo durante el arranque para verificar si la aplicación es la primera vez que se usa. Para ello lee la 1ª posición y averigua si es 0xFF (valor que se escribe sólo durante la grabación del PIC). Si es así se asume que es la 1ª vez y por tanto la cuenta o el turno empieza desde 00. Si no fuera así se asume que en la 1ª posición de la EEPROM hay un turno válido de anteriores ejecuciones y se reanuda desde él.

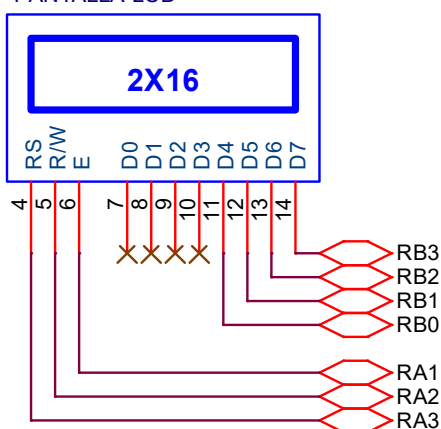
3.3.19 Ejemplo 19: Introducción al manejo de la pantalla LCD

Objetivos

Se trata de hacer una introducción al manejo de la pantalla LCD del laboratorio USB-PIC'School, donde quedará visualizado el mensaje "Hola".

Esquema

PANTALLA LCD



Las líneas de la RB0-RB3 se conectan con las líneas de datos L4-L7 de la pantalla. Se realiza por tanto un interface de 4 bits con ella. Las señales RA1, RA2 y RA3 se conectan con las señales E, R/W y RS respectivamente. Estas conexiones se muestran en el esquema de la figura 3-34.

Figura 3-34. Conexiones con la pantalla LCD

Comentarios

El programa hace uso de las rutinas para el manejo de la pantalla LCD y están contenidas en el archivo **LCD4bitsPIC16.INC**. Dichas rutinas son de propósito general y se pueden emplear en cualquier aplicación que requiera el empleo de una pantalla. Se debe incluir en cualquier programa fuente mediante el simple empleo de la directiva **INCLUDE**. La imagen de la figura 3-35 muestra una fotografía del montaje propuesto y que servirá para sucesivos ejemplos.

Figura 3-35. Montaje y ejecución del ejemplo 19



Para poder hacer uso del fichero **LCD4bitsPIC16.INC** nuestros programas fuente deben incorporar obligatoriamente las siguientes directivas (analizar el programa fuente del ejemplo que nos ocupa):

1. **Lcd_var equ 0x70**: Para definir el inicio de un área libre de memoria RAM donde las rutinas de manejo del LCD almacenan y manipulan sus propias variables. Esas rutinas emplean 3 posiciones contiguas o variables que, en el ejemplo actual, empiezan en la posición 0x70.
2. **include "LCD4bitsPIC16.inc"**: Esta directiva incorpora o "pega" en nuestro programa fuente este fichero que contiene las rutinas y funciones propias para el control de la pantalla.

3. **call nombre_rutina:** Es la llamada, desde nuestro programa fuente, a una de las rutinas para su ejecución. Es posible que antes de llamarlas, algunas rutinas necesiten parámetros de entrada o salida para su correcto funcionamiento. Leer con cuidado los comentarios y explicaciones que acompañan a cada una de ellas en el fichero LCD4bitsPIC16.inc.

En la siguiente tabla se resumen las rutinas más importantes contenidas en el fichero:

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
UP_LCD	Ninguno	Ninguno	Configura las E/S para adaptarlas a la pantalla LCD
LCD_INI	Ninguno	Ninguno	Rutina para la inicialización según especificaciones del fabricante
LCD_DATO	W=Contiene el carácter a visualizar	Ninguno	Envía a la pantalla el dato a visualizar
LCD_REG	W=Contiene el comando a ejecutar por la pantalla LCD	Ninguno	Envía a la pantalla el comando que debe ejecutar.

3.3.20 Ejemplo 20: Seguimos con la pantalla LCD

Objetivos

Se trata de visualizar diferentes mensajes contenidos en tablas previamente definidas por el usuario. En el presente ejemplo se visualizan los mensajes "Hola" y "Adios".

Esquema

Se realizan las mismas conexiones que en el ejemplo anterior y que se muestra en la figura 3-34.

Comentarios

Mostrar una forma cómoda y versátil para visualizar cualquier tipo de mensaje de cualquier tamaño. Los mensajes se definen en tablas como si de datos constantes se tratara. La instrucción RETW devuelve en cada momento el carácter a visualizar. Todo mensaje finaliza con un código especial de fin de mensaje (0x00). Se ha diseñado una rutina llamada "Mensaje" que lee y visualiza todos los caracteres que forman el mensaje hasta llegar al carácter final (0x00). Antes de llamar a esta rutina en el programa principal, se debe indicar el inicio de cada mensaje individual que se desea visualizar.

3.3.21 Ejemplo 21: Visualizando mas mensajes

Objetivos

Al igual que el ejemplo anterior, se trata de visualizar diferentes mensajes contenidos en tablas previamente definidas por el usuario en el programa fuente

Esquema

Se realizan las mismas conexiones que en el ejemplo anterior y que se muestra en la figura 3-34.

Comentarios

El ejemplo es similar al anterior en cuanto a que se trata de visualizar diferentes textos previamente definidos en tablas. La diferencia está precisamente en la forma de definir estas tablas. Efectivamente, mediante la directiva **DT** del ensamblador es posible definir tablas de datos expresados en ASCII, cadenas de caracteres,

hexadecimal, etc. Durante el proceso de ensamblado se generan tantas instrucciones RETLW como datos haya. Es una forma mucho más cómoda, intuitiva y fácil de incluir en nuestros programas tablas de datos constantes.

3.3.22 Ejemplo 22: El teclado

Objetivos

Se trata de una primera toma de contacto con el manejo del teclado del laboratorio USB-PIC'School. Se trata de un teclado matricial de 4x4 de membrana de propósito genera.. El ejemplo consiste en explorar el teclado y visualizar sobre los leds S0-S3 conectados a las líneas RB0-RB3, el código BCD de la tecla pulsada durante 2 segundos.

Esquema

Realizar las conexiones de la figura 3-36. Las líneas RB0-RB3 se conectan con los leds S0-S3 y con las líneas C0-C3 del teclado (las columnas). Las líneas RB4-RB7 se conectan con las líneas F0-F3 (las filas).

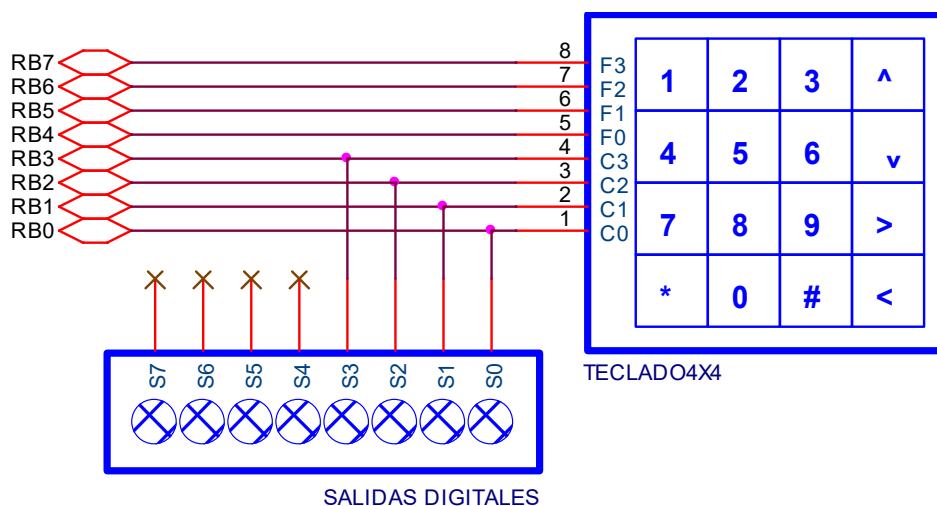


Figura 3-36. Conexión del teclado para el ejemplo 22

Comentarios

Las rutinas para el manejo del teclado se encuentran en el archivo **TECLADO.INC**. Mediante la directiva **INCLUDE** del ensamblador se debe incluir en los programas fuente del usuario que necesite hacer uso de ellas. De todo el contenido de este archivo cabe destacar la rutina **Key_Scan**. Realiza el barrido del teclado y devuelve el código BCD de la tecla pulsada. Si no se pulsa ninguna devuelve el código 0x80.

La exploración o *barrido* de un teclado consiste en activar secuencialmente las columnas (o filas) al tiempo que se lee el estado de las filas (o columnas). La intersección entre una fila/columna activada permite calcular qué tecla se pulsó. A partir de ese momento se pueden emplear determinadas técnicas para codificar dicha tecla y devolver su correspondiente código a nuestro programa principal para que actúe en consecuencia.

Puede ocurrir que tras activar todas las columnas (o filas) no se detecte ninguna fila (o columna) activada. Eso implica que en ese momento no hay ninguna tecla pulsada. Lo normal es devolver un código especial y darse por finalizado el *barrido*.

Teniendo en cuenta la velocidad con la que una persona, por hábil que sea, es capaz de realizar varias pulsaciones seguidas, el tiempo entre un barrido y el siguiente puede ser bastante dilatado desde el punto de vista de un PIC. Suponiendo que alguien sea capaz de pulsar 10 teclas por segundo (¡¡ 600 por minuto !!) implica que al PIC le basta con hacer un barrido cada 0.1 seg (100 mS) y aún así, nos "pilla" con el dedo en la tecla.

Para poder hacer uso del fichero **Teclado.INC** nuestros programas fuente deben incorporar obligatoriamente las siguientes directivas (analizar el programa fuente del ejemplo que nos ocupa):

1. **Key_var equ 0x73**: Para definir el inicio de un área libre de memoria RAM donde las rutinas de manejo del teclado almacenan y manipulan sus propias variables. Esas rutinas emplean 6 posiciones contiguas o variables que, en el ejemplo actual, empiezan en la posición 0x73.
2. **include "Teclado.inc"**: Esta directiva incorpora o "pega" en nuestro programa fuente este fichero que contiene las rutinas y funciones propias para el control del teclado matricial de 4x4 que incorpora el laboratorio USB-PIC'School.
3. **call Key_Scan**: Es la llamada, desde nuestro programa fuente, a la propia rutina de exploración del teclado para su ejecución.

En la siguiente tabla se resume la rutina más importante contenida en el fichero:

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
Key_Skan	Ninguno	W=Código BCD de la tecla pulsada W=0x80 si no se pulsa ninguna tecla en ese instante	Realiza una exploración de las 4 filas y columnas de que consta el teclado del laboratorio.

3.3.23 Ejemplo 23: El teclado y la pantalla LCD

Objetivo

Se trata de combinar el funcionamiento de estos dos importantes periféricos de interface humano. Sobre la pantalla LCD se visualiza la tecla pulsada.

Esquema

Realizar las conexiones mostradas en la figura 3-37. Se puede observar que las líneas RB0-RB3 se emplean tanto para gestionar el teclado (C0-C3) como para la transferencia de datos con la pantalla LCD (L4-L7). Por su parte RB4-RB7 controlan las filas del teclado (F0-F3). Las señales RA1, RA2 y RA3 se emplean como señales de control E, R/W y RS de la pantalla.

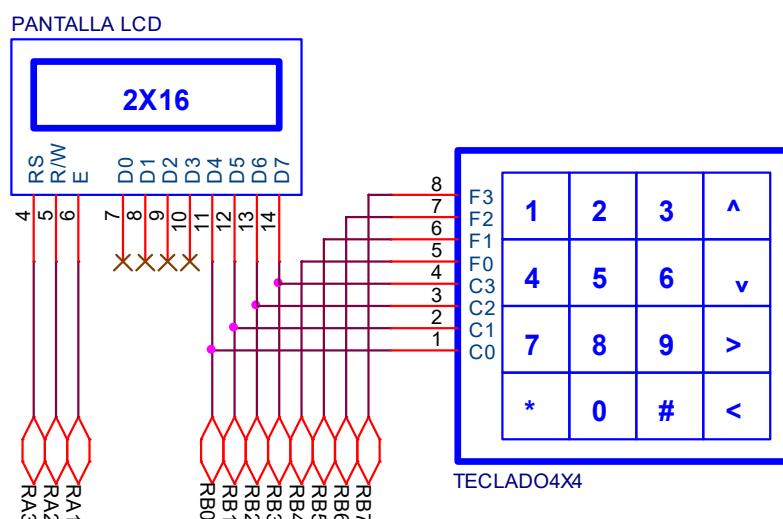


Figura 3-37. Conexiones del Ejemplo 23

Comentarios



El ejemplo trata de mostrar cómo, configurando las puertas en cada momento, el PIC puede gobernar en una misma aplicación periféricos tan dispares como son un teclado y una pantalla LCD. En la fotografía de la figura 3-38 se muestra el montaje práctico y la ejecución del programa.

Figura 3-38. Montaje práctico del ejemplo 23

También es de destacar que, gracias tanto al teclado matricial de 4x4 como al interface de 4 bits con la pantalla LCD, es posible gestionar ambos periféricos consumiendo tan solo 11 líneas de E/S del PIC. Este es un detalle importante ya que ambos periféricos nos pueden solucionar el interface humano con múltiples aplicaciones, a un costo relativamente pequeño de líneas de E/S.

3.3.24 Ejemplo 24: Interrupción por cambio de estado

Objetivos

Se trata de un ejemplo similar al anterior en cuanto a que sobre la pantalla LCD se visualiza la tecla pulsada. La diferencia consiste en que la exploración del teclado sólo se realiza cuando se detecte una interrupción por cambio de estado en las líneas RB4-RB7 como consecuencia de pulsar cualquier tecla.

Esquema

El mismo que se empleó en el ejemplo anterior.

Comentarios

Este ejemplo se basa en la capacidad que tienen las líneas de la puerta B para provocar una interrupción cuando cualquiera de ellas cambia de estado. Esta característica ya se utilizó en el ejemplo 11.

El programa principal se mantiene detenido en el modo SLEEP de bajo consumo. Cuando se pulsa cualquier tecla hay un cambio de estado y se produce una interrupción que “despierta” al PIC. El programa de tratamiento correspondiente se encarga de determinar qué tecla se pulsó para, posteriormente, visualizarla sobre el LCD. Se trata de una muy interesante aplicación. En el ejemplo anterior el PIC estaba en ejecución constante esperando que se pulse una tecla. En este ejemplo el PIC se mantiene en standby y sólo reacciona cuando tiene lugar una pulsación.

3.3.25 Ejemplo 25: Teclado alfanumérico

Objetivos

Este ejemplo pretende ser un compendio de los anteriores. Se trata de controlar el teclado matricial de 4x4 y la pantalla LCD. En esta ocasión el teclado se gestiona como si de un teclado alfanumérico se tratara de forma que una tecla pueda tener asignada diferentes caracteres.

Esquema

El mismo que se empleó en los ejemplos anteriores.

Comentarios

Se trata de un ejemplo de cierta complejidad pero queda justificado por las numerosas aplicaciones posibles que se pueden desarrollar a partir del mismo. Se trata ni más ni menos que de emplear el teclado de 4x4 como si de un teclado alfanumérico se tratara.

Efectivamente, al igual que sucede con los teclados de los teléfonos móviles y otros artículos similares, una misma tecla puede tener asignada diferentes dígitos o símbolos. Por ejemplo, en nuestro teclado, la tecla 2 tiene asignados los caracteres A, B y C.

Cuando se pulsa una tecla cualquiera por primera vez, se obtiene el símbolo principal. Si esa misma tecla se pulsa repetidamente en breves espacios de tiempo, se van obteniendo secuencialmente los restantes símbolos.

En el ejemplo todo lo que se vaya tecleando se irá visualizando sobre la pantalla LCD. Si se estudia con detenimiento (y con paciencia) el programa fuente, se podrá observar que se hace uso de muchas de las cosas que hemos ido viendo en anteriores ejemplos: tablas con constantes, el TMR0, interrupciones por cambio de estado, etc...

En fin, se trata de un buen ejemplo que crea las bases para hacer lo que sea que necesitemos, en cuanto a la gestión de unos periféricos tan importantes en el mundo real como son la pantalla y el teclado.

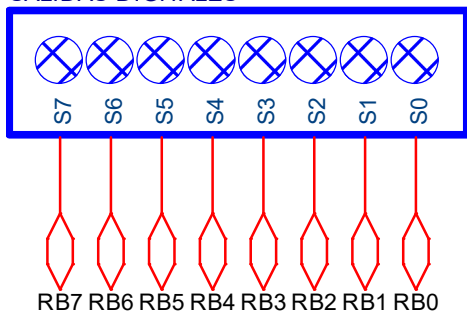
3.3.26 Ejemplo 26: El TMR1 en modo temporizador, interrupción cada cierto tiempo

Objetivos

Se trata de que el TMR1 provoque una interrupción cada 0.1". Un contador binario reflejado mediante los leds S0-S7 conectados a la puerta B, representa en binario el número de interrupciones producidas.

Esquema

SALIDAS DIGITALES



Conectar los leds S0-S7 a las líneas RB0-RB7 de la puerta B, tal y como se muestra en el esquema de la figura 3-39.

Figura 3-39. Conexiones del ejemplo 26.

Comentarios

El ejemplo pretende introducirnos en el manejo del TMR1, sus posibilidades y su configuración. Básicamente el TMR1 consiste en un contador ascendente de 16 bits y dispone de un preescalador de hasta 8. Dispone de diferentes modos de trabajo algunos de los cuales se verán en éste y próximos ejemplos. En esta ocasión se configura para que actúe en el modo temporizador y provoque una interrupción cada 0.1". El número de interrupciones producidas se verá reflejado en formato binario sobre los leds de salida.

3.3.27 Ejemplo 27: El TMR1 como contador de eventos externos

Objetivos

Es otro modo de trabajo del TMR1. Mediante el generador de onda cuadrada del laboratorio USB-PIC'School se aplican pulsos por la línea RC0/T1CKI. La pantalla LCD visualiza en binario el número de pulsos que entran al TMR1.

Esquema

Las líneas RB0-RB3 se conectan con las líneas de datos L4-L7 del LCD. Las señales E, R/W y RS se conectan con RA1-RA3 respectivamente. La salida del generador se conecta con la línea RC0/T1CKI de entrada de pulsos para el TMR1 y también con el led S0 que pilota dicha salida. Ver el esquema de la figura 3-40.

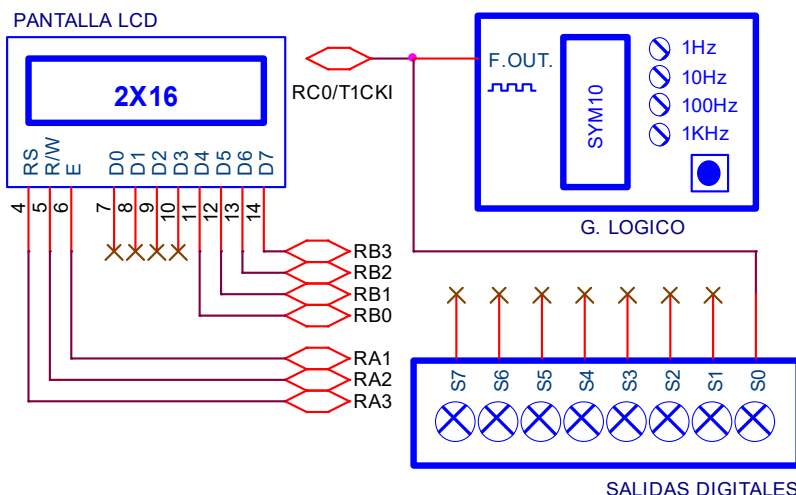


Figura 3-40. Conexiones para el ejemplo 27

Comentarios

Al igual que le TMR0, el TMR1 también puede contar pulsos de entrada con flanco ascendente y que se aplican por la patilla RC0/T1CKI. El generador que dispone el laboratorio USB-PIC'Scholl se encarga de suministrar los pulsos a contar.

Se recomienda variar la frecuencia de dicho generador (entre 1Hz y 1KHz) para apreciar la variación que se produce en el conteo y apreciar, gracias al led S0, cómo las evoluciones del contador se realizan en el flanco ascendente de la señal.

3.3.28 Ejemplo 28: El TMR1 como contador de eventos externos con control de disparo

Objetivos

Es un ejemplo similar al anterior. Se trata de contar el número de pulsos externos que se le aplican al TMR1 y visualizar la cuenta (en binario) sobre la pantalla LCD. En esta ocasión habilitamos la capacidad de disparo que permite detener o reanudar la cuenta del TMR1 mediante una señal exterior.

Esquema

Se muestra en la figura 3-41. Es idéntico al empleado en el ejemplo anterior al que además se le añade la conexión entre RB5 y el interruptor E0 a través del cual se realiza el control de la función de disparo del TMR1.

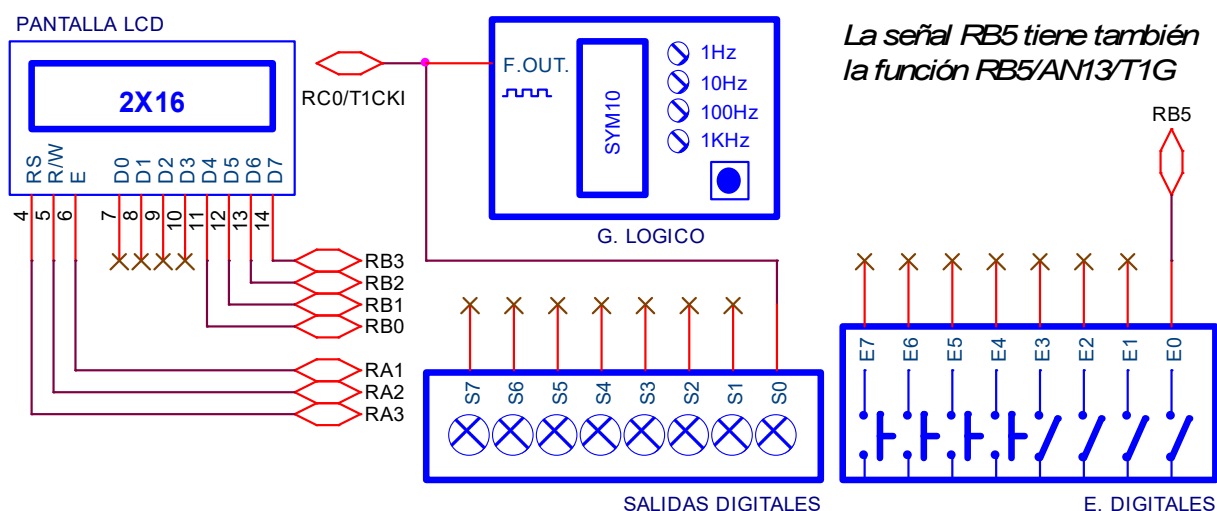


Figura 3-41. Esquema de conexiones para el ejemplo 28

Comentarios

Una característica añadida en la nueva familia PIC16F88X es la posibilidad de controlar el funcionamiento del TMR1 mediante una señal externa que activa o desactiva el conteo sin intervención alguna del software de la aplicación. Esta señal recibe el nombre de *señal de disparo T1G* y se aplica por la línea multifunción RB5/AN13/T1G configurada como entrada.

3.3.29 Ejemplo 29: El TMR1 como divisor de frecuencia

Objetivos

Una aplicación directa del TMR1 como contador de eventos externos. El TMR1 cuenta el número de pulsos externos que le entran por RC0/T1CKI y que se determina en la variable "Valor_N" del programa. Cuando se alcanza ese número finaliza la cuenta y se provoca una interrupción que hace cambiar de estado a la línea de salida RB0. La frecuencia en esta línea de salida será por tanto la del generador externo que aplica los pulsos dividida entre $2 * \text{Valor_N}$ (variable del programa).

Esquema

Realizar las conexiones mostradas en la figura 3-42. La salida del generador del laboratorio se conecta con la entrada de pulsos externos para el TMR1 RC0/T1CKI. La línea RB0 se conecta con el led de salida S0 que cambiará de estado cada vez que el número de pulsos aplicados al TMR1 alcancen el valor prefijado.

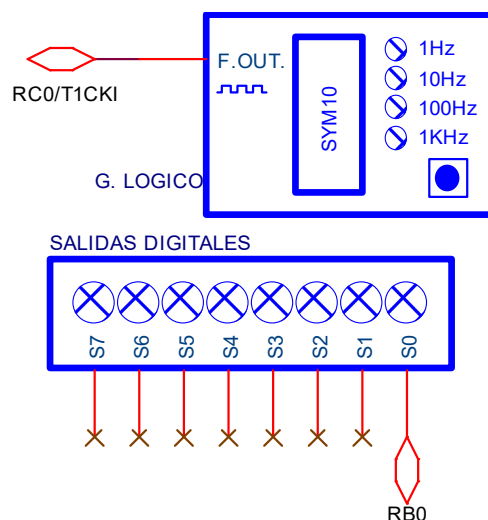


Figura 3-42. Conexiones del ejemplo 29

Comentarios

Este sencillo ejemplo nos muestra la capacidad que tiene TMR1 de provocar interrupción cuando trabaja en el modo contador de eventos externos. Un detalle importante es que el PIC puede estar en el modo SLEEP de bajo consumo. Efectivamente, como el TMR1 está trabajando como contador de eventos externos, no necesita de la frecuencia general del reloj del sistema y puede provocar

interrupción al llegar a 0. Se sugiere modificar el programa variando el valor de la variable *Valor_N* con objeto de obtener otras divisiones de frecuencia. Igualmente se puede variar la frecuencia de entrada procedente del generador, entre 1Hz y 1 KHz.

3.3.30 Ejemplo 30: Frecuencímetro

Objetivo

Se trata de mostrar otra aplicación directa del TMR1 en el modo de contador de eventos externos. El ejemplo propone realizar un frecuencímetro en base al número de pulsos que entran al TMR1 durante un intervalo de tiempo. La frecuencia obtenida se visualiza sobre la pantalla LCD.

Esquema

Conectar la pantalla LCD como se muestra en el esquema de la figura 3-43, tal y como se venido realizando hasta ahora. La salida del generador lógico se conecta con la línea RC0/T1CKI de entrada de pulsos externos al TMR1.

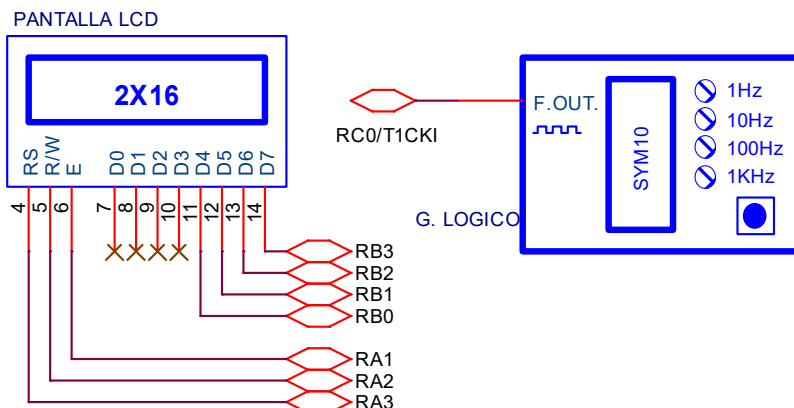


Figura 3-43. Montaje del frecuencímetro

Comentarios

Este es un buen ejemplo de trabajo conjunto entre el TMR0 y el TMR1. El TMR1 se configura para trabajar como contador de pulsos externos. Por su parte el TMR0 provoca una interrupción cada 10mS. Cuando se han producido 100 interrupciones habrá transcurrido 1 segundo, en este momento el TMR1 deja de contar. En otras palabras, el TMR1 cuenta los pulsos que le entran durante un intervalo de 1 segundo. El resultado de la cuenta representa por tanto los ciclos/segundo (Hz) de la señal de entrada y se visualizan en el LCD. Posteriormente se inicia una nueva medida.

3.3.31 Ejemplo 31: El TMR2, interrupción periódica

Objetivo

Mediante el empleo del TMR2 se trata de hacer un segundero que se visualiza en binario sobre unos leds conectados a la puerta B.

Esquema

Realizar las conexiones mostradas en el esquema de la figura 3-44. Las salidas RB0-RB7 controlan los segmentos a-dp de los displays de 7 segmentos. Las salidas RA2 y RA3 controlan los ánodos de los displays de unidades y decenas.

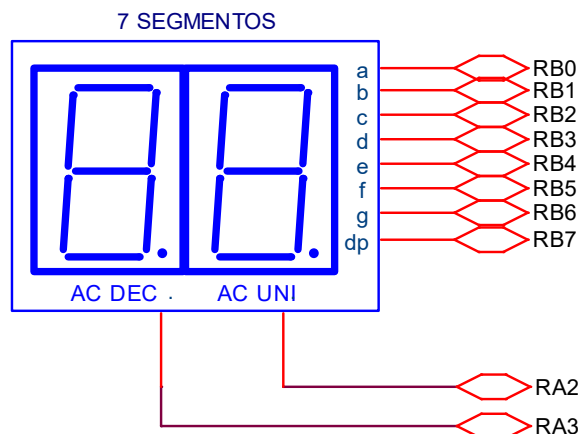


Figura 3-44. Conexiones del ejemplo 31

Comentarios

Se pretende mostrar el funcionamiento del TMR2. Para ello se provoca una interrupción fija cada 10mS que se repite 100 veces para obtener el tiempo total deseado (1 seg.) Una diferencia importante de este timer frente a los anteriores es que posee un registro de periodos que permite la recarga automática del TMR2 cada vez que éste finaliza la cuenta. Este registro determina por lo tanto la temporización a realizar en cada caso. También dispone de un preescaler y un postcaler

Efectivamente, el registro de periodos se carga con el valor 39. El TMR2 con un preescaler de 1:16 evoluciona cada 16 uS trabajando a 4KHz ($4T_{osc} \cdot 16$). Cuando la evolución alcanza el valor del registro de periodos (39) habrá transcurrido un lapsus de 624 uS. En ese instante el TMR2 se pone a 0 y vuelve a reanudar la cuenta.

Mediante el postcaler del TMR2 ajustado a 1:16, se espera a que ese evento de 624 uS se repita 16 veces antes de provocar la interrupción propia del TMR2. Habrá transcurrido un total de 9984 uS.

El programa de tratamiento espera a que se repitan 100 interrupciones de 9984 uS cada una, haciendo un total de 1 segundo. En este momento el segundero se incrementa en decimal y se visualiza sobre los displays.

3.3.32 Ejemplo 32: El TMR2, temporizaciones ajustables

Objetivo

Se trata de obtener por la salida RB0 una señal de frecuencia variable en función de los interruptores de entrada conectados a RA0-RA5. Para ello se emplea el TMR2 cuyo registro de periodos se verá modificado dinámicamente con el valor binario de esas entradas.

Esquema

Tal y como se muestra en el esquema de la figura 3-45 los cuatro interruptores E0-E3 y los dos pulsadores E4 y E5 se conectan con las líneas de entrada RA0-RA5. Por su parte la línea RB0 actúa como salida y se conecta con el led E0. Este cambiará de estado a intervalos de tiempo (o periodo) variable.

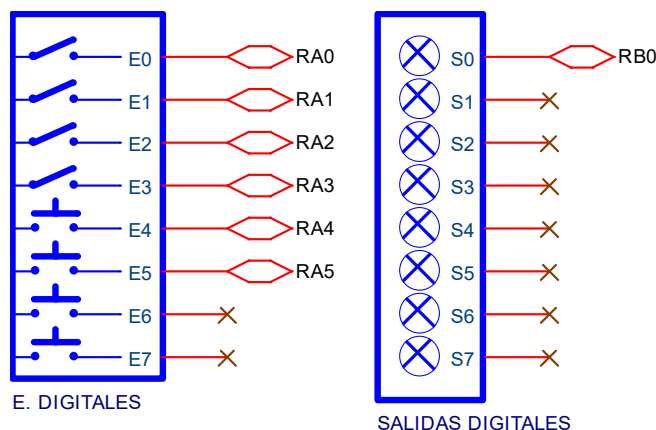


Figura 3-45. Conexiones del ejemplo 32

Comentarios

El programa lee el valor binario que introducen los interruptores E0-E3 y los pulsadores E4 y E5 a través de las entradas RA0-RA5, lo multiplica por 4 y lo almacena en el registro de periodos. De esta forma se consigue variar la temporización del TMR2 en función de ese valor. El TMR2 con un preescaler de 1:16 evoluciona cada 16 uS hasta alcanzar el valor del registro de periodos $(RA5:RA0) \cdot 4$, momento en que pasa a 0 y se reanuda una nueva cuenta. Si ese valor varía, también lo hará la cuenta. Gracias al postcaler de 1:16 se deben producir 16 de esas nuevas cuentas para que se produzca la interrupción, lo que hace un tiempo total de:

$$\text{Interrupción} = (16\mu S \cdot ((RA5:RA0) \cdot 4)) \cdot 16$$

La salida RB0 cambia de estado cuando se hayan producido 10 interrupciones con lo que el periodo será de:

$$\text{Periodo} = ((16\mu S \cdot ((RA5:RA0) \cdot 4)) \cdot 16) \cdot 10$$

En la siguiente tabla se resumen, a modo de ejemplo, unos cuantos periodos y frecuencias en RB0 a partir del valor de RA5:RA0:

RA5:RA4	Interrupción	Periodo en RB0	Frecuencia en RB0
000000 =0	256 uS	2560 uS (2.5 mS)	390 Hz
000001 =1	1024 uS	10240 uS (10.2 mS)	96.65 Hz
000100 =4	4096 uS	40960 uS (40.9 mS)	24.41 Hz
100000 =32	32768 uS	327680 uS (327.6 mS)	3.05 Hz
111111 =63	64512 uS	645120 uS (645.1 mS)	1.5 Hz

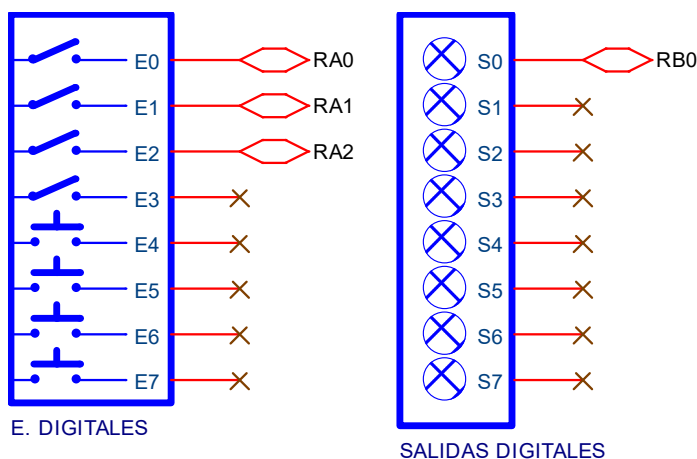
3.3.33 Ejemplo 33: El oscilador interno

Objetivos

Mostrar las posibilidades de las diferentes opciones del oscilador principal del sistema en base al cual se realizan todas las operaciones internas del controlador. Estas opciones de oscilador es otra de las mejoras propias de la familia PIC16F88X que no existían en la predecesora familia PIC16F87X.

El ejemplo trata de cambiar de estado la salida RB0 cada 500000 ciclos máquina. La velocidad con la que se ejecutan esos ciclos máquina dependerá del tipo de oscilador principal elegido y que se puede modificar dinámicamente mediante las entradas RA2:RA0.

Esquema



La figura 3-46 muestra las conexiones a realizar para implementar el presente ejemplo. Basta conectar los interruptores E0-E2 con las líneas RA0-RA2 respectivamente.

El led de salida S0 se conecta con la línea de salida RB0.

Figura 3-46. Conexiones para el ejemplo 33

Comentarios

Con este ejemplo se pretende ilustrar la facultad que tienen los dispositivos PIC16F88X para alterar dinámicamente, en ejecución, la frecuencia del oscilador principal del sistema.

El ejemplo como tal es bastante sencillo. Se trata de hacer que la salida RB0 cambie de estado cada un cierto número fijo de ciclos máquina, concretamente unos 500000. Hasta ahora la velocidad de trabajo del micro era constante de 4MHz producidos por el oscilador externo del laboratorio USB-PIC'School. Con este dato sabemos que cada ciclo máquina se ejecuta en 1 uS (4Tosc), por lo que en este caso la salida RB0 cambiaría cada 500000 uS (0.5 Seg.).

Ahora bien, los dispositivos PIC16F88X disponen de un oscilador interno que se activa mediante el bit SCS del registro OSCCON. En este momento el oscilador externo del laboratorio queda inhabilitado. A su vez ese oscilador interno tiene la posibilidad de generar hasta 8 frecuencias de trabajo diferentes que van desde los 8MHz (más alto que el propio oscilador externo del laboratorio) hasta los 32KHz (muy lento pero de muy bajo consumo).

Existen tres bits en el registro OSCCON que permiten seleccionar de entre uno de esos 8 valores de oscilador interno. Pues bien, en nuestro programa de ejemplo esos bits se cargarán dinámicamente (en tiempo de ejecución) con el valor actual procedente de las líneas RA2:RA0 conectadas con los interruptores E2:E0.

Si se selecciona una alta velocidad interna de trabajo (p.ej. 8MHz), los 500000 ciclos máquina necesarios para hacer que la salida RB0 cambie de estado, se ejecutarán con mayor rapidez que si se selecciona una baja velocidad. En otras palabras, el mismo programa ejecutándose a distintas velocidades provocará notables efectos en el resultado final.

La siguiente tabla presenta un resumen de los tiempos de ejecución en función de la velocidad seleccionada:

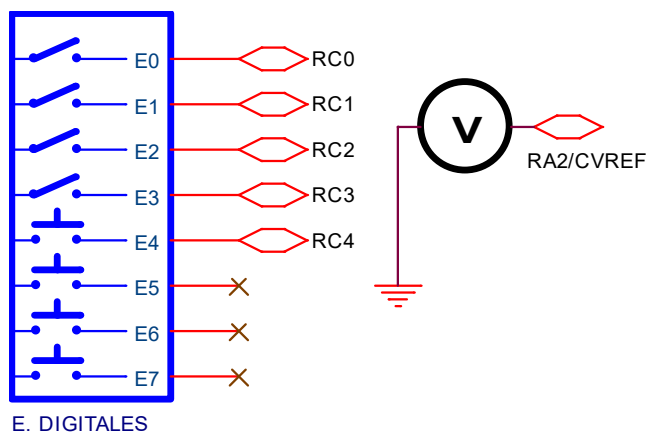
RA2:RA0	Fosc Interno	4 Tosc	RB0 cambia cada...
000	31 KHz	129 uS	64.5 seg.
001	125 KHz	32 uS	16 seg.
010	250 KHz	16 uS	8 seg.
011	500 KHz	8 uS	4 seg.
100	1 MHz	4 uS	2 seg.
101	2 MHz	2 uS	1 seg.
110	4 MHz	1 uS	0.5 seg.
111	8 MHz	0.5 uS	0.25 seg.

3.3.34 Ejemplo 34: El módulo CVREF, Generar una tensión analógica variable

Objetivos

Emplear el módulo CVREF (generador de tensión de referencia) que integran los dispositivos PIC16F88X para generar una tensión analógica variable con salida por RA2/CVREF. El valor de la tensión de salida dependerá del valor binario de las entradas RC4:RC0 conectadas con los interruptores E4:E0.

Esquema



Tal y como se muestra en la figura 3-47 las líneas RC0-RC4 se conectan con los interruptores E0-E3 y el pulsador E4 respectivamente.

Para medir la tensión de salida basta usar un sencillo voltímetro conectado entre RA2/CVREF y tierra.

Figura 3-47. Conexiones del ejemplo 34

Comentarios

Otra de las mejoras que integran los PIC16F88X frente a sus predecesores PIC16F87X. El módulo CVREF consiste en un convertidor D/A de 4 bits y es capaz de generar una tensión analógica variable que se puede emplear como tensión de referencia para otros módulos integrados en el PIC o simplemente como una tensión analógica que se obtiene por la línea RA2/CVREF y que se puede usar en múltiples aplicaciones.

Dispone de un registro, el VRCON, con el que se configura múltiples modos de funcionamiento del módulo CVREF y se selecciona el valor de la tensión a generar. Efectivamente, dispone de 4 bits para seleccionar un valor de entre 16 posibles y otro bit para seleccionar entre dos rangos: el alto y el bajo. En nuestro programa de ejemplo esos 5 bits se modificarán dinámicamente (en tiempo de ejecución) con el valor binario que suministran las líneas RC4:RC0. Las entradas RC3:RC0 se conectan con los interruptores E3:E0 del laboratorio y aplican el valor binario de 4 bits con el que se genera la tensión de salida. La entrada RC4 se conecta con el pulsador E4 y permite seleccionar el rango alto o bajo de trabajo.

En el rango alto (RC4=0) la tensión de salida CVREF queda definida por la siguiente ecuación:

$$CVREF = \frac{Vdd}{4} + \left(\frac{RC3:RC0}{32} \times Vdd \right)$$

Suponiendo que Vdd=5V el valor mínimo será cuando RC3:RC0 = 0000 (0) → 1.25 V

Suponiendo que Vdd=5V el valor máximo será cuando RC3:RC0 = 1111 (15) → 3.593V

En el rango bajo (RC4=1) la tensión de salida CVREF queda definida por la siguiente ecuación:

$$CVREF = \left(\frac{RC3:RC0}{24} \right) \times Vdd$$

Suponiendo que Vdd=5V el valor mínimo será cuando RC3:RC0 = 0000 (0) → 0 V

Suponiendo que Vdd=5V el valor máximo será cuando RC3:RC0 = 1111 (15) → 3.125 V

Como el módulo CVREF consiste en un convertidor D/A de 4 bits, entre el mínimo y máximo disponemos de una escala de 16 valores posibles.

La figura 3-48 muestra el montaje práctico del ejemplo 34 sobre el laboratorio USB-PIC'Scool. Se puede apreciar el empleo de un simple voltímetro para medir la tensión analógica de salida entre la línea RA2/CVREF y GND.

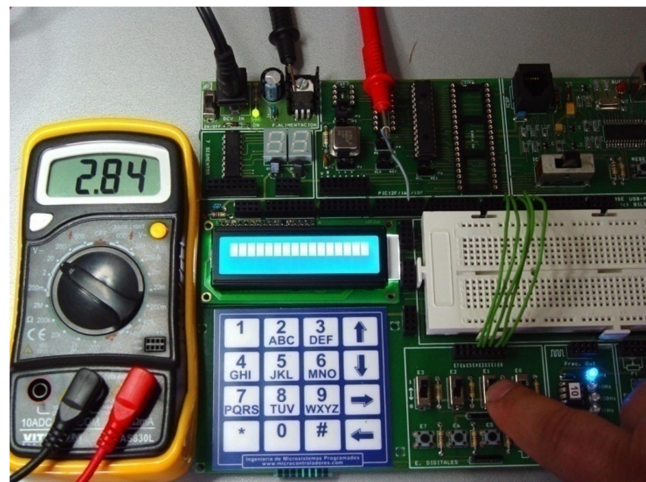


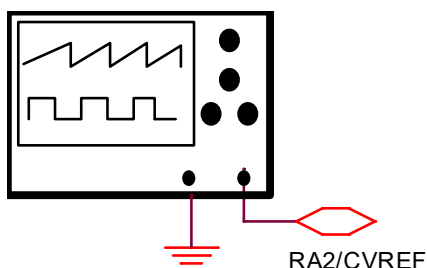
Figura 3-48. Montaje práctico del ejemplo 34

3.3.35 Ejemplo 35: El módulo CVREF, Generando una señal en diente de sierra

Objetivos

Mediante el empleo del módulo CVREF se trata de obtener una señal en diente de sierra por la salida RA2/CVREF.

Esquema



Se muestra en la figura 3-49 y básicamente se basa en conectar la línea de salida de señal RA2/CVREF con un canal del osciloscopio. Este visualizará la señal en diente de sierra de salida.

Figura 3-49. Conexión con un osciloscopio

Comentarios

En el ejemplo la variable "Valor" se va incrementando entre 0 y 16 al tiempo que se va depositando sobre el registro VRCON. De esta manera tensión de salida generada por éste irá incrementándose desde el mínimo ("Valor" = 0) hasta el máximo ("Valor" = 16) y vuelta a empezar desde 0. Recordemos que el módulo CVREF consiste en un convertidor D/A de 4 bits.

Entre un incremento de la variable “Valor” y el siguiente, se intercala una temporización o Delay. Tras 16 incrementos y temporizaciones el ciclo se repite. En el ejemplo se emplea la constante “F_Out” para calcular el delay necesario de forma que se obtenga un periodo o frecuencia fija de la señal de salida (p.e. 1000 Hz).

En la figura 3-50 se muestra la señal que se obtiene por la salida RA2/CVREF cuando se graba y ejecuta el programa de ejemplo.

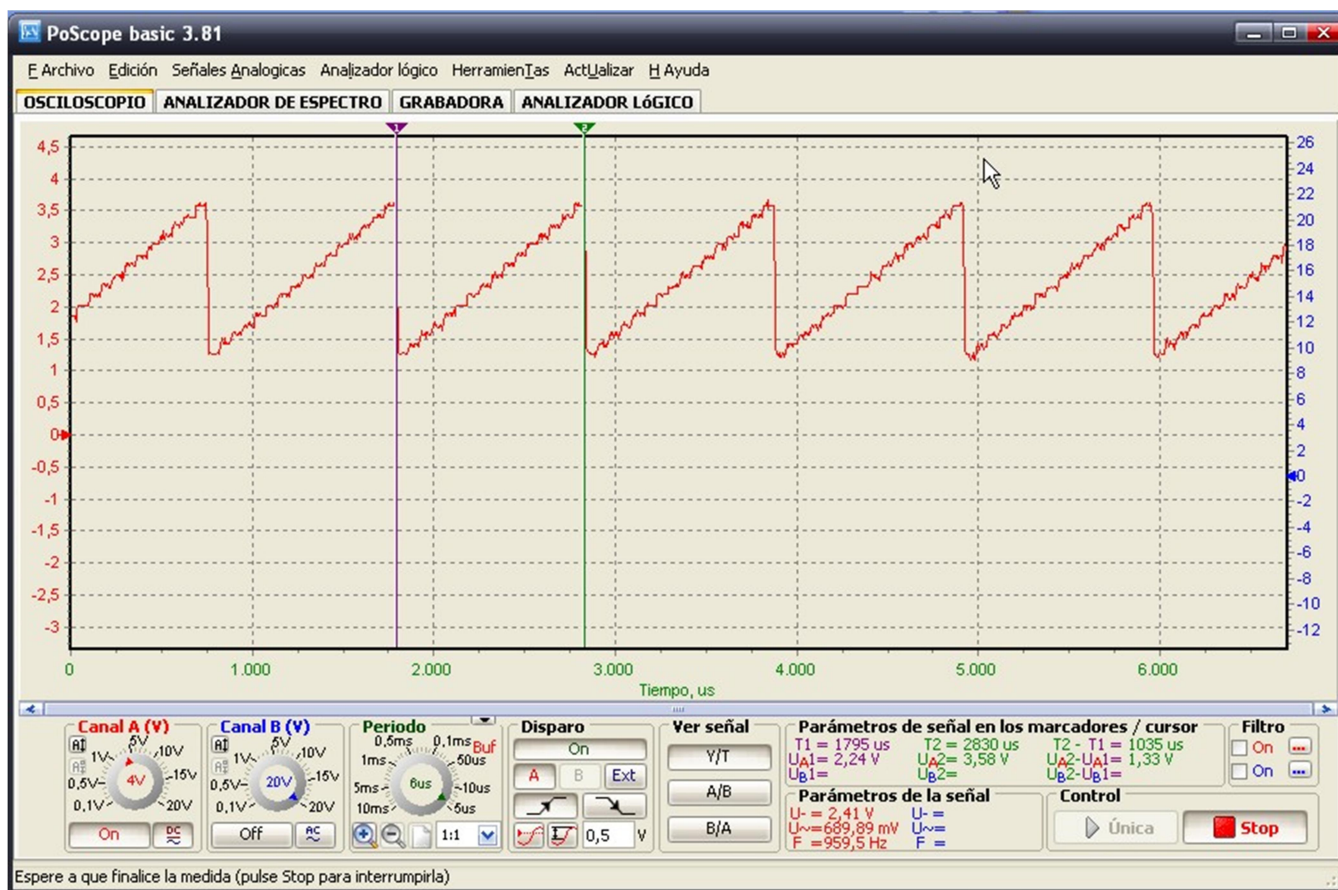


Figura 3-50. La señal de salida en diente de sierra

En el oscilograma de la figura se aprecia claramente la señal en diente de sierra con un periodo de 1035uS entre los cursores 1 y 2 (frecuencia de 1KHz). Dentro de cada periodo también se aprecian los 16 “escalones” correspondientes a las 16 etapas del convertidor D/A. Partiendo de 0 (“Valor”=0) la tensión de salida va aumentando hasta llegar al máximo (“Valor”=16). Luego “Valor” vuelve a 0, la tensión desciende bruscamente y comienza un nuevo ciclo.

Como ya se explicó en el ejemplo anterior, cuando el módulo CVREF se configura en el rango alto, la tensión de salida mínima es de 1.25V (cuando “Valor” =0 y Vdd=5V). Efectivamente, en la figura anterior se aprecia claramente esta tensión de “offset”. Con acoplamiento en DC el osciloscopio representa la señal de salida por encima de 1V en el eje vertical Y.

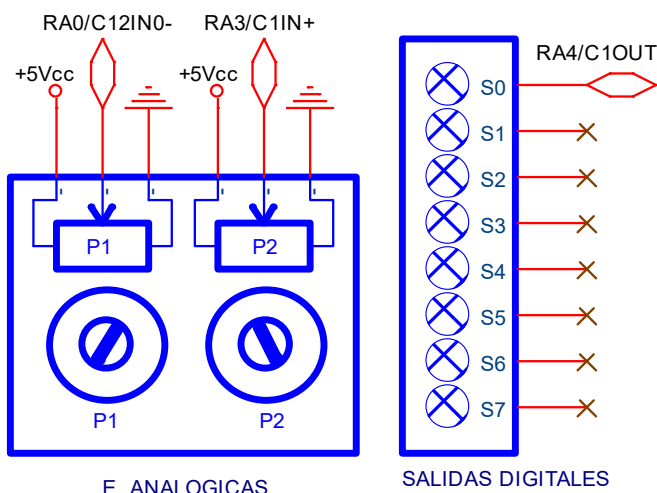
Por último indicamos que para realizar estas medidas y otras de los sucesivos ejemplos, hemos empleado el osciloscopio **PoScope basic 2** de la firma PoLabs (www.poscope.com) y de la cual somos distribuidores. Se trata de un instrumento de bajo coste, que se conecta al PC mediante uno de los puertos USB. Lo hemos elegido como herramienta de trabajo por su bajo coste y reducido tamaño y al que lo podemos conectar al mismo PC (de sobre mesa o portátil) con el que estamos realizando los presentes ejemplos.

3.3.36 Ejemplo 36: El módulo comparador, comparando dos tensiones analógicas

Objetivos

Utilizar el módulo comparador para comparar dos tensiones analógicas independientes que se introducen por RA0/C12IN0- y por RA3/C1IN+. La salida RA4/C1OUT se conecta a un led que se activará cuando la tensión de RA3/C1IN+ sea mayor que la de RA0/C12IN0-

Esquema



Tal y como se muestra en el esquema de la figura 3-51 ambas tensiones analógicas se generan mediante los dos potenciómetros del laboratorio USB-PIC'School y estarán comprendidas entre 0 y 5V. P1 permite ajustar la tensión analógica que se aplica a la entrada RA0/C12IN0- del comparador y P2 a la entrada RA3/C1IN+.

La salida del comparador se realiza por la línea RA4/C1OUT y se conecta con el led S0.

Figura 3-51. Esquema del ejemplo 36

Comentarios

Otra de las ventajas añadidas en la familia PIC16F88X frente a su predecesora PIC16F87X, es que integra un módulo comparador de tensiones analógicas con dos comparadores independientes entre sí. En este ejemplo usamos al comparador C1.

Según cómo se le configure, el comparador C1 puede comparar dos tensiones analógicas que se pueden aplicar desde múltiples fuentes. En el presente ejemplo la entrada invertida se aplica por la línea RA0/C12IN0- y la entrada no invertida por la línea RA3/C1IN+.

La salida del comparador se puede configurar para disparar diferentes eventos internos y también se puede obtener por la línea RA4/C1OUT que es como se usa en el ejemplo. Esta salida se activa cuando la tensión de la entrada no invertida aplicada por RA3/C1IN+ es mayor que la aplicada por RA0/C12IN0-.

Los comparadores son independientes entre sí y también respecto al resto del microcontrolador, es decir, no necesita atención alguna por parte del firmware. Efectivamente, una vez que se le activa y configura, el controlador puede dedicarse a otras tareas sin prestarle mayor atención. Si se desea, puede provocar interrupción cada vez que se active la salida. En el caso del ejemplo que nos ocupa obsérvese que el controlador simplemente queda en el modo Sleep de bajo consumo aunque el comparador sigue con su tarea.

Podemos emplear un voltímetro para medir las tensiones de entrada y contrastarlas con el resultado que ofrece el comparador a su salida.

3.3.37 Ejemplo 37: El módulo comparador, comparando una tensión analógica con otra fija de referencia

Objetivos

Se trata de emplear el comparador C1 para comparar dos tensiones: una tensión analógica que se introduce por RA0/C12IN0- con otra tensión de referencia que la genera el módulo CVREF. Es decir, tenemos un comparador con tensión de referencia ajustable.

Esquema

Se muestra en la figura 3-52.

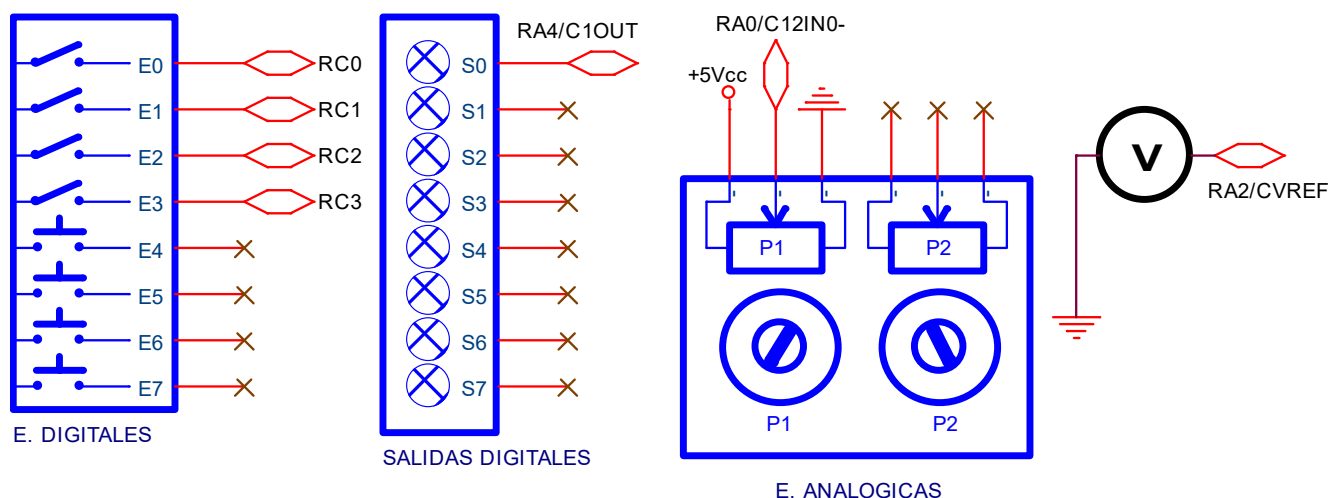


Figura 3-52. Esquema del ejemplo 37

El potenciómetro P1 se conecta con la entrada invertida RA0/C12IN0- del comparador y aplica una tensión analógica comprendida entre 0 y 5V. Por su parte el módulo CVREF del PIC se encarga de generar la tensión de referencia no invertida del comparador. El valor de esta tensión de referencia se puede modificar digitalmente mediante las líneas RC0:RC3 conectadas con los interruptores E0-E3. La tensión de referencia con la que estamos comparando se puede medir con un sencillo voltímetro conectado en RA2/CVREF.

Finalmente la salida del comparador se obtiene por la línea RA4/C1OUT y se conecta al led S0.

Comentarios

Este es un buen ejemplo en el que se emplea de forma combinada el comparador C1 del módulo comparador y el módulo generador de tensión de referencia CVREF que ya se utilizó en anteriores ejemplos.

El módulo generador de tensión de referencia CVREF se usa como un convertidor D/A de 4 bits. La tensión de salida CVREF se obtiene por RA2/CVREF, y se emplea como entrada de tensión de referencia para el módulo comparador C1. Esta tensión variará en función de las entradas RC3:RC0, con lo que se obtiene una tensión de referencia de 16 valores posibles y que en todo momento se puede medir en la patilla RA2/CVREF mediante un voltímetro.

Esa tensión de referencia se compara con la tensión analógica de entrada presente en RA0/C12IN0- y procedente de uno de los potenciómetros del laboratorio. La salida RA4/C1OUT se activa cuando la tensión en RA0/C12IN0- sea mayor que la tensión de referencia ajustada en ese momento.

Hacemos notar que el programa principal simplemente se encuentra en bucle cerrado en el que se leen las entradas RC0:RC3 para actualizar el registro VRCON y con ello ajustar la tensión de referencia empleada en la comparación.

3.3.38 Ejemplo 38: El módulo comparador, comparador con tensión de referencia ajustable

Objetivos

Consiste en un ejemplo similar al anterior. Comparar una tensión analógica con otra de referencia ajustable. La pantalla LCD visualiza el valor actual de la tensión de referencia y si es mayor o no que la analógica.

Esquema

Se muestra en la figura 3-53 y es muy parecido al que se empleó en el ejemplo anterior. Las entradas RC0:RC3 permiten ajustar el valor de la tensión de referencia del generador CVREF. La tensión analógica a comparar se introduce por RA0/C12IN0-. Finalmente la pantalla LCD se conecta como lo hemos venido haciendo en anteriores ejemplos

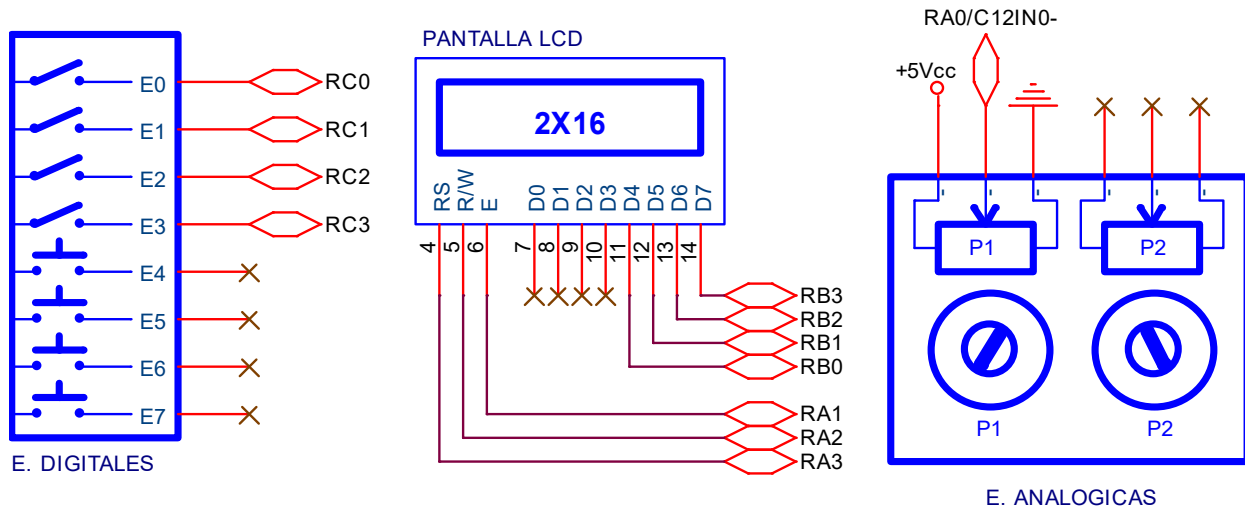


Figura 3-53. Esquema de conexiones del ejemplo 38

Comentarios

Funcionalmente este ejercicio es muy similar el del ejemplo anterior. Básicamente hay dos diferencias. En primer lugar tanto la tensión de referencia como la salida del comparador se usan internamente. Es decir, las líneas RA2/CVREF y RA4/C1OUT que anteriormente se empleaban como salida de tensión de referencia y salida del comparador respectivamente, ahora se pueden emplear como E/S salidas de propósito general.

En segundo lugar empleamos la pantalla LCD como interface de usuario. En ella se visualiza la tensión de referencia actual (en función de la tensión del estado de RC0:RC3) y si es mayor o menor que la tensión analógica de entrada que estamos comparando.

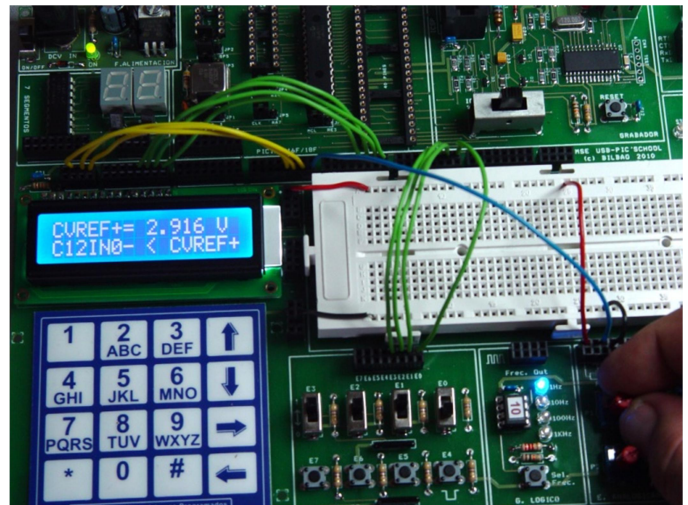


Figura 3-54. Montaje práctico y ejecución en C del ejemplo

La versión del ejemplo escrito en ensamblador simplemente se visualiza el valor de RC0:RC3 al que hay que dividir entre 24 y multiplicar por 5 para conocer exactamente el valor de la tensión de referencia actual, según el estado de RC3:RC0:

$$CVREF = \left(\frac{RC3:RC0}{24} \right) \times Vdd$$

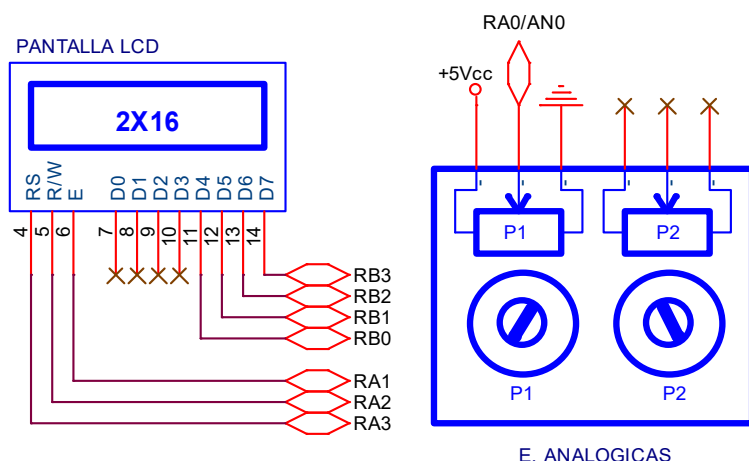
La versión del ejemplo escrito en C hace uso de la capacidad de este lenguaje para realizar cálculos matemáticos. De esta manera la pantalla LCD visualiza directamente el valor de la tensión de referencia actual que se está empleando en la comparación. Ver la figura 3-54.

3.3.39 Ejemplo 39: El módulo convertidor ADC

Objetivos

Proporcionar una primera toma de contacto con el convertidor ADC que integran los dispositivos PIC16F88X. Se trata de visualizar en binario, sobre la pantalla LCD, el valor digital equivalente a la tensión analógica que se aplica por la entrada RA0/AN0.

Esquema



Tal y como se muestra en la figura 3-55 la pantalla LCD se conecta como de costumbre.

Por su parte el potenciómetro P1 del laboratorio USB-PIC'School se conecta con la línea RA0/AN0 que se configurará como entrada analógica.

Figura 3-55. Conexiones del ejemplo 39

Comentarios

Los dispositivos PIC16F88X disponen de un convertidor A/D de 10 bits de resolución y 11 o 14 canales de entrada analógica. La tensión de referencia determina la resolución por bit: ($\text{Res.} = V_{\text{ref}}/1024$). Por ejemplo, con $V_{\text{ref}}=5 \rightarrow \text{res.} = 4.8 \text{ mV/Bit}$; con $V_{\text{ref}}=2.5\text{V} \rightarrow \text{res.} = 2.4 \text{ mV/Bit}$. En los ejemplos propuestos se emplea la propia Vdd (+5V) de alimentación como tensión de referencia.

El ejemplo, a pesar de su sencillez, resume claramente los pasos necesarios para realizar una conversión:

1. Mediante los registros ANSEL y ANSELH se selecciona qué líneas de E/S actuarán como líneas analógicas o digitales. En los ejemplos vistos hasta ahora las líneas de E/S se empleaban básicamente como líneas de E/S digitales. En esta ocasión RA0/AN0 se configura como línea analógica.
2. Por coherencia aquellas líneas que hayan sido configuradas como analógicas, se deben también configurar como entradas mediante los correspondientes registros TRIS. En nuestro ejemplo RA0/AN0 se configura como entrada.
3. Con los registros ADCON0 y ADCON1 se determina la configuración y funcionamiento del propio módulo convertidor. Una de las tareas más frecuentes a realizar es la selección de qué entrada analógica, de todas las que haya, deseamos hacer la conversión. Debemos recordar que aunque hayamos configurado varias líneas como entradas analógicas, convertidor sólo hay uno. Sólo se puede hacer la conversión de una entrada analógica cada vez.
4. El bits GO de ADCON0 y el ADIF de PIR1 permiten realizar una nueva conversión de la entrada analógica seleccionada en ese instante y determinar cuándo una conversión ha finalizado.

El ejemplo propone realizar la conversión de la tensión presente en el canal RA0/AN0. Esta se puede variar con el potenciómetro P1 del laboratorio. El resultado de 10 bits de la conversión se visualiza, en binario, sobre la pantalla LCD.

3.3.40 Ejemplo 40: El módulo convertidor ADC, voltímetro digital

Objetivos

Una de las primeras aplicaciones que se nos ocurre es realizar un voltímetro digital. En la pantalla LCD se visualiza el valor de la tensión analógica de entrada por RA0/AN0.

Esquema

Para realizar este ejemplo implementaremos el esquema que se muestra en la figura 3-56. Vemos que la pantalla LCD se conecta como en otras ocasiones. El potenciómetro P1 se encarga de suministrar en RA0/AN0 la tensión analógica a medir. Esta está comprendida entre 0 y 5V. También podemos colocar un voltímetro en RA0/AN0 para contrastar el valor que aparece en la pantalla LCD con el que ofrece el propio instrumento.

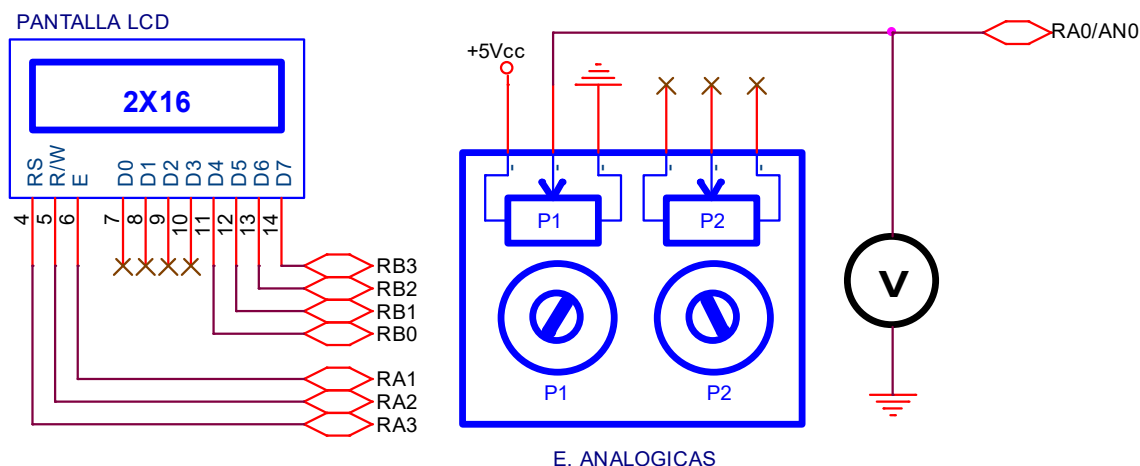


Figura 3-56. Esquema de conexiones del ejemplo 40

Comentarios

La versión de este ejemplo escrita en ensamblador nos presenta en pantalla un valor entre 0 y 1023 correspondiente a la tensión de entrada por RA0/AN0. A ese valor hay que multiplicarlo por la resolución de 0.0048V para obtener el voltaje real. Este debe coincidir con lo indicado por el voltímetro externo.

Sin embargo, con la versión de este mismo ejemplo escrito en C, nos aprovechamos de la potencia de este lenguaje para realizar cálculos aritméticos. El producto del resultado de la conversión por 0.0048 de resolución lo realiza el propio programa. En la pantalla LCD tendremos la tensión real que entra por RA0/AN0 y que debe coincidir con la medida por el voltímetro.

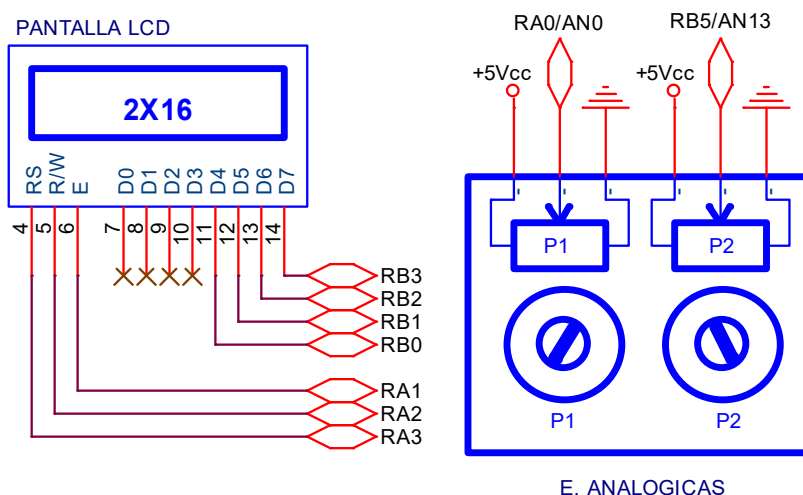
3.3.41 Ejemplo 41: El módulo convertidor ADC, multiplexando entradas analógicas

Objetivos

Proporcionar las bases para realizar conversiones múltiples procedentes de dos o más líneas o canales analógicos de entrada. El ejemplo visualiza en la pantalla LCD el valor de las tensiones procedentes de las líneas RA0/AN0 y RB5/AN13.

Esquema

Se muestra en la figura 3-57 y es similar al del ejemplo anterior. En esta ocasión se emplean los potenciómetros P1 y P2 para aplicar sendas tensiones analógicas por RA0/AN0 y RB5/AN13 respectivamente.



E. ANALOGICAS

Figura 3-57. Esquema del ejemplo 41

Comentarios

Sin añadir nada excepcionalmente nuevo respecto a los anteriores, este ejemplo realiza la conversión de la tensión analógica procedente de dos canales de entrada. Se aprecia claramente el proceso secuencial que se debe seguir: se selecciona una canal, se inicia la conversión, al finalizarla se almacena y/o procesa el resultado.

La versión de este ejemplo escrita en ensamblador nos presenta en pantalla un valor entre 0 y 1023 correspondiente a las tensiones en las entradas RA0/AN0 y RB5/AN13. A esos valores hay que multiplicarlos por la resolución de 0.0048 para obtener el voltaje real. Se puede conectar un voltímetro externo para contrastarlos.

Sin embargo, con la versión de este mismo ejemplo resuelto en C, nos aprovechamos de la potencia de este lenguaje para realizar cálculos aritméticos. El producto del resultado de las conversiones por 0.0048 de la resolución lo realiza el propio programa. En la pantalla LCD tendremos las tensiones reales que entran por RA0/AN0 y RB5/AN13 y que debe coincidir con la medida por el voltímetro. Ver la figura 3-58.

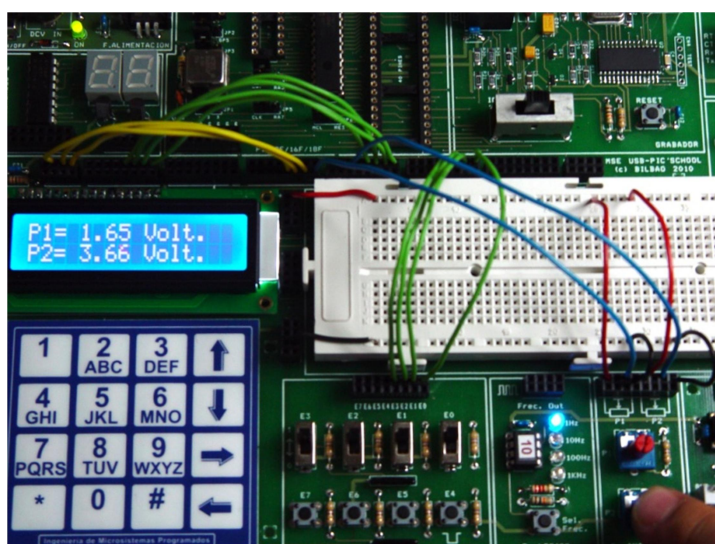


Figura 3-58. Ejecución del ejemplo en C

3.3.42 Ejemplo 42: Capturas con el módulo CCP, midiendo periodos

Objetivos

Se trata de experimentar con el modulo CCP1 en el modo de captura. Para ello se propone un programa que mide el tiempo que transcurre entre dos flancos ascendentes de la señal aplicada por RC2/CCP1. Se obtiene así el periodo de esa señal que será visualizado en la pantalla LCD.

Esquema

La pantalla LCD se conecta como se ha venido haciendo hasta ahora. Por otra parte la señal que proporciona el generador del laboratorio se conecta con la entrada RC2/CCP1. Todo ello se muestra en el esquema de la figura 3-59.

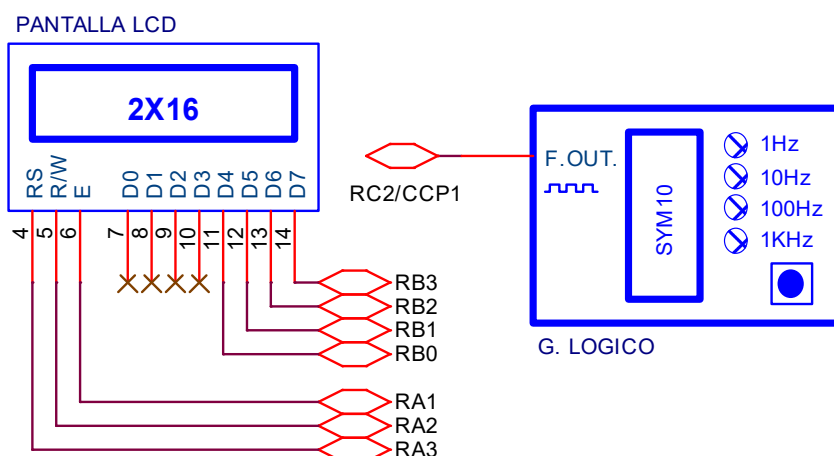


Figura 3-59. El ejemplo 42

Comentarios

En este ejemplo se pretende mostrar el funcionamiento del módulo CCP1 en el modo de captura. El TMR1 evoluciona a una frecuencia conocida. Se captura el valor que adquiere el TMR1 entre dos flancos ascendentes de la señal de entrada y que suministra el generador del laboratorio USB-PIC'School. Ese valor capturado representa el periodo de la señal. Trabajando a 4MHz y con un preescaler de 1:8, el TMR1 evoluciona cada 8µS. El valor mínimo que puede adquirir el TMR1 es de 1 lo que supone un periodo de 8µS y una frecuencia de 125Kz. El valor máximo que puede adquirir el TMR1 es de 65535, lo que supone un periodo de 0.524 seg. y una frecuencia de 2Hz. La lectura que se visualiza sobre el LCD representa el conteo realizado por el TMR1 entre un flanco ascendente y el siguiente.

En el caso del ejemplo escrito en ensamblador a la lectura que ofrece el LCD se le debe multiplicar por 8 µS (cada evolución del TMR1) para obtener el periodo final.

En el caso del ejemplo escrito en, C la pantalla LCD nos ofrece directamente el periodo medido ya que el propio programa se encarga de realizar la multiplicación por 8.

3.3.43 Ejemplo 43: Capturas con el módulo CCP1, midiendo la anchura de un pulso

Objetivo

Se pretende medir el tiempo transcurrido entre el flanco ascendente y el descendente de la señal de entrada que suministra el generador del laboratorio. Con ello se obtiene la duración del pulso de esa señal de entrada.

Esquema

El mismo que en el ejemplo anterior.

Comentarios

El ejemplo es similar al anterior. Se trata de conocer la evolución del TMR1 entre un flanco ascendente y otro descendente. Si conocemos que el TMR1 evoluciona cada 8 µS trabajando a una velocidad de 4MHz con un preescaler de 1:8, multiplicamos este valor por el valor actual del TMR1 y que aparece en el LCD. Se obtiene así la duración de la anchura del pulso.

En el caso del ejemplo escrito en ensamblador a la lectura que ofrece el LCD se le debe multiplicar por 8 μ S (cada evolución del TMR1) para obtener la anchura del pulso.

En el caso del ejemplo escrito en C la pantalla LCD nos ofrece directamente la anchura medida ya que el propio programa se encarga de realizar la multiplicación por 8.

3.3.44 Ejemplo 44: Capturas con el módulo CCP1, midiendo el tiempo entre un pulso y el siguiente

Objetivos

Se trata de medir el tiempo que transcurre entre un pulso y el siguiente de la señal de entrada aplicada por RC2/CCP1.

Esquema

El mismo que en los ejemplos anteriores

Comentarios

Este es el último ejemplo dedicado al módulo CCP1 en su modo de captura. En esta ocasión se mide el tiempo que transcurre entre el flanco descendente de un pulso y el ascendente del siguiente. Con ello conocemos el lapsus de tiempo entre dos pulsos.

En el caso del ejemplo escrito en ensamblador a la lectura que ofrece el LCD se le debe multiplicar por 8 μ S (cada evolución del TMR1) para obtener el tiempo.

En el caso del ejemplo escrito en C la pantalla LCD nos ofrece directamente el tiempo entre un pulso y otro ya que el propio programa se encarga de realizar la multiplicación por 8.

3.3.45 Ejemplo 45: Modo comparación del CCP1, contador de eventos

Objetivos

Tener una primera toma de contacto con el modo de comparación disponible en los módulos CCP de los dispositivos PIC16F88X. La salida RB0 cambia de estado cada vez que se reciben tantos impulsos externos como se indique en la constante "Pulsos"

Esquema

Se muestra en la figura 3-60. El generador lógico del laboratorio USB-PIC'School se emplea para aplicar los pulsos a contar. Se conecta con la línea RC0/T1CKI. El TMR1 actúa como contador de eventos externos.

La línea RB0 actúa como salida y se conecta al led S0. Esta salida cambiará de estado cada vez que al TMR1 le entren un determinado número de pulsos externos.

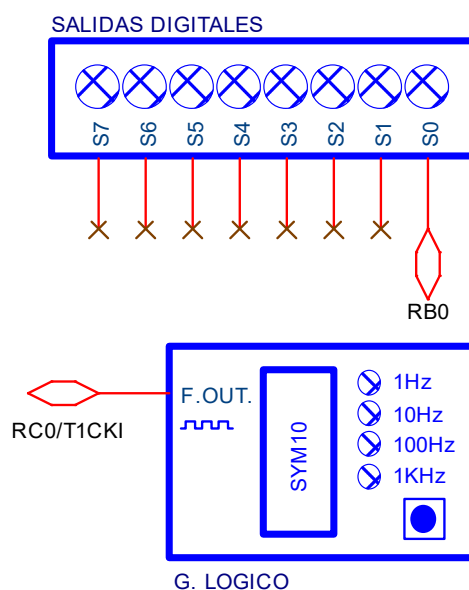


Figura 3-60. Esquema del ejemplo 45

Comentarios

El modo de comparación de los módulos CCP1 consiste básicamente en comparar constantemente el valor actual del TMR1 con el valor almacenado en un registro llamado CCPR1. Cuando los valores de ambos coinciden se dispara un evento.

En el ejemplo el registro CCPR1 se carga con el valor de la constante "Pulsos" (1000). El TMR1 se configura como contador de eventos externos que se aplican por RC0/T1CK1 y procedentes del generador del laboratorio. Cuando el TMR1 alcanza el valor de CCPR1 se provoca la interrupción CCP1 cuyo tratamiento implica un cambio de estado en la salida RB0 y una puesta a 0 del TMR1. El proceso se vuelve a repetir.

Obsérvese que el programa principal no hace prácticamente nada. Una vez que se han configurado debidamente los diferentes módulos, se mete en un bucle cerrado. El TMR1 y el sistema de comparación trabajan sin intervención del firmware (excepto cuando se produce la interrupción).

3.3.46 Ejemplo 46: Comparación con el módulo CCP1, contador de eventos ajustable

Objetivos

Se basa en el ejemplo anterior. El TMR1 cuenta tantos pulsos externos como indique el valor que se introduce desde los interruptores conectados en RA0-RA5. Cuando se alcanza el valor fijado la salida RB0 cambia de estado.

Esquema

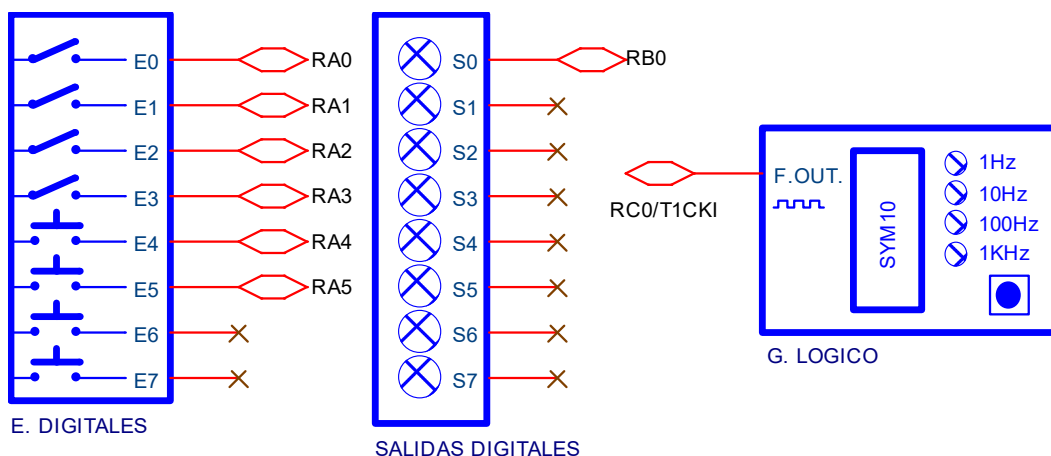


Figura 3-61. Conexiones del ejemplo 46

Comentarios

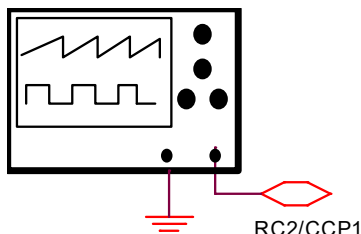
El modo de comparación de un módulo CCP consiste en dejar que el TMR1 evolucione hasta llegar al valor prefijado en los registros CCPR1H y CCPR1L. En el presente ejemplo estos registros se cargan con el valor binario que se introducen mediante las entradas E0-E5. Cuando el TMR1 alcanza ese valor se provoca una interrupción cuyo tratamiento reinicia al TMR1 y hace cambiar el estado lógico de la salida S0 conectada a RB0. El TMR1 actúa como contador de eventos externos sensibles al flanco ascendente de la señal que suministra el generador lógico del laboratorio USB-PIC'School. La salida del generador también se puede conectar a uno de los leds de salida para poder apreciar las transiciones de la señal. Se recomienda seleccionar una baja frecuencia.

3.3.47 Ejemplo 47: Modulaci3n de anchura de pulsos (PWM) mediante el m3dulo CCP1

Objetivos

Obtener por la salida RC2/CCP1 una se1al modulada en anchura. El periodo de esta se1al se determina mediante la constante “*periodo*” y la anchura del pulso o “Duty Cycle” mediante la constante “*Duty*”.

Esquema



La salida del m3dulo CCP1 se realiza por la l3nea RC2/CCP1 y por ella se obtiene la se1al modulada en anchura. Se recomienda emplear un osciloscopio para poderla visualizar, tal y como se aprecia en la figura 3-62.

Figura 3-62. Conexi3n con un osciloscopio

Comentarios

La modulaci3n de anchura de pulsos consiste en obtener una se1al de salida ajustable tanto en periodo como en la anchura del pulso. Se emplea el TMR2 y su registro de periodos PR2 as3 como el registro CCP1RL en caso de emplear el m3dulo CCP1. Cuando el TMR2 alcanza el valor de PR2 (periodo de la se1al), la salida RC2/CCP1 cambia de estado y pasa a nivel “1”. El TMR2 inicia una nueva cuenta. Cuando se alcanza el valor fijado en el registro CCPR1L (anchura del pulso) la salida RC2/CCP1 vuelve a cambiar de estado pasando ahora a nivel “0”. El TMR2 se inicia nuevamente y el ciclo se repite. Variando los valores de PR2 y CCPR1L se puede modificar la se1al de salida tanto en periodo como en anchura.

En el ejemplo se emplea el TMR2 con un preescaler 1:1 por lo que trabajando a 4MHz evoluciona cada 1μS. Conocido este dato es f1cil determinar el valor qu3 hay que cargar en PR2 y CCPR1L para obtener el periodo y la anchura.

Para comprobar el funcionamiento del presente ejemplo se recomienda el empleo de un osciloscopio que visualice la se1al presente en RC2/CCP1 y permita constatar los resultados. Tamb3n hay que destacar que el funcionamiento del m3dulo PWM es totalmente independientemente del programa en ejecuci3n. Efectivamente, una vez programado el m3dulo CCP1 y el TMR2, la se1al de salida se obtiene de forma autom1tica sin que el programa de aplicaci3n tenga que intervenir a no ser que sea para modificar los valores de PR2 y/o de CCPR1L.

Por 3ltimo indicamos que para realizar estas medidas y otras de los sucesivos ejemplos, hemos empleado el osciloscopio **PoScope basic 2** de la firma PoLabs (www.poscope.com) y de la cual somos distribuidores. Se trata de un instrumento de bajo coste, que se conecta al PC mediante uno de los puertos USB. Lo hemos elegido como herramienta de trabajo por su bajo coste, reducido tama1o, emula diferentes instrumentos y lo podemos conectar al mismo PC (de sobre mesa o port1til) con el que estamos realizando los presentes ejemplos.

3.3.48 Ejemplo 48: PWM mediante el m3dulo CCP1, variando la anchura

Objetivos

Se trata de un ejemplo similar al anterior. La diferencia consiste en que la anchura del pulso se modifica din1micamente en funci3n del valor binario que se introduce por RA0-RA5.

Esquema

Los interruptores E0-E5 se conectan con las señales RA0-RA5 de entrada, tal y como se muestra en el esquema de la figura 3-63.

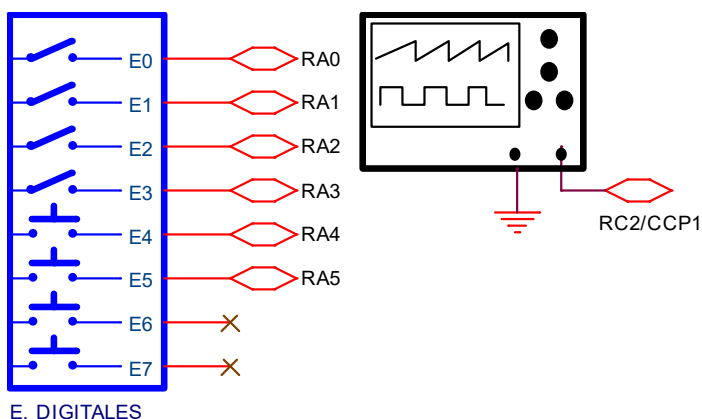


Figura 3-63. Esquema de montaje del ejemplo 48

Comentarios

El ejemplo nos muestra la posibilidad de variar la anchura del pulso de salida de acuerdo a parámetros externos como puede ser el valor binario que se introduce por las líneas RA0-RA5, conectadas a sendos interruptores/pulsadores. A una frecuencia de 4MHz y con un preescaler de 1:16, el TMR2 evoluciona cada 16 μ S. El registro de periodos PR2 se carga con el valor 200 por lo que el periodo de la señal de salida es de 3200 μ S ($200 * 16$). El registro CCPR1L se carga con el valor binario procedente de las entradas RA0-RA5 y puede variar entre 0 y 63. De esta manera la anchura del pulso oscila entre 0 y 1024 μ S ($64 * 16$). Se recomienda emplear un osciloscopio conectado a la salida RC2/CCP1 para comprobar el correcto funcionamiento de la aplicación. También se recuerda que los pulsadores E4 y E5 conectados en RA4 y RA5, cuando están en reposo proporcionan nivel lógico "1". Tenerlo en cuenta a la hora de introducir el valor binario de entrada.

Por último indicamos que para realizar estas medidas y otras de los sucesivos ejemplos, hemos empleado el osciloscopio **PoScope basic 2** de la firma PoLabs (www.poscope.com) y de la cual somos distribuidores. Se trata de un instrumento de bajo coste, que se conecta al PC mediante uno de los puertos USB. Lo hemos elegido como herramienta de trabajo por su bajo coste, reducido tamaño, emula diferentes instrumentos y lo podemos conectar al mismo PC (de sobre mesa o portátil) con el que estamos realizando los presentes ejemplos.

3.3.49 Ejemplo 49: Modulaci3n PWM mediante el convertidor ADC

Objetivos

Se trata de modular la anchura del pulso de la se1al de salida que proporciona el m3dulo CCP1 sobre la l3nea RC2/CCP1, en funci3n de la tensi3n anal3gica introducida por RA0/AN0 previa conversi3n.

Esquema

Se muestra en la figura 3-64. Mediante P1 se aplica una tensi3n anal3gica a la entrada RA0/AN0.

Mediante un osciloscopio conectado en la salida RC2/CCP1 podremos visualizar la se1al de salida y las variaciones en la anchura del pulso que se producen.

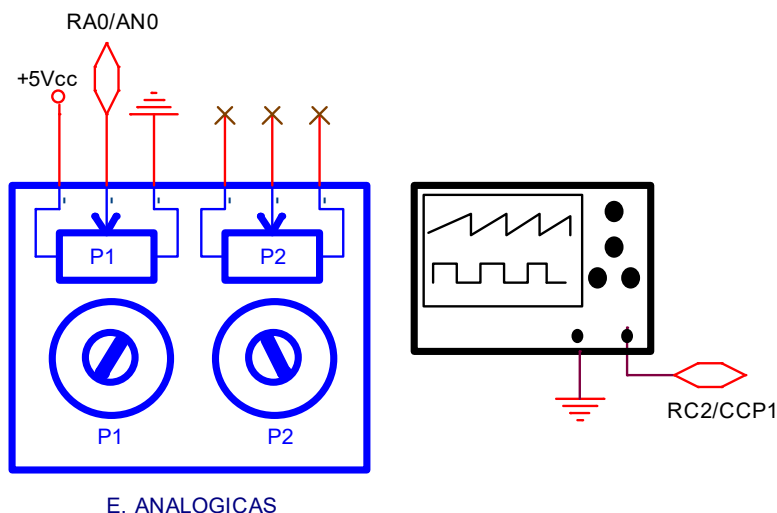


Figura 3-64. Esquema del ejemplo 49

Comentarios

En este ejemplo se muestra el uso combinado del convertidor ACD y el módulo CCP1 en el modo PWM. El convertidor ADC obtiene un valor binario en función de una tensión analógica de entrada (potenciómetro). Este valor se emplea para ajustar el registro CCPR1L y regular así la anchura del pulso de la señal de salida. Con un osciloscopio conectado en la línea RC2/CCP1 se puede apreciar claramente la variación de la anchura del pulso según se varía la posición del eje del potenciómetro tal y como se muestra en la figura 3-65.

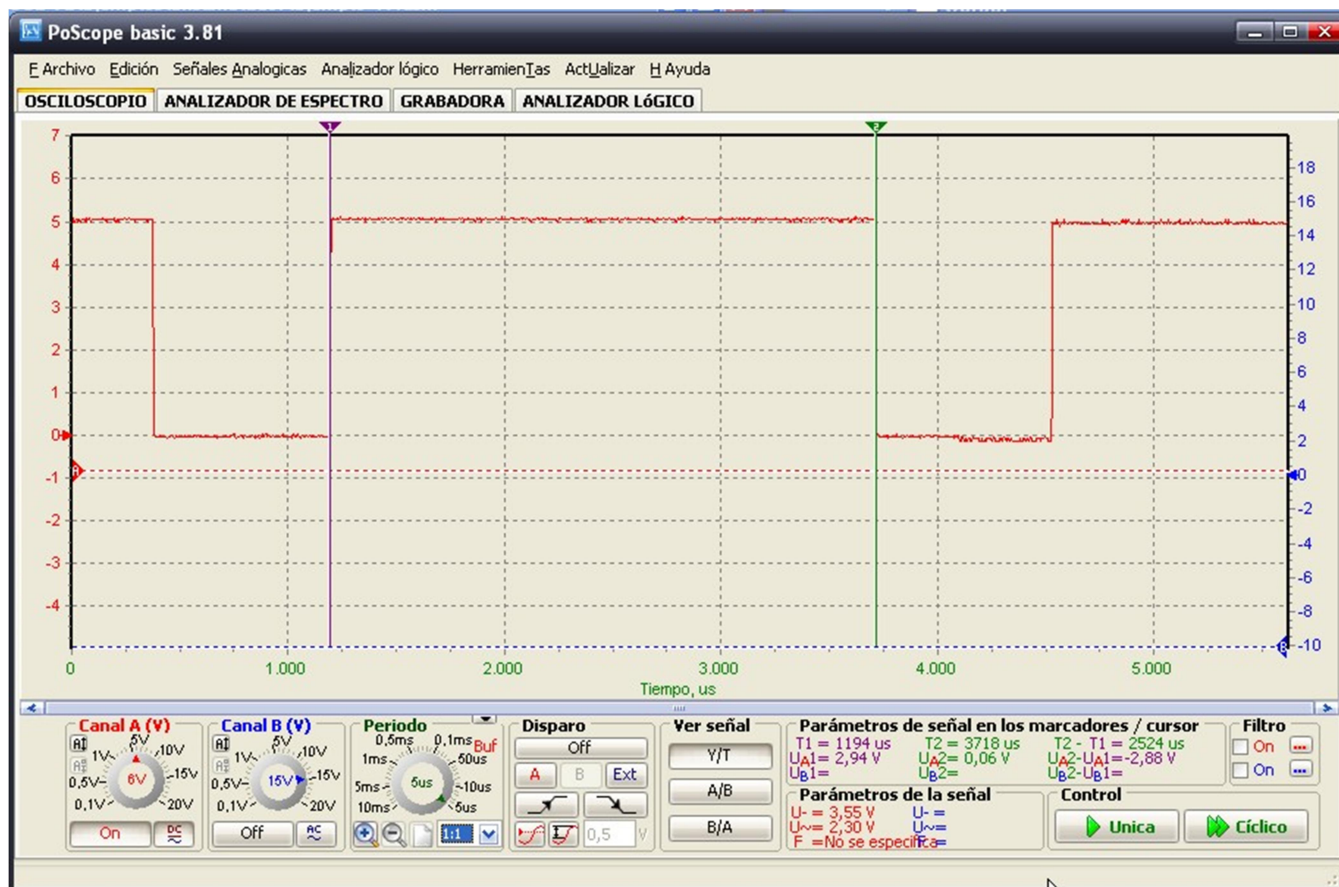


Figura 3-65. Señal PWM regulada mediante potenciómetro

Por último indicamos que para realizar estas medidas y otras de los sucesivos ejemplos, hemos empleado el osciloscopio **PoScope basic 2** de la firma PoLabs (www.poscope.com) y de la cual somos distribuidores. Se trata de un instrumento de bajo coste, que se conecta al PC mediante uno de los puertos USB. Lo hemos elegido como herramienta de trabajo por su bajo coste, reducido tamaño, emula diferentes instrumentos y lo podemos conectar al mismo PC (de sobre mesa o portátil) con el que estamos realizando los presentes ejemplos.

3.3.50 Ejemplo 50: Modulación PWM, modo "Pulse Steering"

Objetivos

El modo "Pulse Steering" es una característica propia del módulo CCP1 mejorado que integran los dispositivos PIC16F88X. Consiste en sacar una misma señal PWM de salida por hasta 4 canales diferentes: RC2/P1A, RB2/P1B, RB1/P1C y RB4/P1C.

Esquema

Como se muestra en la figura 3-66 únicamente es necesario emplear el potenciómetro P1 para aplicar una tensión analógica por RA0/AN0 que regula la anchura del pulso.

Las salidas RC2/P1A, RB2/P1B, RB1/P1C y RB4/P1D proporcionan una misma señal PWM.

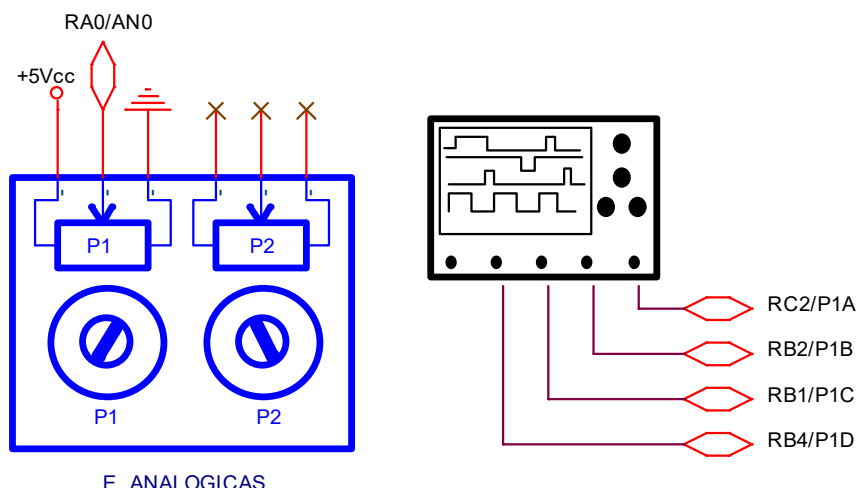


Figura 3-66. Ejemplo 50

Comentarios

En el ejemplo se genera una señal PWM a partir de una tensión analógica por RA0/AN0. Dicha señal PWM se obtiene simultáneamente por las 4 salidas RC2/P1A, RB2/P1B, RB1/P1C y RB4/P1C y puede ser empleada en aplicaciones en que se deban mantener sincronizados diferentes dispositivos (p.e. servos).

Mediante un osciloscopio de doble trazo podemos ir analizando las señales de salida, de 2 en 2, para observar que son idénticas entre sí. También se puede emplear un analizador lógico para visualizar simultáneamente los cuatro canales de salida. Nosotros hemos empleado el modelo **PoScope Basic 2** en el modo analizador lógico de funcionamiento y el resultado obtenido se puede apreciar en la figura 3-67.

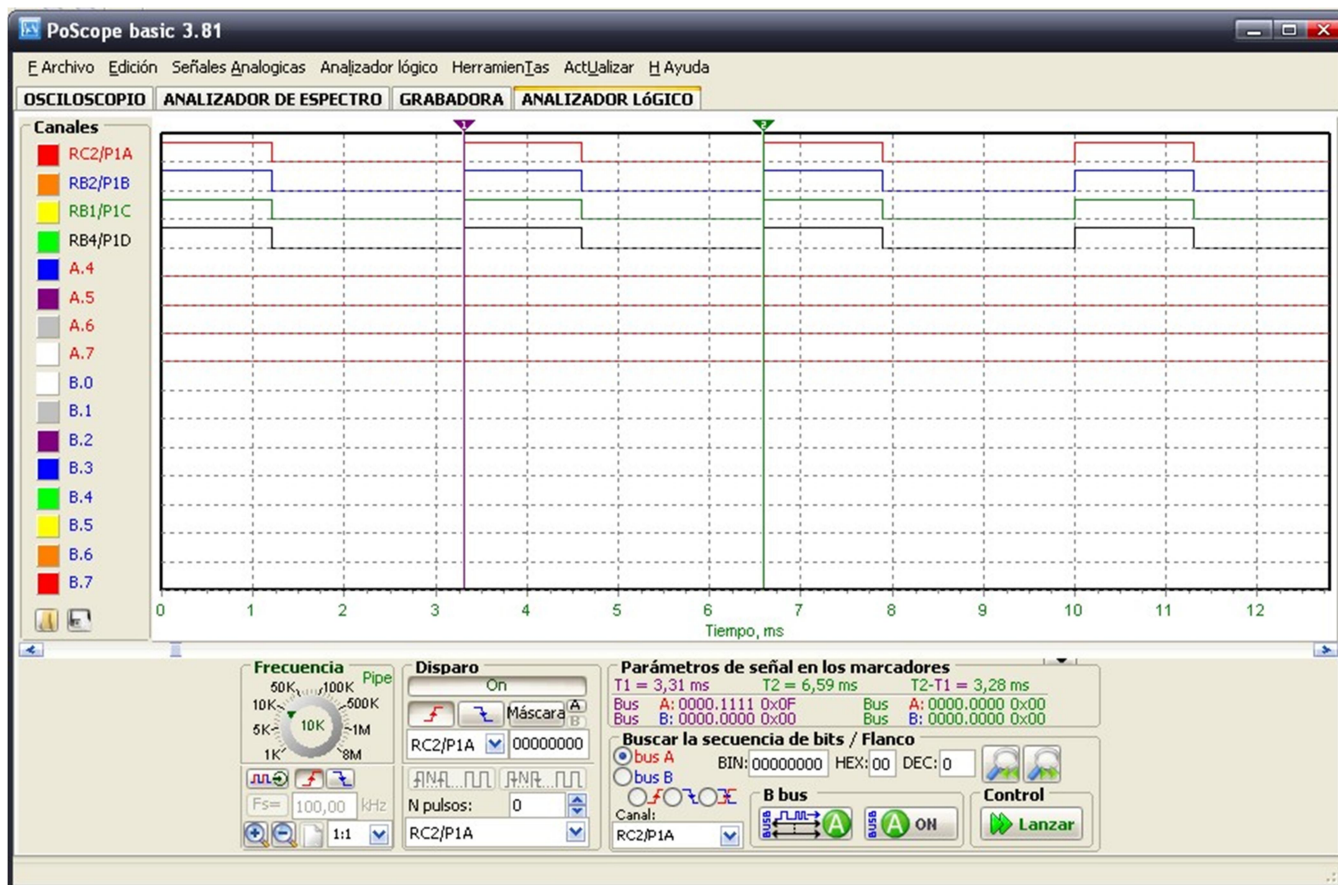


Figura 3-67. Resultado obtenido en el ejemplo 50

Por último indicamos que para realizar estas medidas y otras de los sucesivos ejemplos, hemos empleado el osciloscopio **PoScope basic 2** de la firma PoLabs (www.poscope.com) y de la cual somos distribuidores. Se trata de un instrumento de bajo coste, que se conecta al PC mediante uno de los puertos USB. Lo hemos elegido como herramienta de trabajo por su bajo coste, reducido tamaño, emula diferentes instrumentos y lo podemos conectar al mismo PC (de sobre mesa o portátil) con el que estamos realizando los presentes ejemplos.

3.3.51 Ejemplo 51: Modulación PWM, modo Half-Bridge

Objetivos

Analizar el funcionamiento de este modo de trabajo de la modulación PWM del módulo CCP1 mejorado de los PIC16F88X. El sistema genera dos señales idénticas desfasadas 180° con una modulación regulada mediante un potenciómetro conectado a la entrada RA0/AN0.

Esquema

Se muestra en la figura 3-68. El potenciómetro P1 se conecta con la entrada analógica RA0/AN0 y se aplica la tensión analógica que permitirá ajustar la anchura del pulso. Un osciloscopio de doble trazo o un analizador lógico nos permitirá visualizar las señales en RC2/P1A y RB2/P1B.

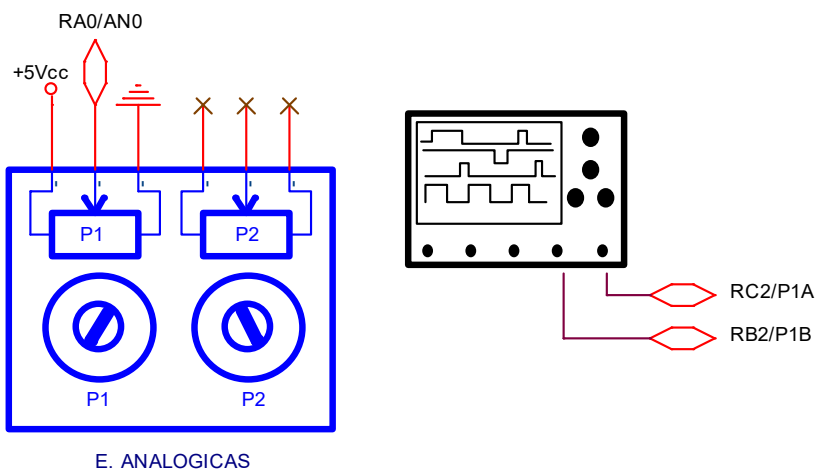


Figura 3-68. Esquema del ejemplo 51

Comentarios

Esta capacidad se aplica sólo al generador PWM del módulo CCP1 mejorado de los dispositivos PIC16F88X. El modo semi puente en H (Half-Bridge). Genera dos señales de salida útiles para activar cargas o dispositivos de tipo push-pull. La señal PWM se obtiene por RC2/P1A y su complementaria por RB2/P1B. Para evitar instantes en que ambas cargas push-pull pudieran estar activadas simultáneamente, con el consiguiente corto circuito que esto pudiera provocar, se puede generar un retardo programable entre ambas señales ("*Dead-Band Delay*").

En el ejemplo se genera una señal PWM a partir de una tensión analógica por RA0/AN0. Mediante un osciloscopio de doble trazo podemos analizar las señales de salida en RC2/P1A y en RB2/P1B, medir el periodo de las mismas, la anchura del pulso y el retardo entre ambas como se muestra en la figura 3-69. Nosotros hemos empleado el modelo PoScope Basic 2 en el modo oscilador de funcionamiento.

La señal superior, en rojo, se corresponde con la señal en RC2/P1A y la azul con la de RB2/P1B. Se aprecia claramente que ambas señales están desfasadas 180° y que la señal en RB2/P1B está retrasada respecto a la de RC2/P1A. Ese retraso se llama "*Dead-Band Delay*" y es programable. Puede evitar daños eléctricos en caso de que dos cargas se activen simultáneamente.

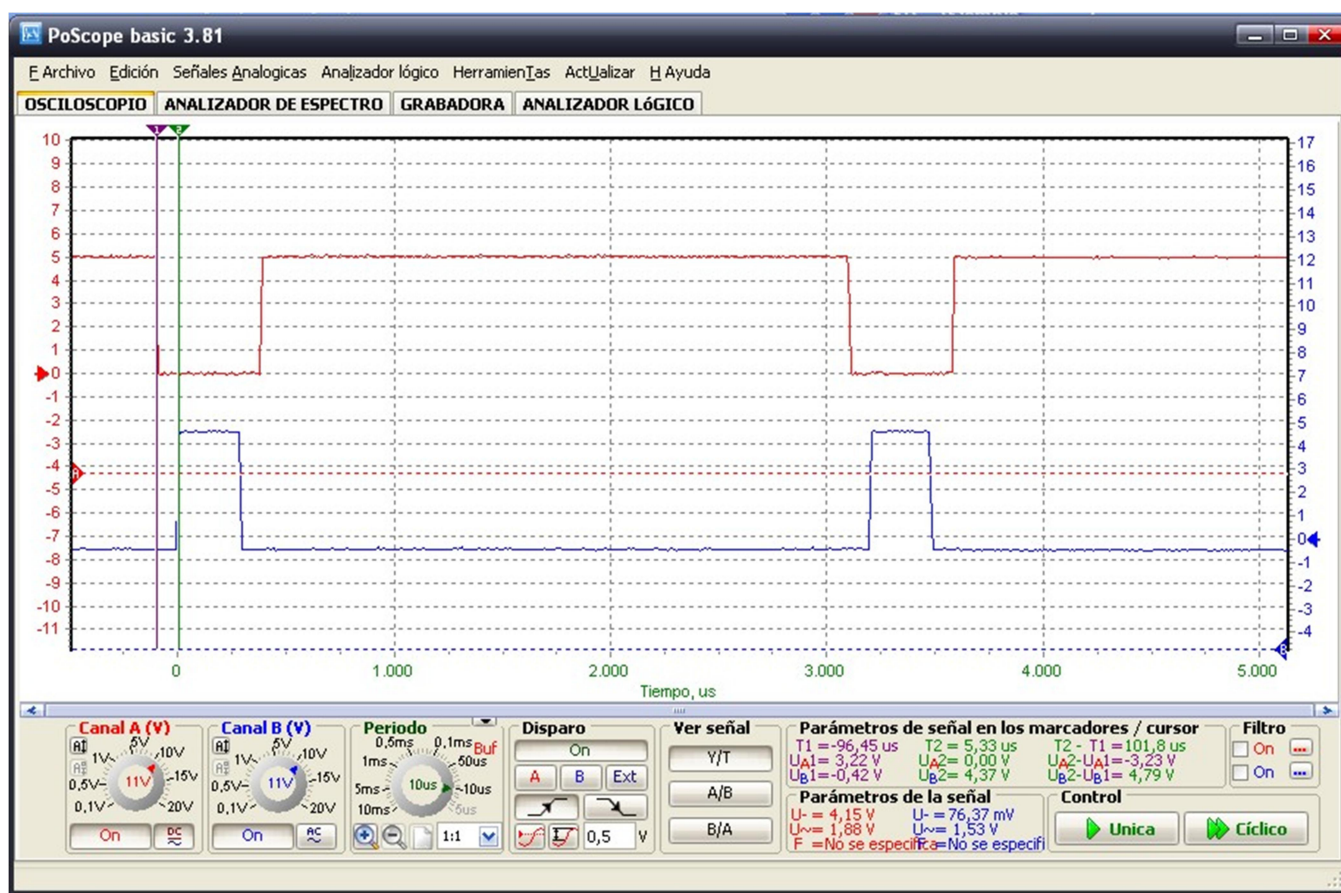


Figura 3-69. Salida de señales para puentes en H (Half-Bridge)

3.3.52 Ejemplo 52: Modulación PWM, modo “Full-Bridge”

Objetivos

El modo puente en H (Full-Bridge). Genera cuatro señales de salida para controlar las cuatro entradas típicas de un clásico puente en H. En el modo directo la salida RC2/P1A se mantiene activa mientras que por RB4/P1D se obtiene la señal PWM. RB2/P1B y RB1/P1C se mantienen inactivas. En el modo inverso la salida RB1/P1C se mantiene activa mientras que por RB2/P1B se obtiene la señal PWM. RC2/P1A y RB4/P1D se mantienen inactivas.

Esquema

El esquema de conexiones que sugerimos para implementar el siguiente ejemplo se muestra en la figura 3-70. El potenciómetro P1 del laboratorio se conecta con la entrada analógica RA0/AN0. A través de él se introduce un valor analógico que, previa conversión, se emplea para regular la anchura del pulso de la señal PWM.

Dicha señal PWM se obtiene por las líneas RC2/P1A, RB2/P1B, RB1/P1C y RB4/P1D y se pueden conectar a un analizador lógico o a un osciloscopio de doble trazo para su estudio y análisis.

Por último el interruptor E0 se conecta con la entrada RA2. Con él se selecciona el sentido de directo o inverso.

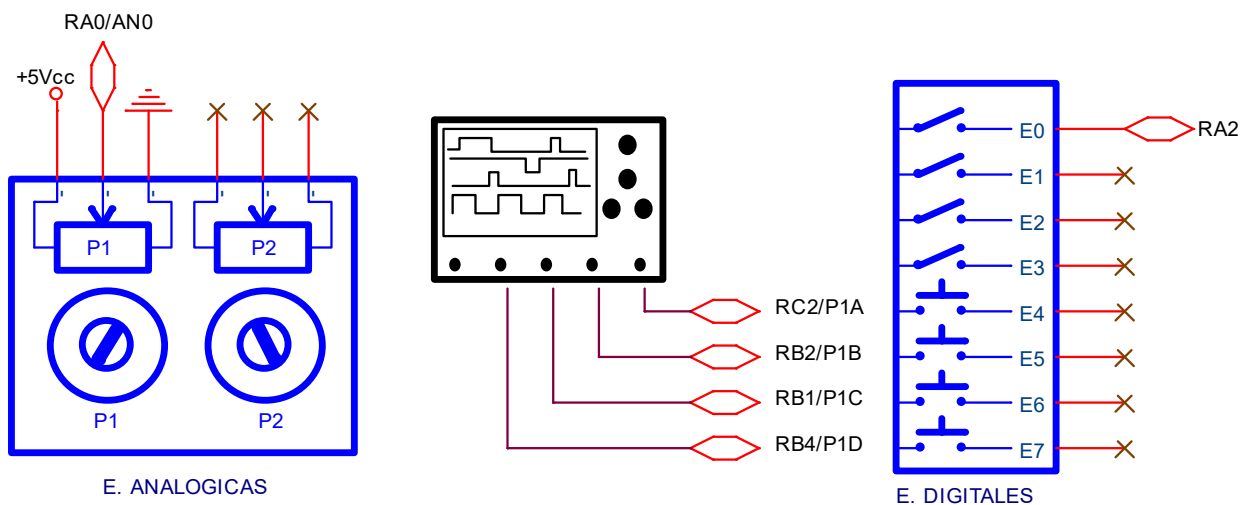


Figura 3-70. Conexiones del ejemplo 52

Comentarios

Esta funcionalidad solo está disponible en el módulo CCP1 mejorado de los dispositivos PIC16F88X.

En el ejemplo se trata de generar una señal PWM a partir de una tensión analógica aplicada por RA0/AN0. Mediante la entrada digital RA2 se controla el modo: a "0" modo directo, a "1" modo inverso del puente H. Con un osciloscopio de doble trazo podemos analizar las señales de salida: en RC2/P1A y RB4/P1D para el modo directo y en RB1/P1C y RB2/P1B para el modo inverso. Como se muestra en la figura 3-71 también podríamos emplear un analizador lógico para visualizar simultáneamente las 4 señales. Nosotros hemos empleado el modelo **PoScope Basic 2** en el modo analizador de funcionamiento.

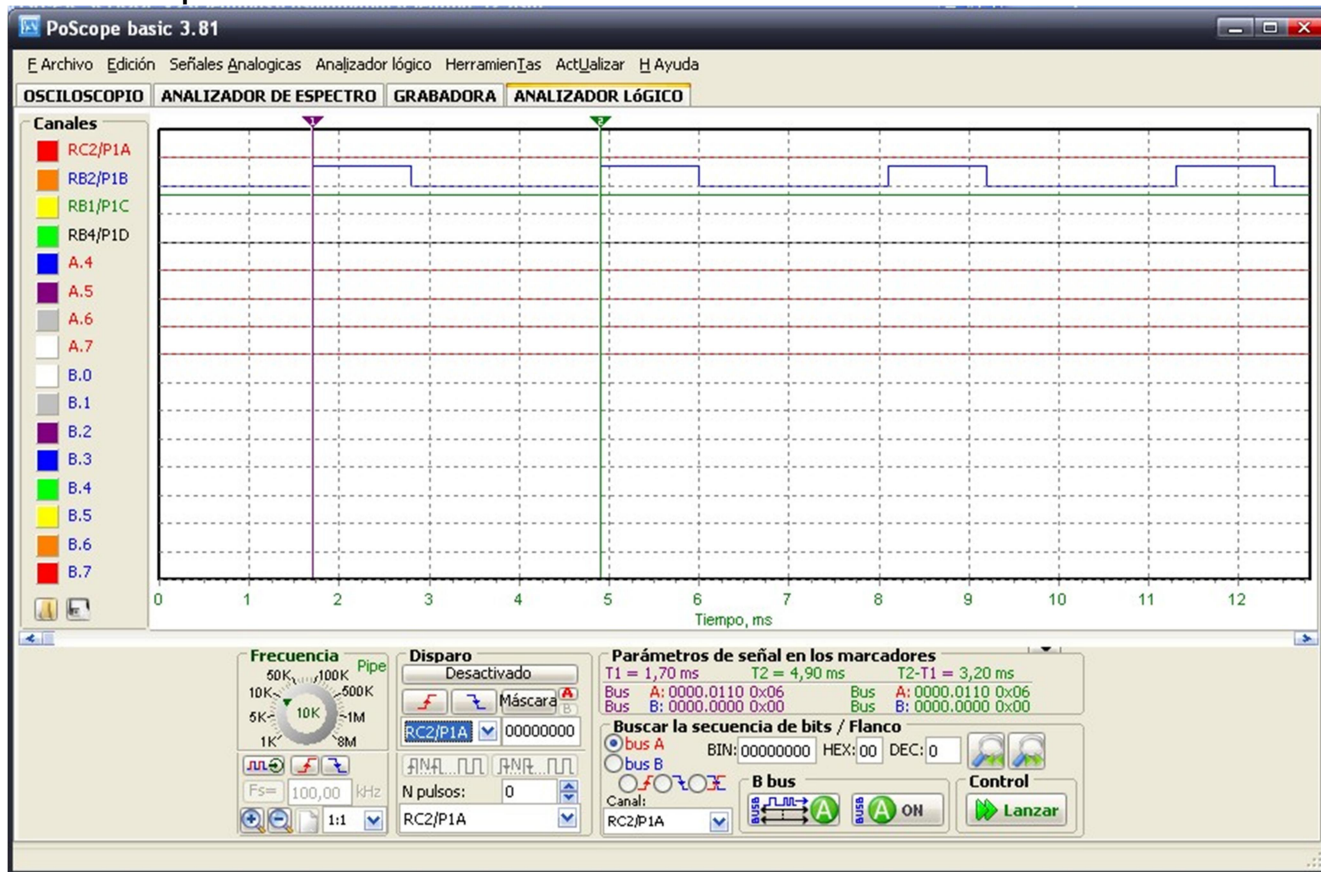


Figura 3-71. Señales de salida para aplicar a un puente en H

A la vista de la figura se puede apreciar que la salida RB2/P1B proporciona señal PWM y la salida RB1/P1C está activa a "1". De la misma manera las salidas RC2/P1A y RB4/P1D están desactivadas a "0". Se está desarrollando por tanto un movimiento de tipo inverso.

Una aplicación directa de este ejemplo la podemos encontrar en el control de giro y regulación de velocidad de un motor DC.

3.3.53 Ejemplo 53: Modulación PWM, modo "Full-Bridge" y "Auto-Shutdown"

Objetivos

Es un ejemplo similar al anterior y en el que se genera una señal PWM para el control de un puente en H. En esta ocasión también se usa el control "Auto-Shutdown" disponible en el módulo CCP1 mejorado de los dispositivos PIC16F88X.

Esquema

Se muestra en la figura 3-72.

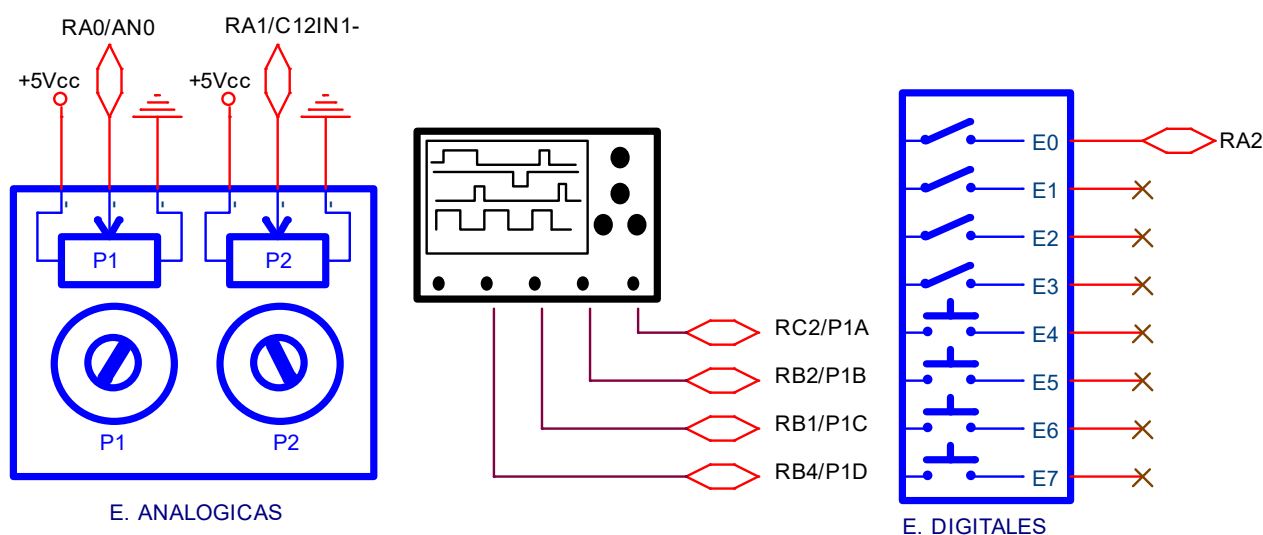


Figura 3-72. Esquema del ejemplo 53

Comentarios

Se trata de un completo ejemplo en el que trabajan conjuntamente varios de los módulos integrados en los dispositivos PIC16F88X y que hemos visto hasta el momento:

- El módulo CCP1 mejorado en el modo PWM de trabajo. Se encarga de generar una señal modulada en anchura a través de 4 salidas y que se puede aplicar directamente a un puente en H. En este ejemplo también se contempla el control "Auto-Shutdown" que permite la desconexión inmediata de las señales de salida en caso de ocurrir un determinado evento.
- El módulo convertidor ADC para convertir la tensión analógica presente en la entrada RA0/AN0 en un valor binario gracias al cual se obtiene la señal PWM de salida.
- El módulo CVREF para generar una tensión de referencia fija que, en el ejemplo, es de 2.5V
- El módulo comparador C1. Este compara la tensión presente en la entrada RA1/C12IN-, procedente del potenciómetro P2 del laboratorio, con la tensión fija de referencia de 2.5V que proporciona el módulo CVREF. Cuando el comparador determine que es mayor, dispara el control de "Auto-Shutdown".

En el ejemplo se genera una señal PWM a partir de una tensión analógica que se aplica por RA0/AN0. Mediante la entrada digital RA2 se controla el modo: a "0" modo directo, a "1" modo inverso del puente H.

También se usa el control *Auto-Shutdown*. Este consiste en desactivar automáticamente todas las salidas cuando ocurre un determinado evento. En este ejemplo el evento es el generado por el módulo comparador C1.

Se compara una tensión analógica externa, aplicada por la entrada RA1/C12IN1-, con la que genera el módulo CVREF y que es fija de 2.5V. Cuando la externa sea superior a 2.5V todas las salidas quedarán desactivadas a nivel "0". También se utiliza la facultad del rearme automático. Este consiste en que las salidas se activan de nuevo automáticamente cuando desaparezca la causa o evento que las desactivó, es decir, la tensión en RA1/C12IN1- sea inferior a 2.5V.

Obsérvese que este ejemplo también tiene una aplicación directa en el control y regulación de motores DC. Efectivamente, la tensión externa que se compara con la de referencia generada por CVREF, puede proceder de un sensor que detecte sobre corriente, sobre peso, exceso de velocidad, etc.. Cuando sea superior a la tensión de referencia, el sistema entra automáticamente en el modo *Auto-Shutdown* desconectando las señales de salida y deteniendo al motor. Este vuelve a ponerse en marcha, *Auto-Rearme*, cuando la tensión externa sea, de nuevo, inferior a la de referencia.

Con un osciloscopio de doble trazo podemos analizar las señales de salida: en RC2/P1A y RB4/P1D para el modo directo y en RB1/P1C y RB2/P1B para el modo inverso. También podríamos emplear un analizador lógico para visualizar simultáneamente las 4 señales. Nosotros hemos empleado el modelo **PoScope Basic 2** en el modo analizador de funcionamiento.

3.3.54 Ejemplo 54: El USART, Transmitiendo caracteres

Objetivos

Emplear el módulo USART disponible en algunos dispositivos PIC, para transmitir el carácter "A" a intervalos regulares de 1 segundo.

Esquema

Conectar la línea RC6/TxD con la señal TxD y la línea RC7/RxD con la señal RxD. Ambas señales se encuentran en el conector AP5 de la sección de Interface RS-232 del laboratorio USB-PIC'School. Ver la figura 3-73.

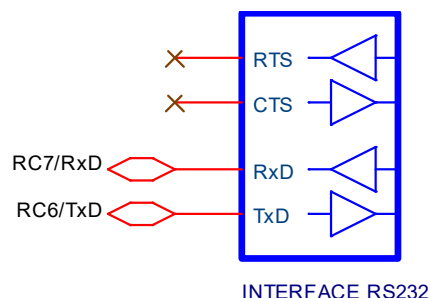


Figura 3-73. Ejemplo 54

Comentarios

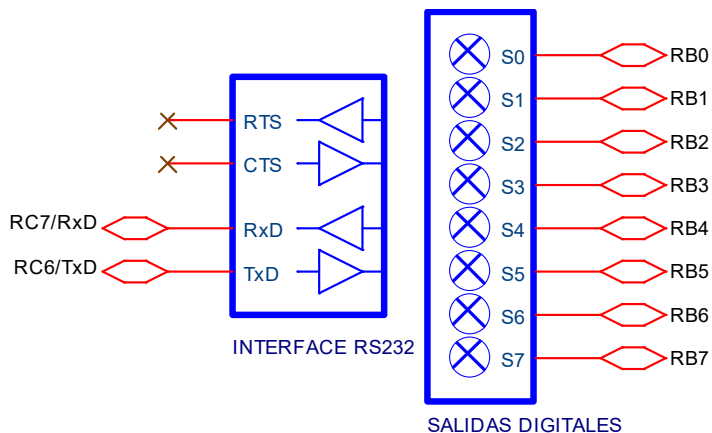
Este ejemplo muestra lo sencillo que resulta realizar la transmisión serie de cualquier carácter o byte. Emplea el USART que integran los PIC de la serie 16F88X entre otros. La recepción se puede realizar sobre un PC dotado de un software estándar de comunicaciones como puede ser el "Hyper Terminal" que viene incorporado en todas las versiones de Windows. Hay que configurarlo para realizar una comunicación a 9600 baudios, 8 bits por carácter y sin paridad. Se puede emplear un cable serie DB9 macho-hembra pin a pin que se conecta entre el conector de interface RS-232 del laboratorio y el canal libre del PC.

3.3.55 Ejemplo 55: EL USART, recepción/transmisión de caracteres

Objetivos

En este ejemplo se realiza transmisión y recepción mediante el USART. El programa espera que se reciba un carácter o byte. Este se visualiza en binario sobre unos leds conectados a las líneas RB0-RB7. Acto seguido el mismo carácter recibido se retransmite a modo de eco.

Esquema



Conectar las líneas RC6/Tx con la señal TxD y la línea RC7/Rx con la señal RxD del laboratorio. Igualmente se conectan los leds de salida S0-S7 con las líneas RB0-RB7 tal y como se muestra en la figura 3-74.

Figura 3-74. Conexiones para el ejemplo 55

Comentarios

En este ejemplo se propone el empleo de una interrupción que se provoca cada vez que se recibe un byte. El programa de tratamiento lo visualiza sobre los leds de salida y lo retransmite. Tanto la transmisión como la recepción se puede realizar sobre un PC dotado de un software estándar de comunicaciones como puede ser el "Hyper Terminal" que viene incorporado en todas las versiones de Windows. Hay que configurarlo para realizar una comunicación a 9600 baudios, 8 bits por carácter y sin paridad. Se puede emplear un cable serie DB9 macho-hembra pin a pin que se conecta entre el conector de interface RS-232 del laboratorio y el canal libre del PC.

3.3.56 Ejemplo 56: Auto detección del USART

Objetivos

Habilitar la capacidad de auto detección de baudios que tiene el módulo USART mejorado de los dispositivos PIC16F88X. El ejemplo, una vez sincronizado, se limita a retransmitir el byte o carácter recién recibido.

Esquema

Se trata del mismo esquema que se empleó en el ejemplo 55

Comentarios

Una nueva aportación de la familia PIC16F88X frente a sus antecesores de la familia PIC16F87X, es la capacidad que tiene el USART para detectar automáticamente los baudios con los que se ha de establecer una comunicación.

El ciclo de auto detección se inicia cuando se activa el bit ABDEN del registro BAUDCTL. A partir de ese momento el USART espera a que el terminal remoto le envíe el carácter ASCII 'U' que en binario corresponde con el código 0b01010101, como primer carácter de todos. Cuando se recibe, el USART mide el intervalo de tiempo en cada transición 0-1-0 y a partir de ahí determina los baudios con los que el terminal remoto le transmitió dicho carácter 'U'. Ya está el ajuste de los baudios realizado, el bit ABDEN se pone a "0" y puede comenzar la comunicación propiamente dicha.

En el ejemplo dicha comunicación consiste simplemente en retransmitir los caracteres que se van recibiendo, al tiempo que su código se visualiza en los leds de salida conectados a las líneas de la Puerta B.

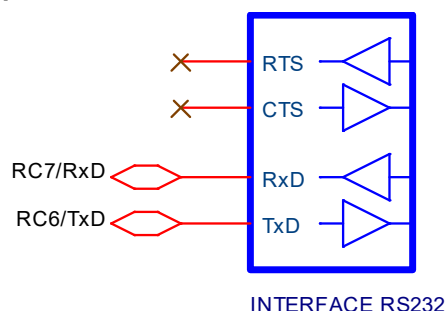
Como terminal remoto se puede usar un PC dotado de un canal serie y de un programa de comunicaciones como puede ser el Hyper terminal de Windows. Debemos recordar que una vez grabado el programa de ejemplo y realizadas las conexiones entre el canal serie del laboratorio USB-PIC'School y el canal serie del PC, tenemos que enviar desde el PC el carácter 'U' como primer carácter antes de empezar la comunicación.

3.3.57 Ejemplo 57: EL USART, auto detección y transmisión de mensajes

Objetivos

Usar la capacidad de auto detección de baudios de la USART mejorada de los dispositivos PIC16F88X, para sincronizarse con el terminal remoto. Nuestro ejemplo se limita a transmitir una serie de mensajes.

Esquema



Se muestra en la figura 3-75

Figura 3-75. Esquema del ejemplo 57

Comentarios

El ejemplo usa la misma capacidad de auto detección que se empleó anteriormente. Una vez realizado el sincronismo se procede a transmitir una serie de mensajes.

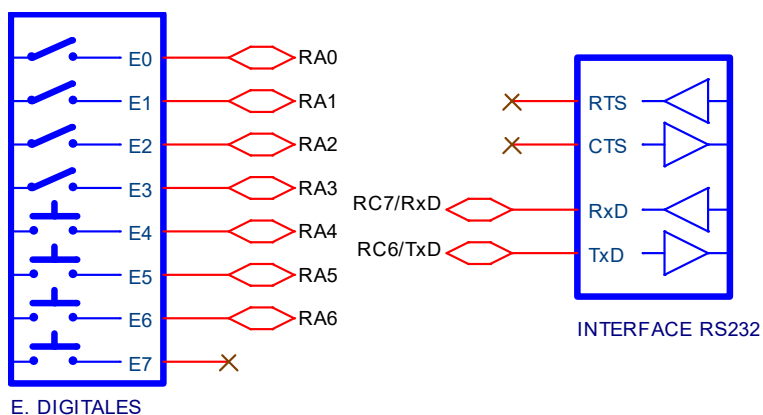
Como terminal remoto se puede usar un PC dotado de un canal serie y de un programa de comunicaciones como puede ser el Hyper terminal de Windows. Debemos recordar que una vez grabado el programa de ejemplo y realizadas las conexiones entre el canal serie del laboratorio USB-PIC'School y el canal serie del PC, tenemos que enviar desde el PC el carácter 'U' como primer carácter antes de empezar la transmisión de mensajes.

3.3.58 Ejemplo 58: EL USART, monitorización remota

Objetivos

Monitorizar de forma remota, sobre el terminal remoto, el estado de las entradas RA0:RA6 conectadas a los interruptores E0:E6 del laboratorio USB-PIC'School.

Esquema



Se muestra en la figura 3-76. Las líneas RA0:RA6 se conectan con los interruptores y pulsadores E0:E6 del laboratorio.

Por su parte la línea RC7/RxD y RC6/TxD se conectan con sus homólogas en el interface RS232

Figura 3-76. Conexiones del ejemplo 58

Comentarios

Se trata de un ejemplo con evidentes aplicaciones prácticas como es el poder monitorizar de forma remota sobre un terminal remoto, el estado de una serie de señales. En el ejemplo se usan sencillos interruptores y pulsadores contenidos en el laboratorio, pero piénsese que en su lugar podría tratarse de sensores de diferente índole, magnitudes analógicas de diferente tipo y un largo etcétera.

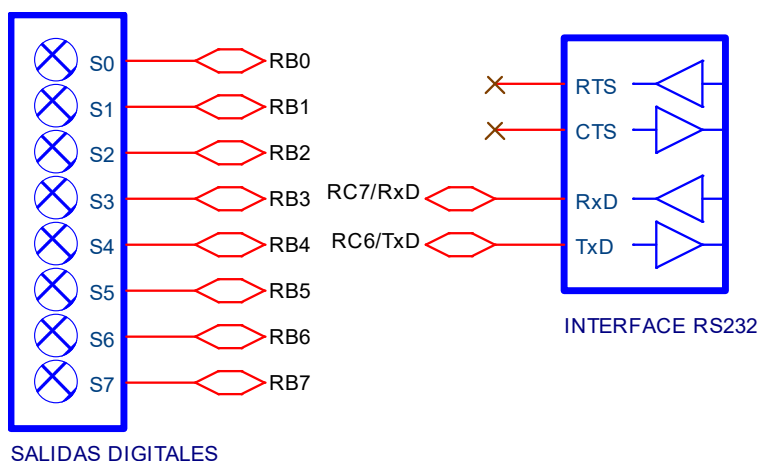
Como terminal remoto podemos utilizar un PC dotado de un software estándar de comunicaciones como puede ser el "Hyper Terminal" que viene incorporado en todas las versiones de Windows. Hay que configurarlo para realizar una comunicación a 9600 baudios, 8 bits por carácter y sin paridad. Se puede emplear un cable serie DB9 macho-hembra pin a pin que se conecta entre el conector de interface RS-232 del laboratorio y el canal libre del PC.

3.3.59 Ejemplo 59: EL USART, Control remoto

Objetivos

De la misma forma que podemos monitorizar de forma remota el estado de diferentes señales, también podemos controlar de forma remota diferentes periféricos. En el ejemplo se trata de controlar y monitorizar los leds del laboratorio conectados a las líneas RB0:RB7.

Esquema



Los leds S0:S7 a controlar se conectan con las salidas RB0:RB7 respectivamente tal y como se muestra en la figura 3-77.

Figura 3-77. El ejemplo 59

Comentarios

Aquí tenemos otro caso con una clara utilidad práctica. Se trata de gobernar desde el terminal remoto el estado de las salidas RB0:RB7 al tiempo que se monitoriza su estado. Desde el terminal se manda una serie de comandos del 0 al 7. Cuando uno de estos comandos es recibido por nuestra aplicación, la salida RB0:RB7 correspondiente cambia de estado. A continuación se retransmite al terminal el estado actual de todas las salidas para su monitorización.

También podemos pensar en este ejemplo que, en lugar de activar o desactivar los leds del laboratorio, podemos controlar de forma remota cualquier otro tipo de periférico.

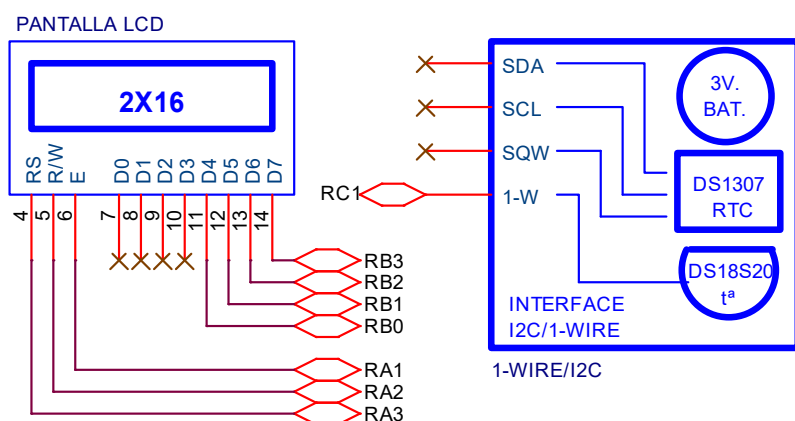
Como terminal remoto podemos utilizar un PC dotado de un software estándar de comunicaciones como puede ser el "Hyper Terminal" que viene incorporado en todas las versiones de Windows. Hay que configurarlo para realizar una comunicación a 9600 baudios, 8 bits por carácter y sin paridad. Se puede emplear un cable serie DB9 macho-hembra pin a pin que se conecta entre el conector de interface RS-232 del laboratorio y el canal libre del PC.

3.3.60 Ejemplo 60: Protocolo y comunicación 1-Wire

Objetivos

Presentar las nociones básicas y funciones relacionadas con el protocolo de comunicación serie conocido como "1-Wire". En este caso se trata de leer el código ID del dispositivo 1-wire DS18S20 (sensor de temperatura) disponible en el laboratorio USB-PIC'School y visualizarlo sobre la pantalla LCD.

Esquema



El esquema de montaje se muestra en la figura 3-78. El dispositivo sensor de temperatura DS18S20 se comunica con el controlador a través de un único hilo (como todos los dispositivos 1-wire). En el ejemplo usamos la línea RC1 para la comunicación 1-Wire con el DS18S20.

La pantalla LCD se conecta como lo hemos venido haciendo hasta ahora.

Figura 3-78. Esquema del ejemplo 60

Comentarios

El protocolo 1-wire de comunicación es una de la formas más versátiles de comunicar nuestro PIC con otros dispositivos como es el sensor de temperatura DS18S20 que nos ocupa. **¡¡ Sólo se necesita un hilo !!**

Todos los dispositivos 1-wire, por normativa, deben incluir un número ID único de 8 bytes que lo identifica de todos los demás. Este ejemplo consiste en leer dicho número ID y visualizarlo sobre el LCD.

La familia PIC16F88X no integra ningún módulo hardware para realizar una comunicación según la norma o protocolo 1-Wire. Habrá que implementarlo por software. En este caso empleamos la línea RC1 como línea de bus para esa comunicación.

En primer lugar necesitamos unas librerías o rutinas genéricas que realicen las tareas más comunes en la comunicación 1-wire. Dichas rutinas diseñadas por MK Electrónica están incluidas en el fichero "**1-wire_PIC16.inc**" del CD-ROM que se entrega con el equipo.

Para poder hacer uso del fichero **1-wire_PIC16.inc** nuestros programas fuente deben incorporar obligatoriamente las siguientes directivas (analizar el programa fuente del ejemplo que nos ocupa):

1. **"#define Fosc 4000000"**: Para definir la frecuencia a la que trabaja el controlador, y a partir de la cual se calculan los diferentes valores de tiempo. En el caso del laboratorio USB-PIC'School de serie incorpora un oscilador de 4MHz.
2. **_1Wire_var equ 0x66**: Para definir el inicio de un área libre de memoria RAM donde las rutinas de manejo del bus 1-wire almacenan y manipulan sus propias variables. Esas rutinas emplean 4 posiciones contiguas o variables que, en el ejemplo actual, empiezan en la posición 0x66.
3. **include "1-wire_PIC16.inc"**: Esta directiva incorpora o "pega" en nuestro programa fuente este fichero que contiene las rutinas y funciones básicas del protocolo 1-wire.
4. **call nombre_rutina**: Es la llamada, desde nuestro programa fuente o desde otras librerías, a una de las rutinas para su ejecución. Es posible que antes de llamarlas, algunas rutinas necesiten parámetros de

entrada o salida pasa su correcto funcionamiento. Leer con cuidado los comentarios y explicaciones que acompañan a cada una de ellas en el fichero **1-wire_PIC16.inc**.

En la siguiente tabla se resumen las rutinas más importantes contenidas en el fichero:

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
_1WReset	Ninguno	El bit C del STATUS=0 si el dispositivo 1-wire está presente. En caso contrario C=1	Rutina que genera el pulso de Reset sobre el bus 1-wire y detecta si hay presencia o no de algún dispositivo
_1WRead_Byte	Ninguno	El dato leído en la variable _1W_Dato	Lee del bus 1-wire un byte completo que se devuelve en la variable _1W_Dato
_1WWrite_Byte	La variable _1W_Dato debe tener el dato a escribir	Ninguno	Escribe el byte presente en la variable _1W_Dato sobre el bus 1-wire

El ejemplo emplea estas rutinas para enviar comandos y recibir datos del DS18S20 para luego visualizarlos sobre la pantalla LCD.

3.3.61 Ejemplo 61: Comunicación 1-wire, medir temperatura con el DS18S20

Objetivos

Ejemplo práctico de uso del sensor DS18S20 para medir y visualizar la temperatura

Esquema

Se emplean las mismas conexiones que en el ejemplo anterior

Comentarios

De la misma forma que tenemos unas librerías con funciones de propósito general para la comunicación 1-wire, disponemos de otras rutinas escritas expresamente para el dispositivo DS18S20 en particular. Están contenidas en otro fichero llamado **"DS18S20_PIC16.inc"** e incluido en el CD-ROM que acompaña al equipo.

Para poder hacer uso del fichero **DS18S20_PIC16.inc** nuestros programas fuente deben incorporar obligatoriamente las siguientes directivas (analizar el programa fuente del ejemplo que nos ocupa):

1. **include "1-wire_PIC16.inc"**: El fichero **DS18S20_PIC16.inc** contiene rutinas y funciones que a su vez hacen uso de las rutinas y funciones del fichero **1-wire_PIC16.inc**. Es por ello que primero se debe incluir éste fichero y todo lo que necesite antes de incluir aquel.
2. **DS18S20_var equ 0x71**: Para definir el inicio de un área libre de memoria RAM donde las rutinas propias de gestión del sensor DS18S20 almacenan y manipulan sus propias variables. Esas rutinas emplean 11 posiciones contiguas o variables que, en el ejemplo actual, empiezan en la posición 0x71.
3. **include "DS18S20_PIC16.inc"**: Esta directiva incorpora o "pega" en nuestro programa fuente este fichero que contiene las rutinas y funciones propias para el uso del dispositivo DS18S20.
4. **call nombre_rutina**: Es la llamada, desde nuestro programa fuente o desde otras librerías, a una de las rutinas para su ejecución. Es posible que antes de llamarlas, algunas rutinas necesiten parámetros de entrada o salida para su correcto funcionamiento. Leer con cuidado los comentarios y explicaciones que acompañan a cada una de ellas en el fichero **DS18S20_PIC16.inc**.

En la siguiente tabla se resumen las rutinas más importantes contenidas en el fichero:

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
<i>DS18S20_Convert:</i>	Ninguno	El bit C del STATUS=0 si el dispositivo 1-wire está presente. En caso contrario C=1	Esta función realiza una nueva medida de temperatura
<i>DS18S20_Read</i>	Ninguno	El buffer desde DS18S20_Byte_0 a DS18S20_Byte_8 contienen el resultado de la última medida	La última medida realizada por el sensor DS18S20 se almacena en el buffer de memoria del controlador.
<i>DS18S20_Convert_T</i>	Ninguno	El buffer desde DS18S20_Byte_0 a DS18S20_Byte_8 contienen el resultado de la última medida	Esta función es equivalente a ejecutar las funciones <i>DS18S20_Convert:</i> y <i>DS18S20_Read</i> ejecutando un Delay de 1 seg entre ambas. Que es el tiempo que tarda el sensor en realizar una nueva medida
<i>DS18S20_ASCII</i>	El buffer DS18S20_Byte_0 a DS18S20_Byte_8 tiene la última lectura medida	El buffer DS18S20_Byte_0 a DS18S20_Byte_4 tiene la cadena ASCII	Convierte el valor actual de la temperatura leída en su correspondiente cadena ASCII en formato sxxx.x, donde s es el signo

Por lo demás del ejemplo en cuestión no hay mucho que decir. Emplea las funciones *DS18S20_Convert_T* y *DS18S20_ASCII* para realizar una nueva medida de temperatura, leerla y convertirla en una cadena ASCII para visualizarla directamente sobre la pantalla LCD.

En la versión de este ejemplo escrito en C nos hemos permitido, que la visualización se realice tanto en grados centígrados como en grados Fahrenheit, como se muestra en la fotografía de la figura 3-79 . Recordemos que el lenguaje C no presenta grandes problemas a la hora de resolver todo tipo de cálculos matemáticos.

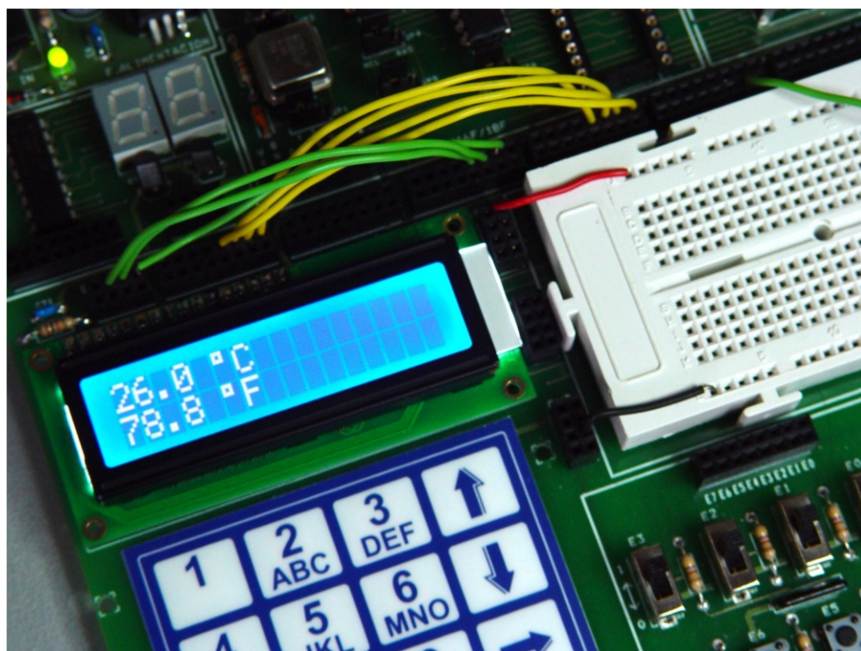


Figura 3-79. Resultado de la ejecución del ejemplo 61

3.3.62 Ejemplo 62: Comunicación 1-wire, termostato

Objetivos

Otro ejemplo práctico de uso del sensor de temperatura DS18S20. Consiste en realizar un termostato que dispare una señal de salida en RB4 conectada al led S7 del laboratorio, cuando se supere una temperatura máxima definida por la constante *T_Max* del programa (p.e. 35°)

Esquema

Básicamente es el mismo que el empleado en los dos ejemplos anteriores al que se le ha añadido un led de salida S7 conectado a RB4 como señal de alarma. Se muestra en la figura 3-80.

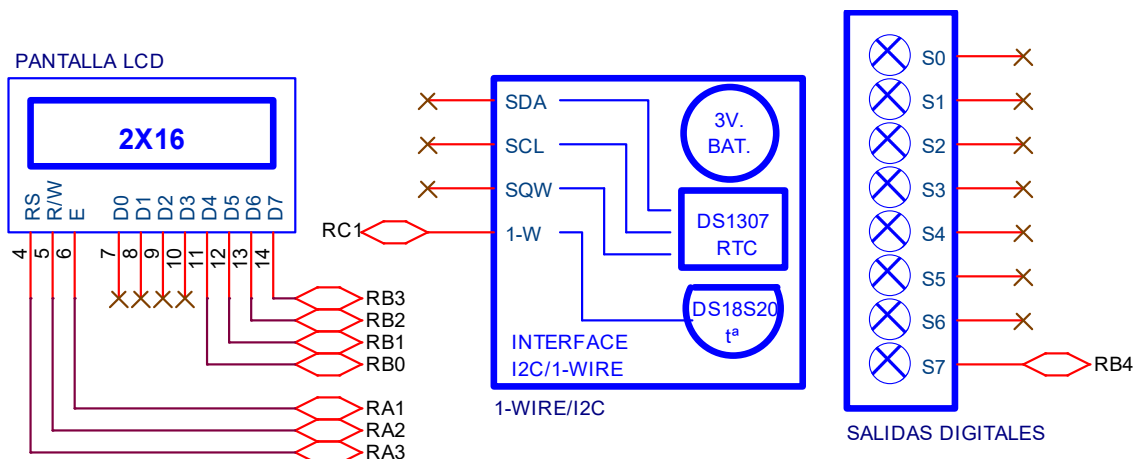


Figura 3-80. Conexiones para implementar el ejemplo 62

Comentarios

No hay mucho que decir. Se trata de un sencillo ejemplo que abre las puertas a infinidad de aplicaciones en las que es necesario un control de temperatura: alarmas, regulación, data loggers, etc...

En este caso la salida RB4 conectada con el led S7 se activa cuando la temperatura medida supera un determinado valor definido por el usuario. La fotografía de la figura 3-81 muestra la ejecución del ejemplo. En un entorno amigable la pantalla LCD nos muestra la temperatura máxima permitida y la temperatura actual que proporciona el sensor DS18S20.

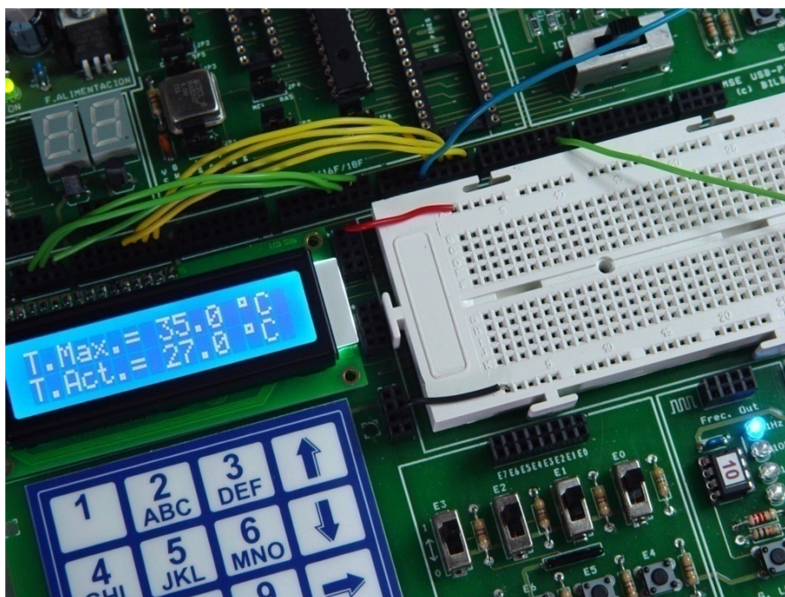


Figura 3-81. Ejecución del ejemplo

3.3.63 Ejemplo 63: Protocolo y comunicación I2C.

Objetivos

Presentar las nociones básicas y funciones relacionadas con el protocolo de comunicación serie conocido como "I2C". En este caso se trata de escribir sobre la memoria RAM del dispositivo I2C DS1307 (reloj/calendario en tiempo real RTC) disponible en el laboratorio USB-PIC'School, una serie de códigos binarios para luego leerlos secuencialmente y visualizarlos sobre unos leds conectados a la puerta B.

Esquema

Se muestra en la figura 3-82. El reloj/calendario en tiempo real (RTC) DS1307 se comunica con el controlador mediante el protocolo del bus I2C, empleando para ello dos únicas líneas que son SDA (datos) y SCL (reloj). Estas se conectan con las líneas RC4 y RC3 del PIC16F88X.

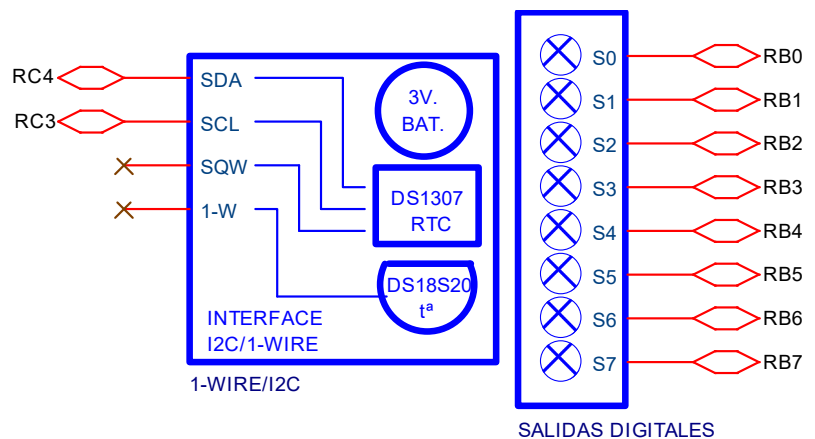


Figura 3-82. Esquema del ejemplo 63

Comentarios

El protocolo I2C de comunicación es otra de las formas más versátiles de comunicar nuestro PIC con otros dispositivos como es el reloj/calendario en tiempo real (RTC) DS1307 que nos ocupa. **¡¡ Sólo se necesitan dos hilos !!**

En el mercado existen un gran número de dispositivos I2C diferentes. Podemos encontrar dispositivos con memorias de diferentes tipos, relojes en tiempo real, convertidores ADC y DAC, controladores de diferentes periféricos y un largo etcétera. Probablemente el protocolo del bus I2C sea el más extendido del mercado.

El bus I2C para la comunicación serie síncrona emplea dos únicas líneas: SDA es la línea de datos y es bidireccional; SCL es la línea de sincronismo o reloj y es unidireccional.

La familia PIC16F88X integra un módulo hardware llamado MSSP (Master Synchronous Serial Port) capaz de realizar comunicaciones serie según las normas o protocolos SPI o I2C. En el laboratorio USB-PIC'School y en los ejemplos propuestos emplearemos el I2C cuyas líneas son RC4/SDA y RC3/SCL.

En primer lugar necesitamos unas librerías o rutinas genéricas que realicen las tareas más comunes en la comunicación I2C. Dichas rutinas diseñadas por MK Electronica están incluidas en el fichero **"I2C_16FXXX.inc"** del CD-ROM que se entrega con el equipo.

Para poder hacer uso del fichero **I2C_16FXXX.inc** nuestros programas fuente deben incorporar obligatoriamente las siguientes directivas (analizar el programa fuente del ejemplo que nos ocupa):

1. **I2C_Var equ 0x60**: Para definir el inicio de un área libre de memoria RAM donde las rutinas de manejo del bus I2C almacenan y manipulan sus propias variables. Esas rutinas emplean 12 posiciones contiguas o variables que, en el ejemplo actual, empiezan en la posición 0x60.
2. **include "I2C_16FXXX.inc"**: Esta directiva incorpora o "pega" en nuestro programa fuente este fichero que contiene las rutinas y funciones básicas del protocolo I2C.
3. **call nombre_rutina**: Es la llamada, desde nuestro programa fuente o desde otras librerías, a una de las rutinas para su ejecución. Es posible que antes de llamarlas, algunas rutinas necesiten parámetros de entrada o salida para su correcto funcionamiento. Leer con cuidado los comentarios y explicaciones que acompañan a cada una de ellas en el fichero **I2C_16FXXX.inc**.

En la siguiente tabla se resumen las rutinas más importantes contenidas en el fichero:

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
<i>I2C_INI</i>	Ninguno	Ninguno	Inicia el módulo MSSP del PIC para trabajar en el modo I2C
<i>I2C_Send_Start</i>	Ninguno	Ninguno	Genera la secuencia de inicio necesaria para comenzar una transferencia de datos
<i>I2C_Sentd_Stop</i>	Ninguno	Ninguno	Genera la secuencia de fin necesaria para finalizar una transferencia de datos.
<i>I2C_Send_Byte</i>	En W el byte a transmitir	Ninguno	Transmite el byte presente en W y espera a recibir el bit ACK
<i>I2C_Read_Byte</i>	Bit 0 de ACK=0 se transmite ACK. Bit 0 de ACK=1 se transmite NACK	En W el byte recibido	Recibe un byte y lo devuelve en W. Genera el bit ACK o NACK en función del bit 0 de la variable ACK
<i>Leer_I2C</i>	Dir_I2C=Dirección del dispositivo I2C_Dir_Ini=Dirección inicial interna I2C_N_Bytes=Nº de bytes a leer	A partir de I2C_Buffer se almacenan los bytes leídos	Lee un número de bytes a partir de una dirección interna del dispositivo direccionado. Los bytes leídos se almacenan a partir de la dirección I2C_Buffer.
<i>Escr_I2C</i>	Dir_I2C=Dirección del dispositivo I2C_Dir_Ini=Dirección inicial interna I2C_N_Bytes=Nº de bytes a escribir A partir de I2C_Buffer los bytes a escribir	Ninguno	Escribe los bytes almacenados en I2C_Buffer sobre el dispositivo direccionado, a partir de una dirección interna, un número determinado de bytes

También son necesarias una serie de funciones o rutinas para el control específico del reloj/calendario en tiempo real (RTC) modelo DS1307. Dichas rutinas hacen uso de las rutinas anteriormente mencionadas y están incluidas en el fichero "**DS1307_16FXXX.INC**" del CD-ROM que se entrega con el equipo.

Para poder hacer uso del fichero **DS1307_16FXXX.INC** nuestros programas fuente deben incorporar obligatoriamente las siguientes directivas (analizar el programa fuente del ejemplo que nos ocupa):

1. **DS1307_Var equ 0x73**: Para definir el inicio de un área libre de memoria RAM donde las rutinas de manejo del DS1307 almacenan y manipulan sus propias variables. Esas rutinas emplean 9 posiciones contiguas o variables que, en el ejemplo actual, empiezan en la posición 0x73.
2. **include " DS1307_16FXXX.INC"**: Esta directiva incorpora o "pega" en nuestro programa fuente este fichero que contiene las rutinas y funciones para el control del RTC DS1307.
3. **call nombre_rutina**: Es la llamada, desde nuestro programa fuente o desde otras librerías, a una de las rutinas para su ejecución. Es posible que antes de llamarlas, algunas rutinas necesiten parámetros de entrada o salida para su correcto funcionamiento. Leer con cuidado los comentarios y explicaciones que acompañan a cada una de ellas en el fichero **DS1307_16FXXX.INC**.

En la siguiente tabla se resumen las rutinas más importantes contenidas en el fichero:

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
<i>DS1307_On</i>	Ninguno	Ninguno	Activa el funcionamiento del RTC DS1307
<i>DS1307_Off</i>	Ninguno	Ninguno	Desactiva el funcionamiento del RTC. Mientras haya alimentación se mantienen todos los datos internos. El consumo se reduce a la mínima expresión (unos 10nA)
<i>DS1307_W_Byte</i>	W=Dato a escribir I2C_Dir_Ini=Dirección interna a escribir	Ninguno	Escribe un byte en cualquiera de las posiciones internas del DS1307 (0x00-0x3F)
<i>DS1307_R_Byte</i>	I2C_Dir_Ini=Dirección interna a leer	W=Dato leído	Lee un byte de cualquiera de las posiciones internas del DS1307 (0x00-0x3F)
<i>DS1307_R_Data</i>	Ninguno	DS1307_Seg= Segundos (00-59) DS1307_Min = Minutos (00-59) DS1307_Hor = Horas (1-12 AM/PM) o (00-23) DS1307_Dia = Día de la semana (1-7) DS1307_Fecha = Día del mes (1-31) DS1307_Mes = Mes (1-12) DS1307_Anno = Año (00-99) DS1307_Control = Registro de control	Lee los 8 bytes del DS1307 que corresponden con la fecha y la hora del dispositivo así como el valor del registro de control. Obtiene por tanto la fecha y la hora
<i>DS1307_W_Data</i>	DS1307_Seg= Segundos (00-59) DS1307_Min = Minutos (00-59) DS1307_Hor = Horas (1-12 AM/PM) o (00-23) DS1307_Dia = Día de la semana (1-7) DS1307_Fecha = Día del mes (1-31) DS1307_Mes = Mes (1-12) DS1307_Anno = Año (00-99) DS1307_Control = Registro de control	Ninguno	Escribe el valor actual de los 8 bytes almacenados en el buffer, sobre el DS1307. Sirve por tanto para la puesta en hora del DS1307
<i>DS1307_Out_0</i>	Ninguno	Ninguno	Salida SQW=0
<i>DS1307_Out_1</i>	Ninguno	Ninguno	Salida SQW=1
<i>DS1307_Out_1H</i>	Ninguno	Ninguno	Salida SQW=1Hz
<i>DS1307_Out_4K</i>	Ninguno	Ninguno	Salida SQW=4KHz
<i>DS1307_Out_8K</i>	Ninguno	Ninguno	Salida SQW=8KHz
<i>DS1307_Out_32K</i>	Ninguno	Ninguno	Salida SQW=32KHz

Centrándonos en el ejemplo actual. Diremos que trata de mostrar cómo escribir y leer datos en la memoria RAM del dispositivo DS1307 siguiendo el protocolo del bus I2C. Se trata de una memoria RAM de propósito general y con la ventaja de que sus datos pueden mantenerse mientras se alimente al DS1307, bien sea desde la alimentación general del sistema o bien mediante una pila. Ambos casos son contemplados por el laboratorio USB-PIC'School. Dado el bajo consumo del dispositivo DS1307 una simple pila como la del laboratorio permite unos 3 años de retención de los datos en RAM.

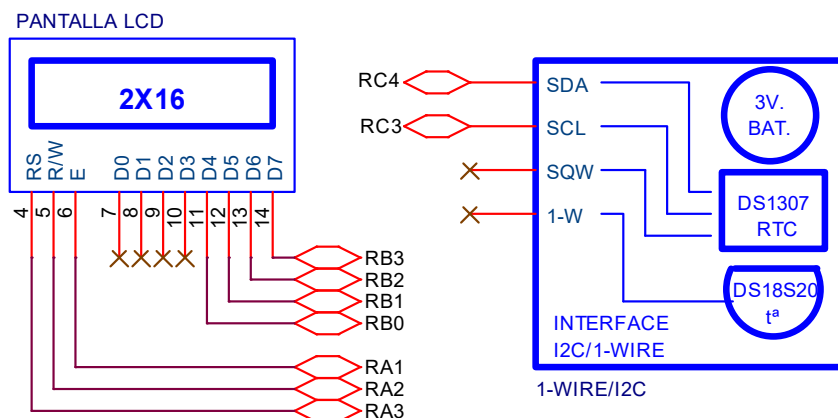
Al principio del programa del ejemplo, la memoria RAM del DS1307 se carga con una serie de valores procedentes de una tabla. Posteriormente se va leyendo secuencialmente y a intervalos de 0.3 seg. cada uno de esos valores al tiempo que se visualizan mediante los leds conectados la puerta B, proporcionando un sencillo juego de luces.

3.3.64 Ejemplo 64: Comunicación I2C, el reloj del DS1307.

Objetivos

Así como en el ejemplo anterior se mostró la forma de guardar y recuperar datos de la memoria RAM del DS1307, en el presente ejemplo se trata de extraer la hora contenida en él y visualizarla sobre el LCD.

Esquema



Se muestra en la figura 3-83. La pantalla LCD se conecta como viene siendo habitual. Por su parte las líneas RC4 y RC3 forman el bus I2C y se corresponden con las señales SDA y SCL respectivamente.

Figura 3-83. Conexiones del ejemplo 64

Comentarios

Una primera aplicación del bus I2C y del dispositivo DS1307 que consiste sencillamente en visualizar sobre la pantalla LCD la hora actual contenida en el dispositivo DS1307. Dicha visualización se realiza en el formato: HH:MM:SS.

Si se conecta por primera vez el dispositivo, lo más probable es que el reloj parta de 0 y se visualice 00:00:00. A partir de ese momento se apreciará el avance del mismo. Si disponemos de una pila observaremos que, a pesar de desconectar la alimentación general y volver a conectarla, el reloj mantiene la hora.

3.3.65 Ejemplo 65: Comunicación I2C, Reloj/Calendario con el DS1307

Objetivos

Presentamos una aplicación práctica real con el dispositivo DS1303. Consiste en un clásico reloj/calendario que se visualiza sobre la pantalla LCD y dispone de los recursos necesarios para ajustar la hora y la fecha.

Esquema

Es el mostrado en la figura 3-84. La pantalla LCD y el dispositivo DS1307 se conectan como en el ejemplo anterior. Además conectamos el pulsador E4 con la línea RB4 que servirá para hacer el ajuste y el pulsador E7 con la línea RB5 para seleccionar.

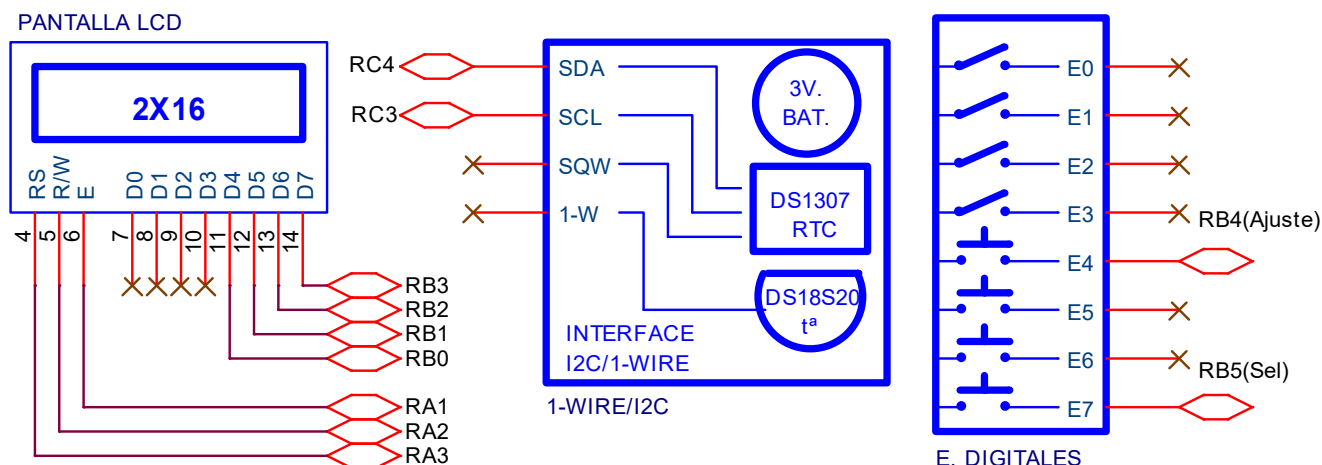


Figura 3-84. Esquema para el montaje del ejemplo 65

Comentarios

Disponemos ya de un ejemplo completo y operativo de un reloj/calendario sobre la pantalla LCD y cuya ejecución se plasma en la fotografía de la figura 3-85. Con el pulsador E7 conectado en RB5 se selecciona el campo que se desea modificar (día, mes, año, etc.). Mediante el pulsador E4 conectado en RB4 se procede a la modificación del campo seleccionado.

El programa del ejemplo presenta un cierto tamaño y complejidad. La mayor parte del programa se consume en realizar un interface amigable entre el usuario y la pantalla LCD así como con los pulsadores de ajuste. Pero no por ello podemos perder de vista que, todas las funciones propias de mantener la hora y la fecha, las lleva el dispositivo DS1307 de forma totalmente autónoma.

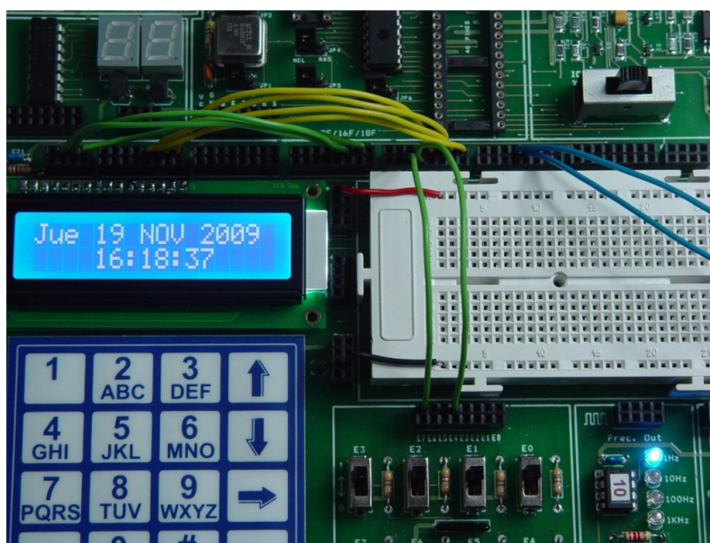


Figura 3-85. Montaje práctico y ejecución del ejemplo 65

Para el ajuste inicial pulsamos E7 sucesivamente. En cada pulsación veremos que el cursor del LCD se va desplazando por los campos a modificar. Cada campo puede modificarse y ajustarse mediante el pulsador E4. El ciclo de ajuste finaliza tras pasar el último campo, el de los minutos. En este momento el reloj/calendario queda ajustado y mantiene la fecha y la hora aunque desconectemos la alimentación general (siempre y cuando esté conectada la batería).

3.3.66 Ejemplo 66: Cartel publicitario

Objetivos

Haciendo uso del reloj/calendario DS1307 y del sensor de temperatura DS18S20 se trata de visualizar por la pantalla LCD la fecha/hora y la temperatura alternativamente, como los carteles publicitarios que podemos encontrar en numerosos municipios.

Esquema

Se muestra en la figura 3-86 y es idéntico al del ejemplo anterior al que además hemos conectado el bus 1-wire del sensor de temperatura DS18S20 con la línea RC1.

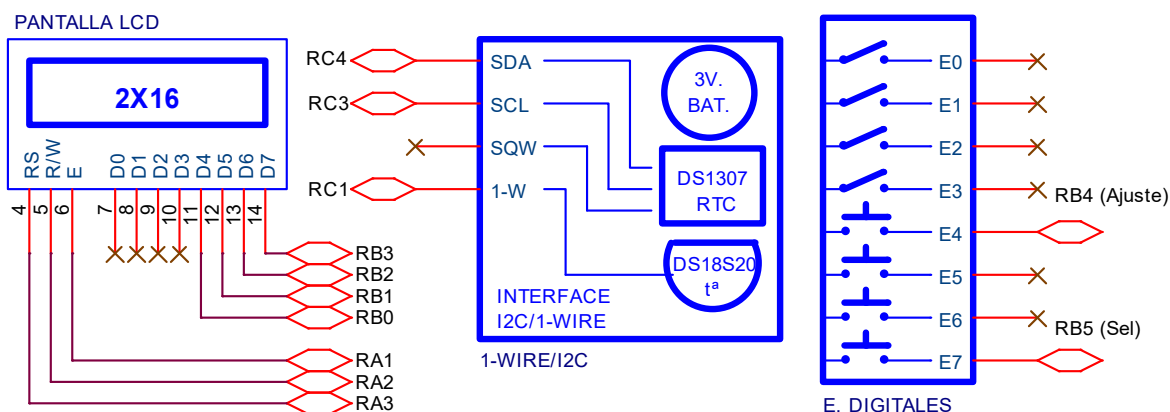


Figura 3-86. Ejemplo 66

Comentarios

Este ejemplo es una consecuencia de los anteriores ejemplos 61 y 65. Empleamos el sensor de temperatura con su interface 1-wire y el reloj/calendario DS1307 con el interface I2C. Hacemos notar que en el ejemplo van a convivir ambos protocolos de comunicación serie.

Sobre la pantalla LCD se visualiza alternativamente la fecha/hora y la temperatura a intervalos de tiempo ajustable mediante la constante "Tiempo_Total" del programa. Esta constante expresa unidades de 0.5seg. y por defecto su valor es de 30 (15 segundos).

La versión de este ejemplo escrita en C visualiza la temperatura en grados centígrados y en grados Fahrenheit.

3.3.67 Ejemplo 67: Data Logger

Objetivos

Realizar una aplicación real consiste en transmitir cada cierto tiempo, vía serie, la Fecha, la Hora y la Temperatura, a modo de registrador de datos o "data logger".

Esquema

Es el mostrado en la figura 3-87. Es parecido al anterior al que hemos añadido además el interface RS232 para transmitir al PC la fecha, hora y temperaturas medidas.

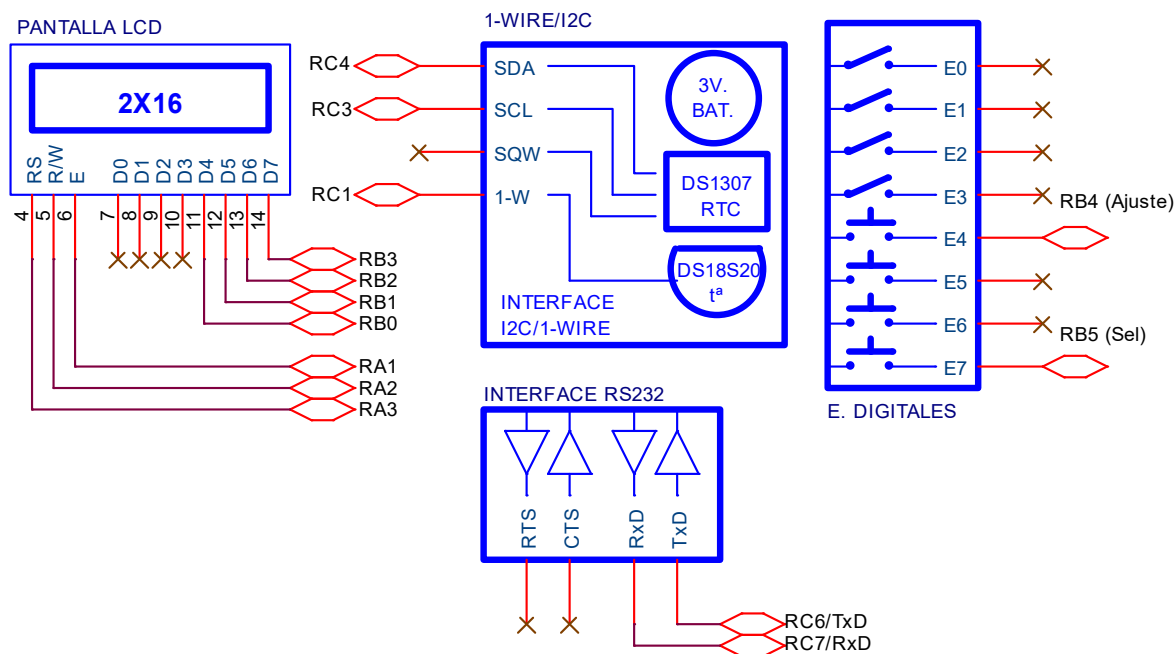


Figura 3-87. Esquema del ejemplo 67

Comentarios

También en esta ocasión se trata de una aplicación basada en ejemplos anteriores. Sobre la pantalla LCD se visualiza la fecha y la hora como si se tratara de un simple reloj digital con sus correspondientes funciones de ajuste.

Cada cierto tiempo se toma una muestra de la temperatura actual y se transmite vía serie junto con la fecha y hora en la que se tomó esa muestra. La transmisión se realiza vía serie mediante el empleo del USART del PIC.

Como terminal remoto podemos utilizar un PC dotado de un software estándar de comunicaciones como puede ser el "Hyper Terminal" que viene incorporado en todas las versiones de Windows. Hay que configurarlo para realizar una comunicación a 9600 baudios, 8 bits por carácter y sin paridad. Se puede emplear un cable serie DB9 macho-hembra pin a pin que se conecta entre el conector de interface RS-232 del laboratorio y el canal libre del PC.

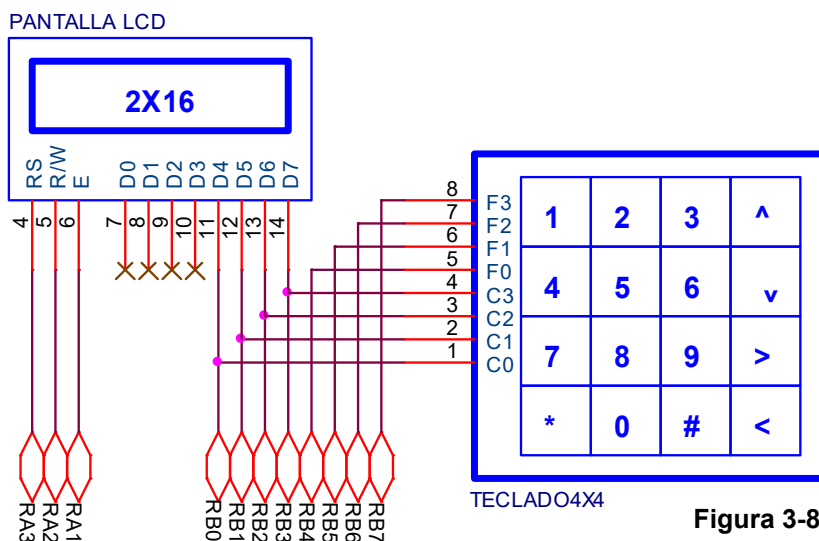
El software Hyper Terminal se puede configurar para capturar y almacenar toda la información recibida sobre un fichero de tipo texto. Luego podremos examinar dicho fichero y hacer un historial de la evolución de la temperatura en base a la fecha y la hora. **¡¡ Todo un completo sistema de adquisición de datos !!**

3.3.68 Ejemplo 68: Entretenimiento, adivina el número

Objetivos

Se trata de un ejemplo de entretenimiento en el que se manejan el teclado y la pantalla LCD. Consisten en adivinar un número comprendido entre 00 y 99. Se dispone de una serie de intentos para conseguirlo

Esquema



Se realizan las conexiones mostradas en el esquema de la figura 3-88 y que ya se ha empleado en anteriores ejemplos.

Figura 3-88. Conexión para el ejemplo 68

Comentarios

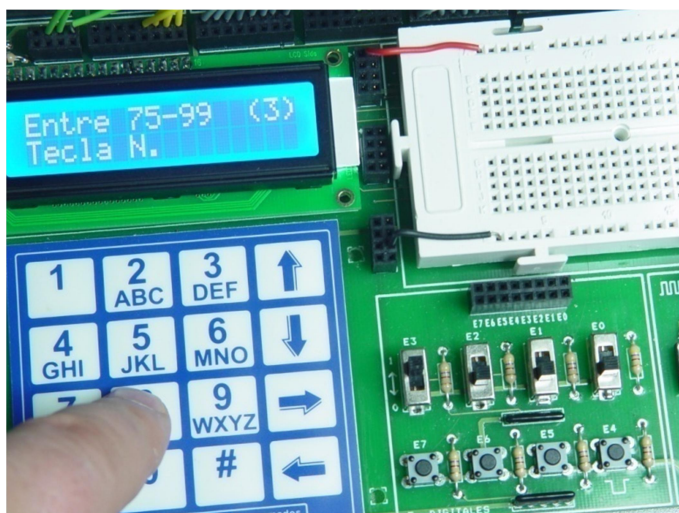


Figura 3-89. Desarrollo del juego

Es un simple ejemplo de divertimento en el que se conjuga el manejo de la pantalla LCD y el teclado. Se emplean una serie de rutinas de propósito general que ya han sido empleadas en múltiples ocasiones. Cabe mencionar la forma en que el programa obtiene un número aleatorio. Para ello emplea el TMR0.

Este timer está constantemente evolucionando, hasta que en un momento se lee su valor y a partir de él se obtiene un número aleatorio entre 00 y 99. Se entiende que es un número aleatorio dado que es imposible predecir el valor que tiene el TMR0 en un momento dado. En la fotografía de la figura 3-89 se aprecia un instante del desarrollo del juego.

3.3.69 Ejemplo 69: Entretenimiento, Control de accesos

Objetivos

Presentar otro ejemplo de entretenimiento que consiste en emular un control de accesos mediante una clave de 4 dígitos que se introduce por el teclado. La pantalla LCD nos mantendrá informados de los pasos a seguir. La línea RA4 se conecta con el led S0 y simulará la bobina de apretura de la puerta. La línea RA5 se conecta con el led S7 y simulará la activación de un pequeño piezo eléctrico a modo de señal acústica o "beep" cada vez que se pulsa una tecla

Esquema

Se muestra en la figura 3-90. Es parecido al del ejemplo anterior al que se le añaden las salidas S0 y S7 controladas por RA4 y RA5

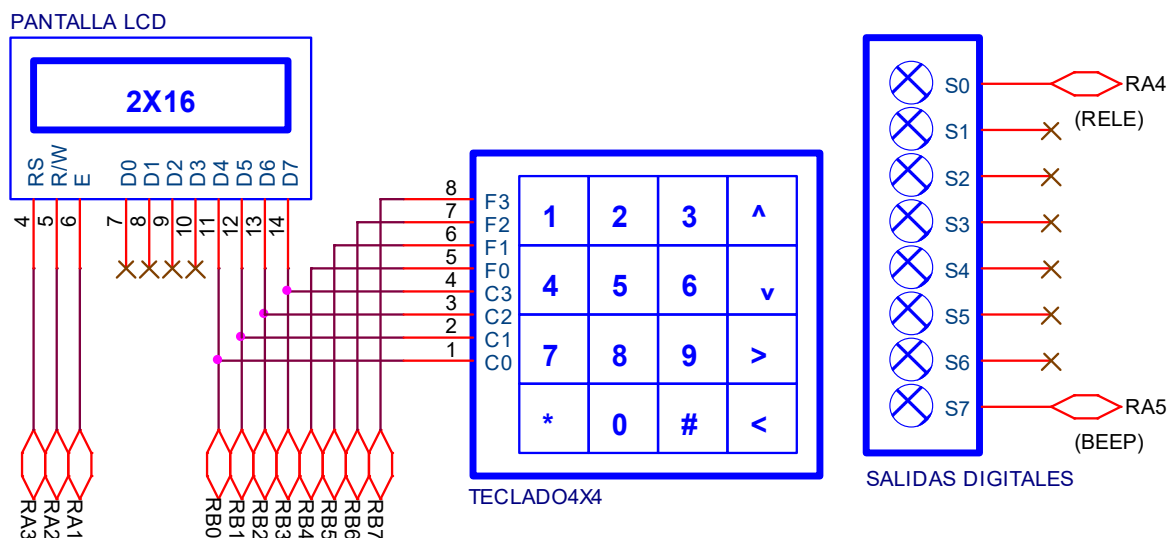


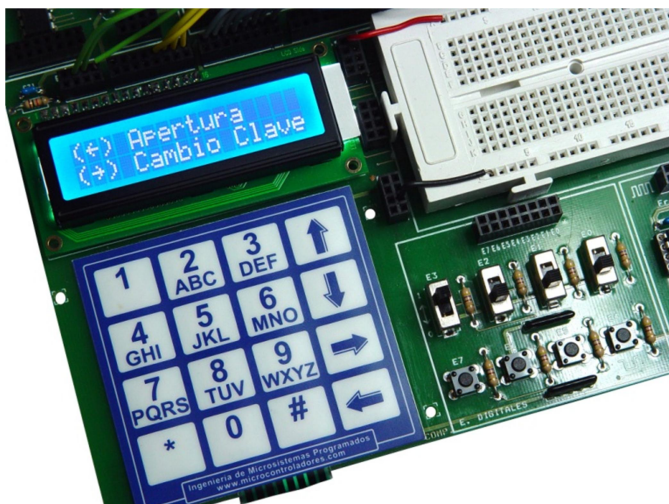
Figura 3-90. Esquema de conexiones del ejemplo 69

Comentarios

El ejemplo no necesita una mayor explicación. Los mensajes de la pantalla LCD son lo suficientemente explícitos para entender el funcionamiento del sistema.

Indicar que es un buen ejemplo de demostración de las posibilidades y uso de la memoria EEPROM de datos de los dispositivos PIC16F88X. Cuando se ensambla el programa y se graba por primera vez, la clave de acceso por defecto es 1234. Esto es así gracias a las siguientes directivas:

```
org    0x2100
de     0x01,0x02,0x03,0x04
```



Estas directivas incluyen en el fichero HEX, la información suficiente como para grabar, además del programa propiamente dicho, a partir de la 1ª posición de la EEPROM (org 0x2100) los bytes 0x01, 0x02, 0x03 y 0x04. La figura 3-91 muestra la fotografía con el desarrollo de la ejecución del presente ejemplo.

Figura 3-91. Ejecución del ejemplo 69

3.3.70 Ejemplo 70: Programa de test

Objetivos

Presentamos uno de los programas de test a los que sometemos a cada uno de los laboratorios USB-PIC'School antes de su comercialización.

Esquema

Se muestra en la figura 3-92 y se corresponde con el montaje de test con el que realizamos una de las pruebas de funcionamiento del laboratorio USB-PIC'School una vez montado en fábrica.

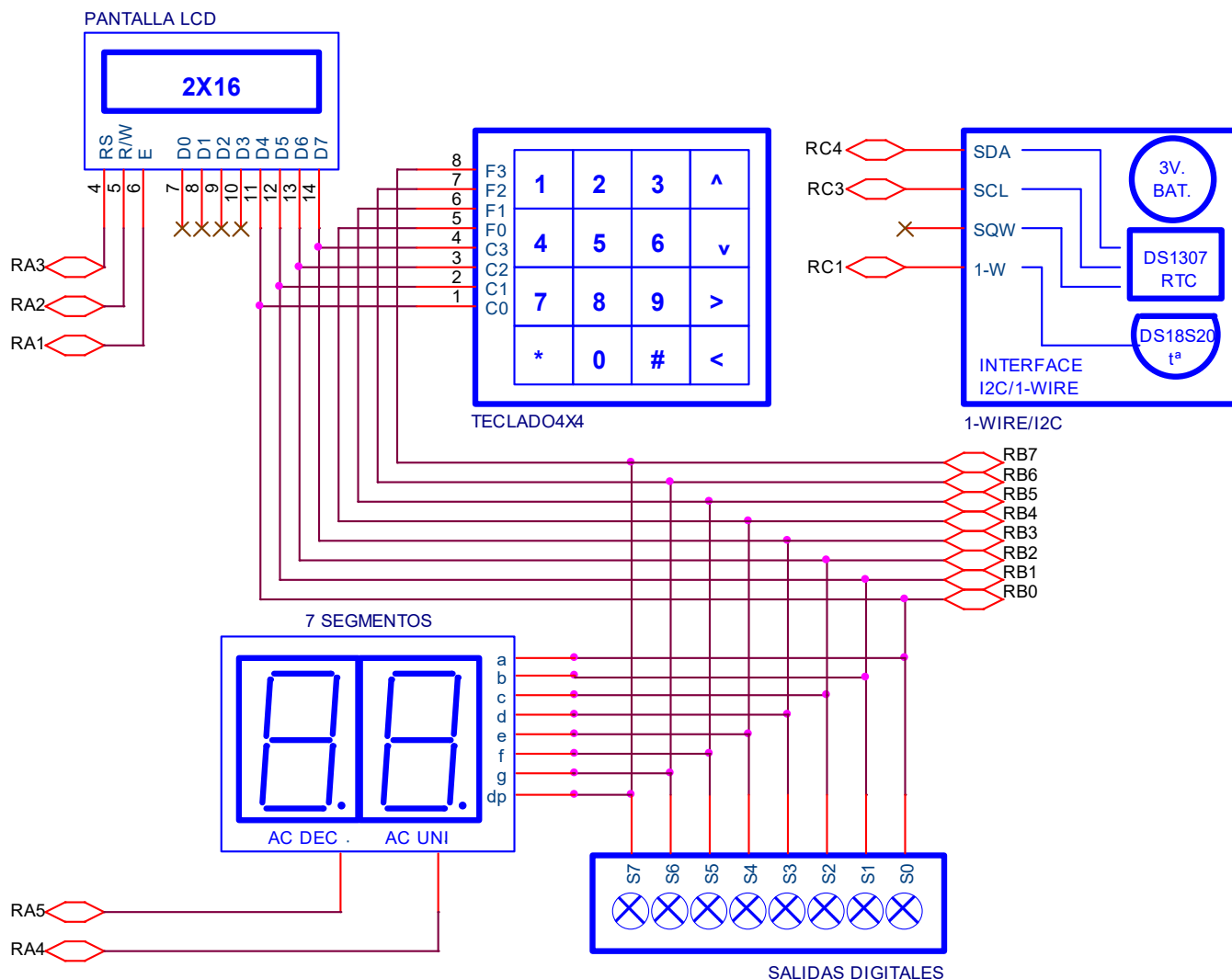


Figura 3-92. Montaje de test

Comentarios

Este programa de test verifica el funcionamiento de algunos de los periféricos de que consta el laboratorio. Realiza una serie de comprobaciones. Empieza por la pantalla LCD que nos va mostrando la prueba que se va a realizar. Se comprueba así el funcionamiento de la misma. A continuación una secuencia binaria por la puerta B nos permite verificar el funcionamiento de cada uno de los leds de salida. De la misma manera se procede con los dos displays de 7 segmentos. Primero se actúa sobre el de las unidades y luego sobre las decenas.

Seguidamente se realiza el test del teclado matricial de 4x4. Este consiste en visualizar sobre la pantalla LCD la tecla que se pulsa. Este test finaliza al pulsa la tecla 0.

Por último se realiza el test del reloj/calendario DS1307 y del sensor de temperatura DS12S20. En la pantalla LCD se visualiza la hora y la temperatura. Esto nos permite a su vez comprobar el correcto funcionamiento del bus 1-wire y el I2C.

